ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Konzeption und Enwicklung einer auf Tablets optimierten mobilen Anwendung für kollaboratives Checklisten-Management

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Andreas Köll
andreas.koell@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Nicolas Mundbrod

2013

Fassung 13. November 2013

proCollab

Satz: PDF-LaTeX 2ε

Kurzfassung

Durch die zunehmende Globalisierung und Auslagerung von Arbeiten an Maschinen und in Niedriglohnländer, wird das spezielle Fachwissen der hochentwickelten Länder zur industriellen Grundlage. Doch wissensintensiven Prozesse, welche sich genau in diesem Bereich finden und die Kollaboration mehrerer Beteiligter erfordern, stellen aufgrund ihrer Komplexität große Herausforderungen in der heutigen Zeit dar.

Kollaboratives Checklisten-Management ist eine Möglichkeit diese Herausforderungen anzugehen und Personen bei ihrer wissensintensiven Arbeit und der gegenseitigen Interaktion zu unterstützen. Im Rahmen des Projekts proCollab soll evaluiert werden, mit welchen Mitteln kollaboratives Checklisten-Management unterstützt werden kann.

In dieser Arbeit wird in diesem Zusammenhang eine mobile Anwendung nach dem Vorgehensmodell des Usability-Engineerings konzipiert, die ein benutzerfreundliches Checklisten-Management auf Tablets ermöglicht. Für diese werden Anforderungen analysiert, ein Gestaltungskonzept entworfen und dieses anschließend in Form eines Prototyps umgesetzt.

Inhaltsverzeichnis

1

Einleitung

Durch die zunehmende Globalisierung nehmen Geschäftsprozesse in hochentwickelten Ländern, wie Deutschland, mehr und mehr an Komplexität zu. Dies liegt daran, dass sich durch die Globalisierung bisher bestehende Arbeitsweisen oder gar ganze Arbeitsstrukturen verändern. So werden Routinearbeiten standardisiert, durch Maschinen automatisiert oder sogar in Niedriglohnländer ausgelagert. In den hochentwickelten Ländern verbleiben jene Arbeiten, die schwierig vorhersehbar sind, sowie ein hohes Maß an Komplexität und Fachwissen voraussetzen. Um diese wissensintensiven Arbeiten zu erledigen, sind meist viele unterschiedliche Fachkräfte beteiligt, die durch kooperatives Vorgehen gemeinsame Probleme lösen. Zu den Bereichen, in denen wissensintensive Arbeiten stattfinden zählen zum Beispiel Entwicklung, Forschung, Entwurf oder Medizin. Wissensintensiven Prozesse in diesen Bereichen stellen in Bezug auf ihre Unterstützung seitens der Informatik eine große Herausforderung dar, sind jedoch für die Produktivität der Unternehmen von entscheidender Bedeutung.

1.1 Problemstellung

Wissensintensive Prozesse werden für den Menschen bei steigender Komplexität zunehmend schlechter handhabbar, da seine mentale Belastbarkeit und Erinnerungsfähigkeit nicht grenzenlos ist. Es entstehen Abhängigkeiten von unterschiedlichsten Ressourcen und Informationen, sowie Abläufen, die koordiniert werden müssen. Jedoch müssen alle Faktoren wie Zahnräder ineinander verzahnt sein, um einen möglichst reibungslosen Betrieb zu gewährleisten.
Die Probleme, die hierbei entstehen können, beginnen daher bereits in den untersten Schichten dieses Konstrukts. Zum Beispiel müssen bei einfachen Wartungsaufgaben von technischen Anlagen eine Vielzahl von Faktoren beachtet werden. Angefangen bei Sicherheitsbestimmungen bei der Arbeitskleidung, über terminliche Abstimmungen und dem Zusammenspiel anderer Anlagen geht es weiter bis zu grundlegenden Schritten, wie dem Abschalten einer Anlage selbst. Das Versäumnis eines dieser Punkte kann im schlechtesten Fall verheerende Auswirkungen haben und hohe Kosten verursachen.
Aber auch in anderen Domänen, wie der medizinischen Versorgung in Kliniken, kommt es zu komplexen, schwer steuerbaren Situationen. Ein einzelner Arzt, beispielsweise, ist für eine Vielzahl von Patienten verantwortlich. Jedem Patienten ist eine Akte zugeordnet, die Personendaten, den Krankheitsverlauf und alle getätigten Aktionen enthält. Je nach Krankheit sind bestimmte Abläufe vorgeschrieben, die einzuhalten sind. Diese können jedoch in Konflikt mit allergischen Reaktionen des Patienten geraten. Vergisst ein Arzt bestimmte Untersuchungen vorzunehmen, oder schlicht den Patienten zu befragen, kann es schnell zu ernsthaften Komplikationen kommen.
Konkret fehlt es also an einer Lösung zur Unterstützung von wissensintensiven Prozessen, die den beteiligten Menschen effektiv ihre Arbeit erleichtert. Checklisten bieten intuitiv eine Möglichkeit zur Unterstützung wissensintensiver Prozesse und werden bereits in sehr vielen Bereichen eingesetzt, um die kognitive Last vom Menschen zu reduzieren. Sie erinnern an Aufgaben und Abläufe, sodass die Gedächtnisleistung nicht mit dem Erinnern belastet ist und Ressourcen für die Erfüllung der eigentlichen Aufgabe zur Verfügung stehen. Der Einsatz und die Verwaltung dieser Checklisten kann jedoch selbst zu einem komplexen Prozess werden, wenn diese manuell eingepflegt

und überarbeitet werden müssen. Sie müssen je nach organisatorischem Rahmen, wie zum Beispiel einem Projekt, angepasst und erweitert werden. Wenn mehrere Endanwender auf den selben Checklisten arbeiten sollen, kommt eine weitere Dimension an Komplexität hinzu: Kollaboration, Kommunikation und der Austausch von Informationen zwischen den Beteiligten.

1.2 Beitrag

Ziel dieser Bachelorarbeit ist es, eine Anwendung für Tablets —*proCollab Tablet (pCT*)— zu konzipieren und zu entwickeln, die die Verwaltung und Nutzung von Checklisten ermöglicht. Sie soll dabei den Lebenszyklus von Checklisten abbilden. Konkret bedeutet dies, dass es Vorlagen gibt, auf deren Basis Checklisten erzeugt, organisiert, abgearbeitet und schließlich archiviert werden können.
Durch pCT gilt es herauszufinden, ob die Komplexität von wissensintensiven Prozessen durch eine benutzerfreundliche Verwendung und Verwaltung von Checklisten für den Anwender reduziert werden kann. pCT soll hierbei auf Tablets optimiert werden, welche eine zunehmende Verbreitung im privaten und auch im geschäftlichen Umfeld erfahren. Sie sind damit eine mobile Alternative zu stationären Desktop-Computern und den weniger handlichen Laptops.
Zur Konzeption dieser Anwendung zählt eine auf Benutzbarkeit fokussierte Anforderungsanalyse und der anschließende Entwurf der Benutzeroberflächen und Systemarchitektur. Auf Basis dieser Vorarbeit soll ein lauffähiger Prototyp der Anwendung implementiert werden, der als Grundlage für Analysen, dem Sammeln von Erfahrungen und künftigen Entwicklungen dienen soll.
Diese Arbeit ist dabei Teil eines Entwicklungsprojekts, das unter anderem die selbe Problemstellung bezogen auf Smartphones, eine webbasierten Administration und das serverseitige Backend[1] für das Checklisten-Management enthält.

[1] Die für den Anwender im Hintergrund laufende Anwendungslogik und Datenbank

1.3 Aufbau dieser Arbeit

Das Kapitel 1 beschreibt die Problemstellung, den Beitrag und mit diesem Abschnitt den Aufbau dieser Arbeit. Das folgende Kapitel 2 führt in die Grundlagen und Begrifflichkeiten des Themas ein. Die darauffolgenden Kapitel orientieren sich im Vorgehen an dem Modell des Usability Engineerings nach [Off13]. In Kapitel 3 wird zunächst die Anforderungsanalyse für pCT durchgeführt. Die Ergebnisse dieser Analyse werden in Kapitel 4 für Gestaltungsentwürfe verwendet. In Kapitel 5 werden anschließend Aspekte der konkreten Implementierung beschrieben. Das Fazit in Kapitel 6 fasst die Arbeit zusammen und schließt diese mit einem Ausblick auf das weitere Vorgehen ab. Grafisch ist der Aufbau der Arbeit in Abbildung 1.1 dargestellt.

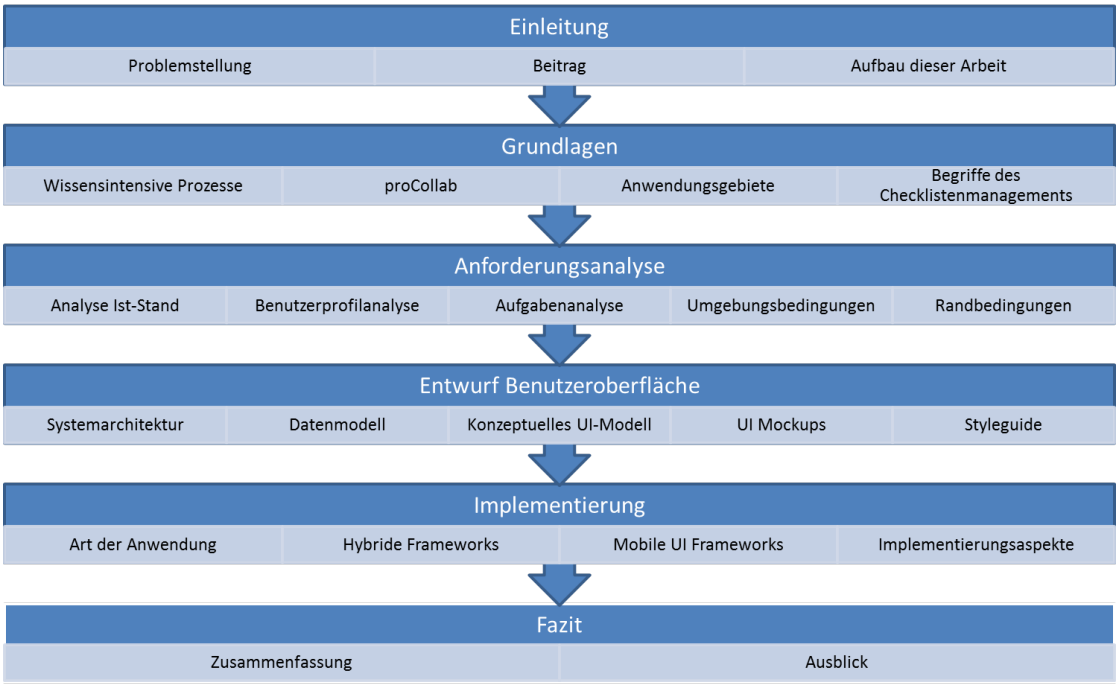


Abbildung 1.1: Aufbau dieser Arbeit über Einleitung, Grundlagen, Anforderungsanalyse, Entwurf, Implementierung und Fazit

2

Grundlagen

Das Kapitel 2 führt in die Grundlagen und Begrifflichkeiten ein, die benötigt werden um die folgenden Kapitel 3–5 zu verstehen. Hierbei werden zuerst die wissensintensiven Prozesse mit ihren Charakteristika erläutert, um die Problemstellung zu detaillieren. Anschließend wird das Projekt proCollab vorgestellt, in dessen Kontext diese Arbeit steht. Über die Anwendungsgebiete von Checklisten wird anschließend zu den Begrifflichkeiten des Checklisten-Managements übergeleitet. Als grafische Übersicht und im Zusammenhang zu den anderen Kapiteln ist dieses Kapitel in Abbildung 2.1 dargestellt.

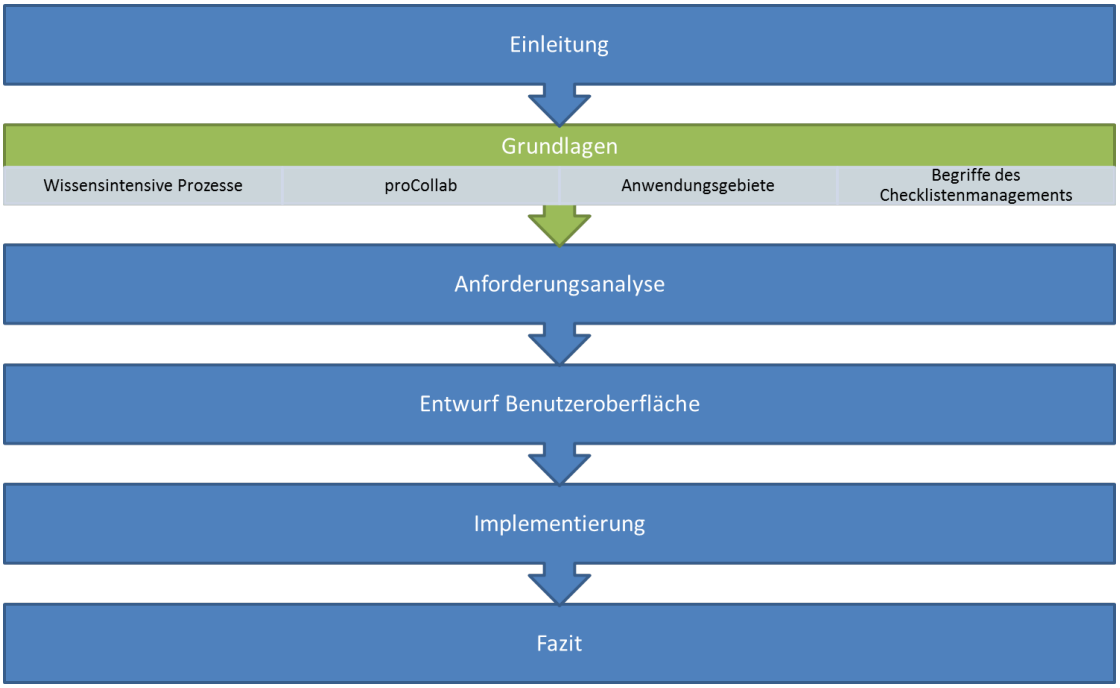


Abbildung 2.1: Aufbau Kapitel 2 — Grundlagen

2.1 Wissensintensive Prozesse

In der heutigen Zeit der Globalisierung werden Prozesse, welche standardisierbar oder automatisierbar sind, entweder in Niedriglohnländer ausgelagert, oder auf Maschinen übertragen. Zu den wissensintensiven Prozessen zählen so zum Beispiel folgende Bereiche nicht:

- Hochautomatisierte Prozesse (Maschinenarbeit)
- Hochstandardisierte Verwaltungsprozesse
- Fließbandarbeit in Produktionen

In den hochentwickelten Ländern, wie zum Beispiel Deutschland, verbleiben Arbeiten, die eben nicht standardisierbar, sondern schwer vorhersehbar sind, einen hohen Grad an Fachwissen benötigen und ein dynamisches Vorgehen nötig machen. Grundsätzlich kommen diese wissensintensive Prozesse in jeder Branche und in jedem Land vor. Denn jeder Manager einer Firma muss Wissensarbeit leisten, um den Betrieb zu verwalten

und letztendlich zum Erfolg zu führen [MKR13].
Ein wissensintensiver Prozess ist ein Prozess, der hoch dynamisch abläuft und dessen Ergebnisse nicht vorhersehbar sind. Fachpersonal entscheidet aufgrund seiner Expertise welche Handlungen durchgeführt werden und über die Reihenfolge dieser [RW12].
Die klassische Domäne der wissensintensiven Prozesse stellt die Forschung und Entwicklung dar. Hier ist ein hoher Grad an Expertise nötig und die Schritte um zum gewünschten Ergebnis zu gelangen sind schwer bestimmbar. Auch dort, wo neuartige und sehr komplexe Arbeitsprozesse der Alltag sind, und Abläufe nicht standardisierbar sind, ist vor der Durchführung des Prozesses schwer oder nicht vorhersagbar wie das Ergebnis ausfallen wird. Eine Person, die ein einem solchen Kontext arbeitet, wird als *Wissensarbeiter* (WA) bezeichnet.
Viele betriebliche Sachverhalte können aufgrund ihrer Komplexität von einem einzelnen WA womöglich nicht eigenständig durchgeführt werden, woraus die Notwendigkeit der Kollaboration mit anderen WA entsteht. Ein solcher wissensintensiver Prozess wird dabei in der Regel nach dem „Divide and Conquer“-Verfahren auf die WA oder Gruppen von WA aufgeteilt. Diese Aufteilung soll Effektivität und Effizienz der WA erhöhen. Jede Unteraufgabe kann im Idealfall unabhängig bearbeitet werden und trägt nach Abschluss zur Lösung des gesamten komplexen Prozesses bei [Mun12].
Für wissensintensive Prozesse können folgende Charakteristika (C) benannt werden [MKR13]:

2.1.1 Unsicherheit (C1)

Komplexität in wissensintensiven Prozessen bezieht sich auf Probleme oder Situationen, die sich aus einer Menge beeinflussender Faktoren zusammensetzen und dynamisch miteinander verknüpft sind. Daraus folgt, dass im wissensintensiven Prozess geplante und tatsächlich durchgeführte Handlungen an mehreren Punkten evaluiert und für das weitere Vorgehen angepasst werden müssen. Der Verlauf der Handlungen ist dabei maßgeblich von den beteiligten WAs, deren Erfahrungen und Kenntnissen abhängig. Zwischen den WAs kommt es zu Kollaboration, aber auch gegenseitiger Störung, was die Dynamik und damit die Unsicherheit des Prozessverlaufs weiter verstärkt.

2.1.2 Zielorientierung (C2)

Die Arbeiten der verschiedenen beteiligten WAs sollten einem gemeinsamen Ziel folgen. Jeder WA hat dabei seine eigene Domäne an Aufgaben, die im Rahmen eines gemeinsamen Ziels eingeschlossen sind. Dazu kann zum Beispiel die eigene Weiterbildung zählen, von der der Prozess anschließend indirekt profitiert. Diese kleineren Aufgaben, mit ihren jeweiligen Zielen, sollen in einer kürzeren Zeitspanne erfüllbar sein und so auch austauschbar und anpassbar bleiben, wohingegen das gemeinsame Ziel vergleichsweise stabil bleibt.

2.1.3 Prozessentstehung (C3)

Arbeiter in wissensintensiven Prozessen müssen ständig ihre Handlungen anpassen, um ihre dem gemeinsamen Ziel untergeordneten Ziele erfolgreich zu erreichen. Durch C1 können sie jedoch nur die nächstfolgenden Handlungen planen. Handlungen, die später auftreten, können angeschnitten, aber noch nicht im Detail definiert werden. Das liegt daran, dass die WAs folgende Handlungen immer in Abhängigkeit absolvierter Handlungen mit den momentan beeinflussenden Faktoren evaluieren. So haben sie in jedem Punkt des Prozesses die Wahl zwischen mehreren möglichen Vorgehen, um ihrem gemeinsamen Ziel näher zu kommen.

2.1.4 Wachsende Wissensbasis (C4)

WAs müssen Informationen in jeder erdenklichen Weise austauschen können, um den Kommunikationsfluss nicht zu beeinträchtigen und andere WAs an ihrem Wissen teilhaben lassen zu können. Am Ende stellt das gesamte gesammelte Wissen eine Lösung für das behandelte Problem und damit das gemeinsame Ziel dar. Ein Teil des Wissens befindet sich in den Köpfen der WAs, der andere Teil stellt die Wissensbasis für den wissensintensiven Prozess dar. Dieser muss in Dokumenten oder Datenbanken verwaltet werden. Der aktuelle Zustand dieser Wissensbasis stellt damit auch den aktuellen Zustand des Prozesses auf den Weg zur Erfüllung des Ziels dar.

2.2 Das Projekt proCollab

proCollab steht für *Process-aware Support for Collaborative Knowledge Workers* und ist ein internes Projekt des Instituts *Datenbanken und Informationssysteme* (DBIS) an der Universität Ulm. Es hat das Ziel, neue intuitive Lösungen zu entwickeln, um die Hindernisse und Probleme der wissensintensiven Prozesse für einen WA zu vermindern [pro13].
Eine mögliche Lösung soll mit dem Checklisten-Management evaluiert werden, für deren Prototyp diese Arbeit ihren Beitrag leistet. pCT stellt hierbei eine Anwendung für Tablets dar, mit der WAs in einem übergeordneten organisatorischen Rahmen Checklisten verwalten und verwenden können. Die von Sabrina Geiger entwickelte Anwendung für Smartphones behandelt die gleiche Aufgabe bezogen auf eine optimierte Oberfläche für diese Geräteklasse [Gei13]. Norman Thiel entwickelt eine webbasierte Administrationsoberfläche für das Projekt [Thi13]. Diese Benutzeroberflächen sind über die von Jule Ziegler entwickelten Schnittstellen und Anwendungslogik [Zie13] mit einer persistenten Datenbank verbunden, die von Daniel Reich entwickelt wird [Rei13]. Genaueres zu den jeweiligen Arbeiten, deren Zusammenhängen und Schnittstellen wird in Kapitel 4.1 erläutert.

2.3 Anwendungsfälle

Checklisten werden in vielen Berufsbranchen bereits verwendet. Dieses Kapitel soll als Einleitung in die Thematik des Checklisten-Managements dienen und ein Verständnis für den Nutzen von Checklisten in komplexen Abläufen in der realen Welt aufbauen. Die Anwendungsgebiete Hilfsorganisationen, Chemieindustrie und Entwicklung mechatronischer Systeme in der Automobilbranche werden in den nächsten Kapiteln vorgestellt.

2.3.1 Hilfsorganisationen

Dieser Anwendungsfall stammt aus einem persönlichen Interview mit einer Sozialdienstleistenden bei einer Hilfsorganisation. Mitarbeiter in diesen Organisationen arbeiten

jeden Tag mit eingeschränkten, kranken oder alten Menschen und tragen die Verantwortung für deren Gesundheit und Sicherheit. Ohne Kontrollabläufe und einem eingewiesenen Personal können hier Menschenleben in Gefahr geraten. Beim Einlernen von neuen Mitarbeitern werden Checklisten deshalb beispielsweise dazu verwendet, sicherzustellen, dass der Einzuweisende alle für ihn wichtigen Personen kennenlernt, und über Regeln und Verbote innerhalb der Einrichtung in Kenntnis gesetzt wird.
Die Checkliste besteht hierbei aus vier Tabellen mit jeweils drei Spalten, gefolgt von den Unterschriften des neuen Mitarbeiters und des Einweisers. Ein Beispiel einer solchen Checkliste ist in Abbildung 2.2 dargestellt. Die erste Tabelle behandelt das Personal der Einrichtung. Hier soll dem neuen Mitarbeiter unter anderem der Werkstatt- und die Gruppenleiter vorgestellt werden. Die erste Spalte enthält dabei den aktuellen Gegenstand der Einführung, hier also die betriebliche Position der Person. Die zweite Spalte dient dem Abhaken des behandelten Gegenstands. Tabelle zwei dient allgemeinen Punkten, wie der Belehrung zur Arbeitssicherheit und dem Brandschutz. Zum Ausfüllen der dritten Tabelle wird ein Rundgang durch die Einrichtung durchgeführt und die besuchten Örtlichkeiten auf der Liste abgehakt. In der letzten Tabelle geht es unter anderem um die Belehrung zur Ersten Hilfe, Medikamenten und dem Standort des nächsten Krankenhauses. Einweiser und neuer Mitarbeiter gehen dabei die Checkliste gemeinsam durch und notieren Bemerkungen, wie die Namen des Personals, Rufnummern oder Probleme jeweils in der dritten Spalte.
Mitarbeiter in Hilfsorganisationen sind jeden Tag neuen, nicht vorhersagbaren Situationen ausgesetzt und müssen ihr Wissen richtig anwenden um diese zu bestreiten. Eine Checkliste sichert in diesem Beispiel ab, dass der neue Mitarbeiter schon bei der Einführung erfährt, an welche Person er sich wenden muss oder wie er das nächste Krankenhaus erreicht.

2.3.2 Chemieindustrie

Ähnlich zu den Checklisten in Kapitel 2.3.1 sind in der Chemieindustrie Checklisten für sogenannte technische Gespräche aufgebaut. Auch diese Anwendungsfälle stammen aus einem persönlichen Interview. Hier geht es um den Start von Ausbauprojekten. Diese

Einführungsbogen

Name: _____ Zeitraum der Einweisung: _____

Einweiser: _____ Telefon: _____

	OK	Bemerkungen
Personal:		
• Geschäftsleiter		
• Bereichsleiter		
• Werkstattleiter		
• Sozialdienste		
• Gruppenleiter		
• BuFDi / FSJ / E-Plus/ Azubi		
• MAV		
• Werkstattrat		
Allgemeines		
Aufgaben des Mitarbeiters / Praktikanten		
Schweigepflicht		
Arbeitszeiten		
Verhalten bei Krankheit und Unfall		
Hausordnung durchgehen und aushändigen		
Raucherregeln		
Getränkeautomaten		
Münzfernsprecher		
Telefonnummer der Gruppe (Karte)		
Terminkalender in der Gruppe		
Gruppenrunde		
Mittagessen / Essensliste / Tischordnung-Begleitung,wer ?		
WC – Hygiene / Binden etc.		
Dienste Plan (Gruppe, Allg.) erklären		
BBB Plan erklären		
Umgang mit Geld / Ausleihen etc.		
Arbeitsbegleitende Maßnahmen		
Mülltrennung		
Fahrdienst / Führerschein		
Arbeitssicherheit		
Brandschutz		
Gebot-/Verbotszeichen		
Pausenaufsicht		
Rundgang		
Gruppenraum		
Umkleide, Spind zuweisen, Sanitär, Personal WC		
Garderobe		
Speisesaal		
Küche (Benutzung)		
Büro SD, Besprechungszimmer		
Berufsbildungsbereich		
Arztzimmer, Gehirnjogging (Benutzung)		
Sanitäre Anlagen (Benutzung)		
Lift (Benutzung)		
Raucherzimmer (Regeln)		

Produktionsräume		
Lagerräume		
Bushalteplatz		
	OK	**Bemerkungen**
Erste Hilfe		
Medikamente		
Diabetiker		
Anfallsleiden		
Hausarzt		
Facharzt		
Nächstes Krankenhaus		

Die Einweisung wurde am _____ beendet.

_____ Einweiser

_____ Unterwiesener

Überschrift einfügen

Abbildung 2.2: Einführungscheckliste einer Hilfsorganisation

behandeln Erweiterungen bestehender chemischer Anlagen oder Neubauten. WAs, die mit der Planung dieser Projekte beauftragt sind, tragen die große Verantwortung, dem Unternehmen durch Fehlplanung keinen wirtschaftlichen Schaden zu erteilten. Aber auch mögliche Gefahren für die am Bau der Anlagen und in laufenden Produktionen arbeitenden Menschen müssen bei der Planung berücksichtigt werden.
Auf der Checkliste zum Start von Ausbauprojekten sind Punkte zu folgenden Themen aufgeführt. Es muss einen Projektgrund geben, ein Konzept ausgearbeitet sein, die Nachhaltigkeit geprüft werden, der Umfang abgeschätzt und ein Kosten- und Finanzierungs- und Terminplan aufgestellt werden. Das Projekt muss außerdem organisiert, sowie Verantwortliche und Manager benannt werden. Jedem dieser Punkte ist wiederum eine Checkliste an Unterpunkten zugeordnet, die abgeklärt werden müssen.
Auch der Veränderung von Anlagen liegt eine Checkliste zugrunde. Hierbei muss der verantwortliche WA verschiedene Bedingungen prüfen, die eine mögliche Auswirkung auf andere Produktionsteile haben. Beispielsweise muss er prüfen, ob sich chemische oder physikalische Bedingungen im Produktionsprozess ändern, welche zu erhöhter Explosionsgefahr führen könnten. Es muss geprüft werden, ob zusätzliche Vorsichtsmaßnahmen festgelegt werden müssen und neue Sicherheitsvorrichtungen nötig werden. Auch der Einfluss der Maßnahme auf die Produktqualität und Energieeffizienz ist zu hinterfragen. Hat der WA die Punkte der Checkliste geklärt, hakt er diese ab oder fügt bei Problemen Bemerkungen zu dem entsprechenden Punkt hinzu.
Fehler in der Planung von Anlagenveränderungen und Erweiterungen bestehender Anlagen können hohe Kosten verursachen und Menschen in Gefahr bringen. Eine sorgfältige Überprüfung aller Risikofaktoren mit Hilfe einer Checkliste mindert die Gefahr, dass Probleme sowie Fehler übersehen werden und damit auch die Wahrscheinlichkeit von Verlusten und Unfällen.

2.3.3 Entwicklung mechatronischer Systeme in der Automobilbranche

Für die Entwicklung von mechatronischen Systemen in der Automobilbranche nutzen beteiligte Entwickler die weit verbreitete Entwicklungsmethode des V-Modells [Mun13]. Die aktuelle Version XT des V-Modells ist in Abbildung 2.3 dargestellt.

Ein Entwicklungsprojekt besteht dabei üblicherweise aus vielen Phasen, für die jeweils feingranularer ebenfalls diese Entwicklungsmethode angewandt wird. Am Ende einer jeder Phase stehen Meilensteine, über deren Abschluss die Phasen miteinander verbunden sind und damit Prozesse vereinigen. Bei Betrachtung des gesamten Projekts führen diese Abhängigkeiten zu vielen verschiedenen parallel ablaufenden Entwicklungsprozessen. Für jedes Projekt wird zu Beginn eine Checkliste erstellt. Diese wird auch im

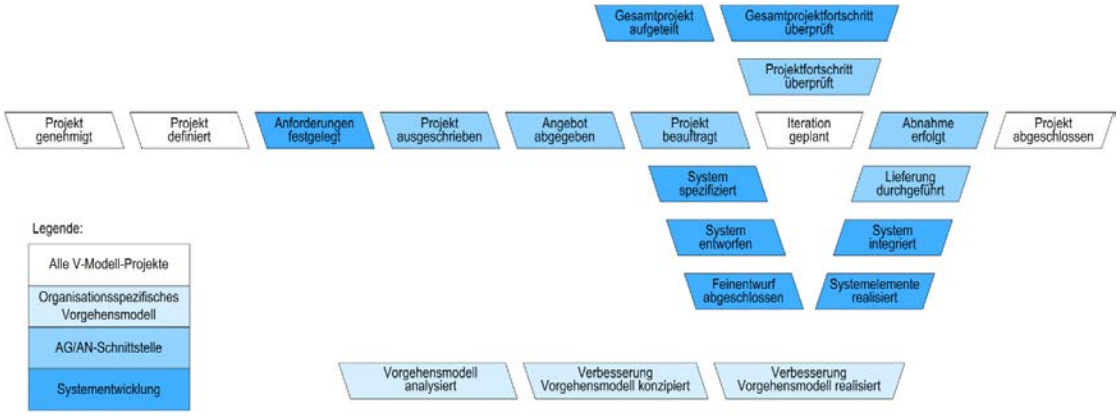


Abbildung 2.3: Entscheidungspunkte in der Projektdurchführung im V-Modell XT 1.4 [IAB12]

weiteren Verlauf des Projekts ständig von einem Verantwortlichen der Qualitätskontrolle angepasst und verwaltet. Durch die wiederkehrende Diskussion der momentan relevanten Einträge der Checkliste haben diese auf die Entwickler einen disziplinarischen Effekt. Die Erstellung einer Checkliste für ein Projekt erfolgt aus dem Export von zentral abgelegten und hierarchisch gegliederten Einträgen. Ein Teil der Einträge gehört dabei zu einer Basismenge, welche für alle Projekte gleich gilt, der andere Teil hängt von den speziellen Anforderungen des Projekts und dessen aktueller Phase ab. Diese zusammengesetzte Checkliste wird in eine Excel-Datei exportiert und zentral für die Entwickler zugänglich gemacht. Für jede Phase des Projekts werden diese Schritte wiederholt.
Die zentral abgelegten Basismengen von Einträgen dienen hier als standardisierte Vorlagen, aus denen sich für die aktuelle Anforderung spezialisierte Checklisten zusammensetzen lassen. Diese Vorlagen werden im Folgenden als *Typen* definiert. Eine fertig zusammengesetzte und exportierte Checkliste, die von den Entwicklern im

Arbeitsprozess verwendet wird, stellt eine sogenannte *Instanz* eines oder mehrerer zusammengesetzter Typen dar.

2.4 Der organisatorische Rahmen

Um ein gemeinsames Ziel zu erreichen, arbeiten WAs in einem *organisatorischen Rahmen* (OR) zusammen. Ein OR kann je nach Anwendungsdomäne unterschiedlich definiert sein: Im technischen Kontext spricht man häufig von einem Projekt, während im juristischen Kontext typischerweise von einem Fall die Rede ist. Auch eine spontane Zusammenarbeit mit einem gemeinsamen Ziel stellt einen OR dar.
Es ist dabei auch möglich, dass es innerhalb eines OR wiederum untergeordnete OR gibt, so wie beispielsweise ein Großprojekt zur Softwareentwicklung in viele kleinere Unterprojekte mit Teilbereichen wie der Entwicklung von Benutzeroberflächen und Datenbanklogik aufgeteilt sein kann.
Da es oft vorkommt, dass sich OR ähneln und somit zu einem allgemeineren Kontext zusammengefasst werden können, wird im Folgenden zwischen organisatorischen Rahmentypen und organisatorischen Rahmeninstanzen unterschieden.

2.4.1 Der organisatorische Rahmentyp

Das Konzept der Basismengen von Einträgen aus dem Anwendungsfall 2.3.3 kann analog auf die OR übertragen werden. Ein solcher OR wird im Folgenden als *organisatorischer Rahmentyp* (ORT) definiert. Ein ORT ist ein OR, der unspezifische Informationen enthält, die sich auf mehrere OR übertragen lassen. Dies kann beispielsweise ein Softwareprojekt sein. Alle Softwareprojekte haben eine Schnittmenge an Abläufen und Informationen, welche in diesem Typ definiert sind. Ein ORT kann nun abgleitet und mit weiteren spezifischen Informationen und Abläufen für den aktuellen OR erweitert werden. Diese abgeleiteten ORT sind als organisatorische Rahmeninstanzen benannt.

2.4.2 Die organisatorische Rahmeninstanz

Eine *organisatorische Rahmeninstanz* (ORI) stellt einen weiter spezifizierten ORT dar. Im Anwendungsfall 2.3.3 wurden Einträge aus der Basismenge zusammengefasst und erweitert, um eine für die aktuelle Projektphase spezifische Checkliste zu erhalten, die von den Entwicklern verwendet werden kann. Übertragen auf den OR und das in 2.4.1 erwähnte Softwareprojekt, wird dieser ORT zu einem spezifischen Softwareprojekt, zum Beispiel „Entwicklung von Programm ABC“. Diese ORI wird mit Informationen und Abläufen erweitert, die speziell für dieses Softwareprojekt gelten.
Eine ORI kann jedoch auch für sich stehen, ohne dass ein ORT zu dieser existiert. Dieser Fall tritt auf, wenn es bisher schlicht keinen passenden ORT gibt, da die Anforderungen an die ORI zum Beispiel sehr speziell sind. Der WA kann so bei Erstellen der ORI alle Parameter dieser selbst definieren ohne auf eine Vorlage zurückzugreifen.
Damit repräsentiert eine ORI den Kontext für wissensintensive Prozesse wie auch die Checklisten, die als koordinatives Hilfsmittel zur Unterstützung genutzt werden.

2.5 Die Checkliste

Die *Checkliste* (CL) wird nach Lessing et al folgendermaßen definiert: Checklisten können „als Arbeitswerkzeug charakterisiert werden, das als Erinnerungshilfe eingesetzt wird oder auch, um Prozesse und Handlungen gleichbleibend zu strukturieren“ [LFKJ+10]. In den Anwendungsfällen in Kapitel 2.3 wurde sie bereits erwähnt und aufgezeigt, wie CLs dem Menschen bei der Bewältigung wissensintensiver Prozesse helfen. Eine CL darf keine umfassende Anleitung sein, sondern ein schnelles und einfaches Werkzeug um die Fähigkeiten eines Fachkundigen zu unterstützen. [Gaw10]
In der Hierarchie ist eine CL dabei immer einem OR untergliedert, welcher den Kontext dieser definiert. Sie wird durch einen Namen beschrieben, der Aussage über ihre Funktion macht und sie enthält Checklisteneinträge.
Analog zu den OR werden bei CL zwei Arten unterschieden: Der generische Checklistentyp und die spezifizierte Checklisteninstanz.

2.5.1 Der Checklistentyp

Analog zu den ORTs (siehe Kapitel 2.4.1) sind *Checklistentypen* (CLT) definiert. Sie stellen eine generische Vorlage dar, welche verwendet werden kann um spezifischere Checklisteninstanzen abzuleiten. Dies kann im Checklisten-Management bei ähnlichen und sich wiederholenden Abläufen Redundanz vermeiden und die Arbeitszeit minimieren. In der Praxis stellt eine CLT innerhalb eines Softwareprojekts beispielsweise die Schritte einer Qualitätssicherung (QS) dar. Diese sind für jedes Projekt ähnlich und in dem CLT zusammengefasst.

2.5.2 Die Checklisteninstanz

Die *Checklisteninstanz* (CLI) ist ähnlich zu der ORI (siehe Kapitel 2.4.2) eine CLT, welche speziell für einen Ablauf spezifiziert wurde. Sie kann aus einem allgemeinen CLT abgeleitet werden, oder wenn kein passender CLT existiert, von dem WA mit allen Parametern frei erstellt werden.
Analog zu dem Beispiel in Kapitel 2.5.1, wo es sich bei der CLT um den Ablauf einer QS in einem allgemeinen Softwareprojekt handelt, stellt die CLI den Ablauf der QS für ein konkretes Projekt dar. Die CLI der QS für dieses Projekt wird dabei mit speziellen Unterabläufen erweitert und den gewünschten Anforderungen angepasst.

2.5.3 Varianten von Checklisten

Da der Begriff der CL unterschiedlich interpretiert werden kann, wird dieser im Folgenden von möglichen Synonymen und Definitionen abgegrenzt.
Der einfachste Typ einer CL ist eine *Aufgabenliste* (To-do-Liste), welche zu erledigende Aufgaben enthält. Diese kann durch weitere Aspekte, wie eine Priorisierung und die Eingrenzung auf Zeiträume erweitert werden.
Eine weiterführende CL wird zum Beispiel in der Luftfahrt verwendet. Hier beschreibt sie die notwendigen Schritte, die von Pilot und Copilot bei regulären Flügen, aber auch Ausnahmesituationen, überprüft und durchgeführt werden müssen.

Der Unterschied zwischen einer Aufgabenliste und einer CL der Luftfahrt liegt also bei ihrem zeitlichen Kontext. Eine Aufgabenliste enthält Aufgaben, welche noch nicht getan wurden und demnach in der Zukunft liegen. Gawande nennt diese entsprechend „Read-Do“-Listen [Gaw10]. Die Aufgabe wird gelesen und anschließend dann erledigt. Die CL hingegen wird bei Gawande „Do-Confirm“-Liste genannt, denn hier wird eine Aufgabe erfüllt und anschließend beim Prüfen mit der CL deren Erledigung bestätigt.
Im Folgenden werden die CL im Sinne der zweiten Anwendung verwendet, obwohl die spätere Konzeption die erste Anwendung nicht komplett ausschließt.

2.5.4 Anforderung an Checklisten

Gawande analysiert verschiedene Einsatzszenarien von CLs und kondensiert darauf aufbauend Anforderungen, die CLs zu erfüllen haben [Gaw10]:

- nicht zu lange
- nicht zu kompliziert
- nicht mehrdeutig
- präzise
- effizient
- auf den Punkt
- leicht zu verwenden
- praktisch

Konkret wird vorgeschlagen, die Länge auf fünf bis neun Checklisteneinträge zu begrenzen, die CL in der Sprache der Anwendungsdomäne zu verfassen und sie frei von unnötigen Zusätzen und Farben zu halten.

2.6 Der Checklisteneintrag

Ein *Checklisteneintrag* (CE) beschreibt eine einzelne Aufgabe einer Checkliste. Er hat folgende Eingenschaften:

- Name: Eine eindeutige Bezeichnung.

- Beschreibung: Die Beschreibung des Eintrags beschreibt die Aufgabe in Form einer Aussage, welche es zu bestätigen gilt, oder einer Frage, die beantwortet werden muss.
- Zustand: Ein boolescher Wert, der angibt, ob die Aufgabe erfüllt ist, oder nicht, oder die Antwort auf eine Frage.

Bei einfachen CLs, in denen Aufgaben abzuhaken sind, wird der Zustand des CE zum Beispiel durch ein Kästchen dargestellt. Andere Arten des CE ermöglichen die Auswahl von einer oder mehreren vorgegebenen Antwortmöglichkeiten, oder lassen eine freie Antwort zu. Welche Art des CE verwendet wird, hängt spezifisch vom OR, der CL, sowie den Vorzügen des Unternehmens und der WA ab.
Wie bei ORs und CLs können hier Checklisteneintragstypen und Checklisteneintragsinstanzen unterschieden werden.

2.6.1 Der Checklisteneintragstyp

Der *Checklisteneintragstyp* (CET) ist äquivalent zu einem Eintrag aus der Basismenge, welche in Kapitel 2.3.3 erläutert wurde. Hierbei handelt es sich um CE, welche für verschiedene CL verwendet werden können.

2.6.2 Die Checklisteneintragsinstanz

Die *Checklisteneintaginstanz* (CEI) ist ein CE, welcher sich in einer CLI befindet. Dieser kann von einem CET abgeleitet, oder für den Kontext der CLI spezifisch erstellt worden sein.

Da nun die Begrifflichkeiten festgelegt wurden, wird im nächsten Kapitel die Anforderungsanalyse für pCT durchgeführt.

3

Anforderungsanalyse

In der Anforderungsanalyse wird die Grundlage der Gestaltung gelegt und der Umfang von pCT definiert indem verschiedene Analysen durchgeführt werden. Hierbei werden zunächst vergleichbare Anwendungen analysiert, eine Benutzerprofilanalyse durchgeführt und anhand von *Anwendererzählungen*[1] die Aufgaben der Benutzer festgehalten. Über die Umgebungsanalyse werden schließlich die Rahmenbedingungen analysiert. Als grafische Übersicht und im Zusammenhang mit de andern Kapiteln ist dieses Kapitel in Abbildung 3.1 dargestellt.

[1] engl.: User Stories

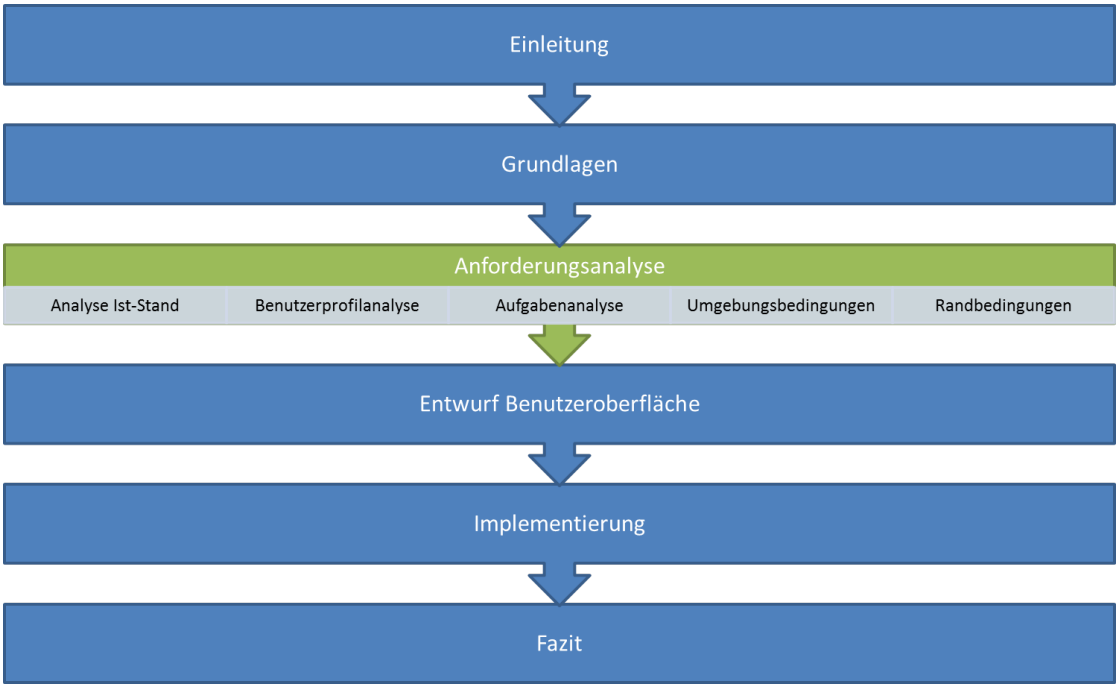


Abbildung 3.1: Aufbau Kapitel 3 — Anforderungsanalyse

3.1 Analyse Ist-Stand

Bei der Analyse des Ist-Standes werden vorhandene Altsysteme und bestehende Abläufe analysiert, die durch das neue System ersetzt oder optimiert werden sollen. Des Weiteren werden Konkurrenzsysteme und vergleichbare Anwendungen analysiert, die als Grundlage für Verbesserungsmöglichkeiten dienen und es ermöglichen gute Konzepte für eigene Entwicklungen abzuleiten.
Die Suche und Analyse verbreiteter Anwendungen, die in Richtung Checklisten gehen, hat jedoch keine direkt mit pCT vergleichbare Anwendung ergeben. Dennoch können die verwendeten Konzepte analysiert und möglicherweise verbessert und verwendet werden. Zu diesem Zweck wurden die folgenden Anwendungen zur Aufgabenverwaltung auf einem Tablet analysiert:

Anwendungen mit über 1.000.000 Installationen:

- *Astrids Tasks & To-do Lis*t [Tod13b]

- *Remember The Milk* [Rem13]

Anwendungen mit über 100.000 Installationen:

- *Todoist* [Tod13a]
- *ProDo* [Pro12]
- *Tasks Free* [Bit13]
- *Tasks N Todos* [Han13]
- *noodles free - To Do List* [mak13]

Anwendung mit über 50.000 Installationen:

- *Tasks+* [Ing13]

Anwendungen mit über 1000 Installationen:

- *Nozbe HD* [Noz13]

Da sich die Anwendungen nach einer ersten Analyse in der Mehrheit der Punkte wenig unterscheiden, werden diese im Folgenden konsolidiert beschrieben. Ausnahmen bei einzelnen Anwendungen werden gesondert hervorgehoben. Für eine bessere Lesbarkeit wurden die verglichenen Funktionalitäten in Bereiche eingeteilt.

3.1.1 Anmeldung und Registrierung

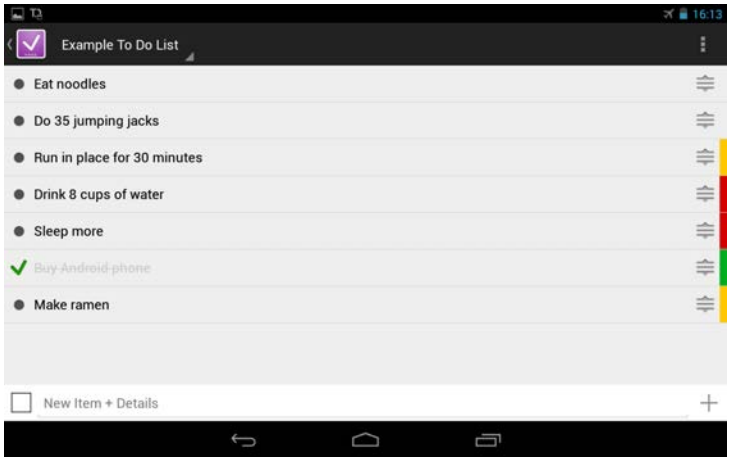
Allgemein: Nach dem Start der Anwendung ist, bis auf wenige Ausnahmen, eine Registrierung mit einer E-Mail Adresse notwendig. Zudem muss man unterscheiden, ob die Anwendung Synchronisationsdienste anbietet oder nicht, und ob diese optional sind. Bis auf eine Ausnahme werden von allen analysierten Anwendungen Dienste dieser Art bereitgestellt. Zur Synchronisation wird entweder auf Google Tasks gesetzt oder ein eigener Dienst der Anwendungsentwickler angeboten.
Die Anmeldung erfolgt mit der registrierten E-Mail Adresse und dem dabei eingegebenen Passwort. Jede der Anwendungen merkt sich die Anmeldedaten nach der ersten Eingabe, wodurch keine neue Anmeldung bei folgenden Starts nötig ist.

Ausnahmen: ProDo, Tasks+, Astrids Tasks und Tasks N Todos lassen sich auch ohne Registrierung verwenden. Die Synchronisation der Aufgaben mit Google Tasks ist hier optional zuschaltbar.
Noodles benötigt keine Anmeldung und bietet auch keine Synchronisation.

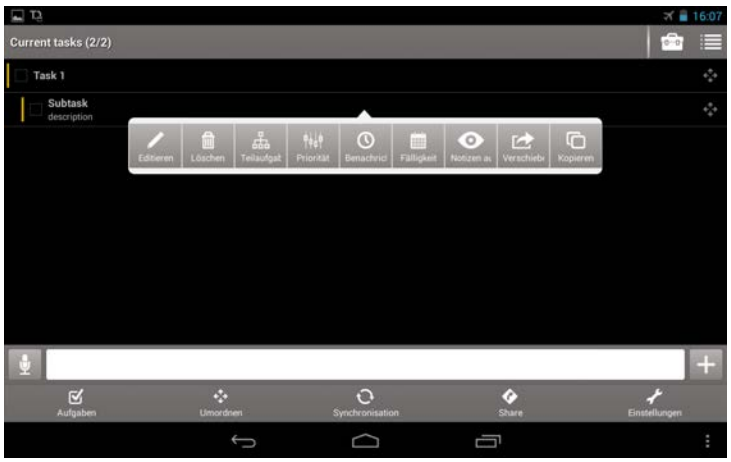
3.1.2 Hauptansicht

Allgemein: Die Hauptansicht ist in der Regel die Ansicht der CEs einer CL. Diese unterscheidet sich im Aufbau von Anwendung zu Anwendung maßgeblich. Die CEs werden nach unterschiedlichen Sortierverfahren untereinander angeordnet. Am unteren Bildschirmrand befindet sich in der Regel ein Eingabefeld zum Hinzufügen neuer CEs. In Abbildung 3.2 sind entsprechende Hauptansichten der Anwendungen Noodles (A), Prodo (B) und Tasks+ (C) dargestellt.

Ausnahmen: Nozbe HD (siehe Abbildung 3.3 A) bietet eine dynamische Ansicht, die zwischen zwei und drei Spalten wechselt. Die erste Spalte dient der Navigation innerhalb der Anwendung. CE werden der zweiten Spalte angezeigt. Die dritte Spalte wird vom Kontext abhängig angezeigt und enthält zum Beispiel die Details des ausgewählten CE. CL, in der Anwendung Projekte genannt, erhalten eine weitere Spalte zu deren Übersicht zwischen der Navigation und den CE. Wenn die Detailansicht eines der CL untergeordneten CE aufgerufen wird, blendet sich die CL-Spalte aus.
Die Hauptansicht von Astrids Tasks (siehe Abbildung 3.3 B) ist dreispaltig gestaltet. In der ersten Spalte befinden sich die CLs, in der zweiten die CEs und in der dritten können CEs bearbeitet werden. Das Hinzufügen eines CE oder einer CL geschieht jeweils am unteren Bildschirmrand in der jeweiligen Spalte.
Auch die Hauptansicht von Remember the Milk (siehe Abbildung 3.3 C) ist in einer dynamischen Spaltenaufteilung mit vier Spalten gegliedert. Die erste dient, wie bei Nozbe HD, der Navigation innerhalb der Anwendung. So kann zu den CLs, oder direkt zu dem aktuellen oder kommenden Tag gewechselt werden. Außerdem befindet sich hier eine Schaltfläche um einen neuen CE anzulegen. In der zweiten wird die aktuelle Woche oder vorhandene CLs angezeigt, sowie für jeden Tag bzw. jede CL die Anzahl



A



B

C

Abbildung 3.2: Die Hauptansicht der Anwendungen Noodles (A), Prodo (B) und Tasks+ (C)

der jeweils enthaltenen CE. Die dritte Spalte zeigt je nach Auswahl in der ersten Spalte, die CEs des aktuellen Tages an. Hier können über eine Schaltfläche wiederum CEs für den angezeigten Tag angelegt werden. Die Auswahl eines CE öffnet eine vierte Spalte mit Details und Bearbeitungsmöglichkeiten.

Tasks Free (siehe Abbildung 3.4 A) hat ein zweispaltiges Layout, bei dem in der linken Spalte die CEs dargestellt werden und in der rechten die Möglichkeit des Bearbeitens geboten wird.

Tasks N Todos (siehe Abbildung 3.4 B) zeigt in seinem zweispaltigen Layout links immer die CLs an, während die rechte Spalte der Ansicht der CEs und Bearbeitungsfunktionen vorbehalten ist.

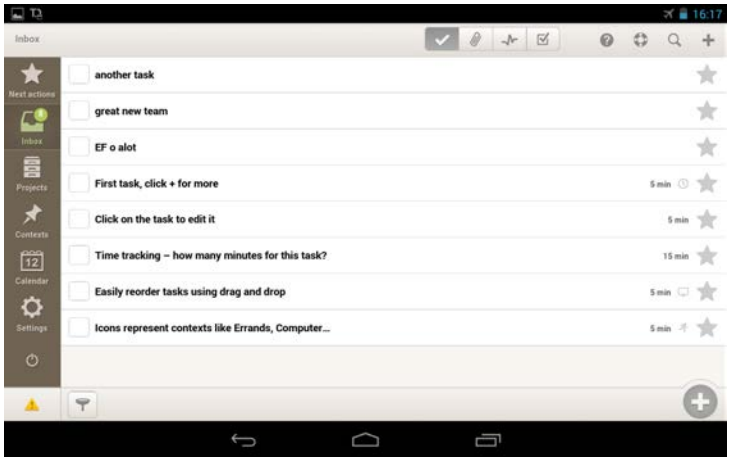
Das ebenfalls zweispaltige Layout von Todoist (siehe Abbildung 3.4 C) ist in eine Navigations- und eine Inhaltsspalte mit der Darstellung aller CEs aufgeteilt.

3.1.3 Navigation

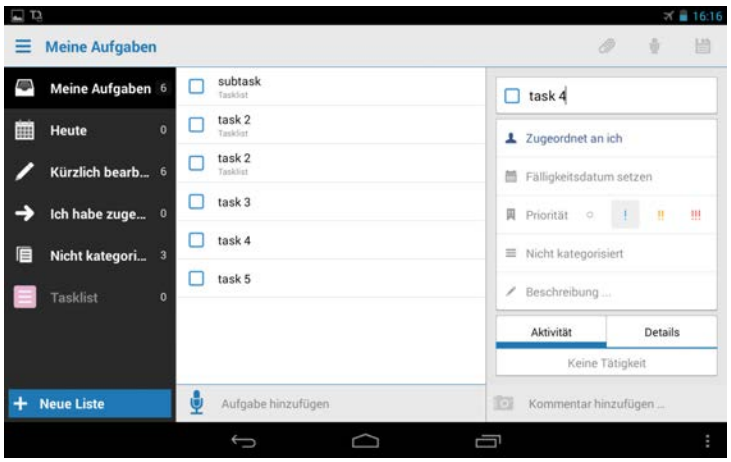
Allgemein: Es wird eine Menüzeile verwendet, um auf die verschiedenen Funktionalitäten der Anwendungen zuzugreifen. Der Wechsel zwischen der CL- und der CE-Ansicht findet über eine Schaltfläche in dieser statt (siehe Abbildung 3.5, Nummer 1). Um zwischen CL zu wechseln wird in der Mehrheit der Anwendungen eine horizontale Wischgeste eingesetzt, welche die Ansicht auf eine andere CL und deren CEs verschiebt (siehe Abbildung 3.5, Nummer 2).

Ausnahmen: Tasks+ und Noodles verwenden statt der Wischgesten eine Auswahlbox am oberen Bildschirmrand, um zwischen CL zu wechseln (siehe Abbildung 3.5, Nummer 1).

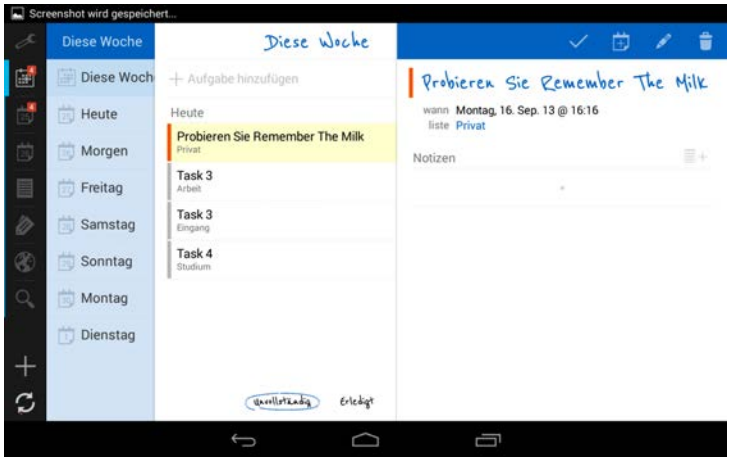
Bei Astrids Tasks, Remember the Milk, Todoist und Nozbe HD werden die CLs in der Navigationsspalte mittels Schaltflächen durchgeschaltet (siehe Abbildung 3.5, Nummer 3).



A

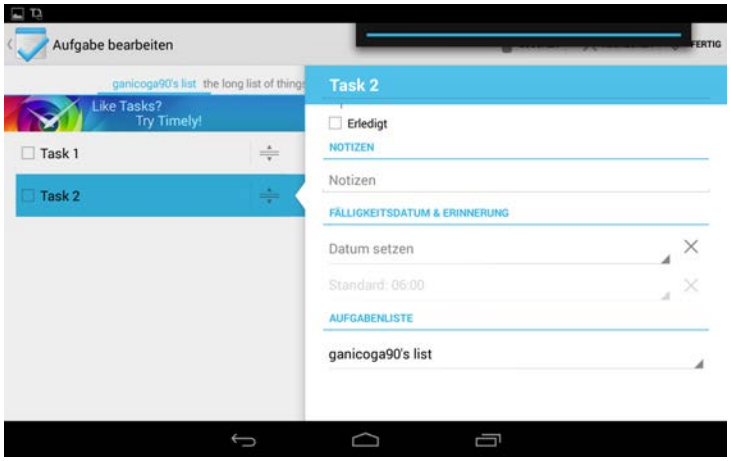


B

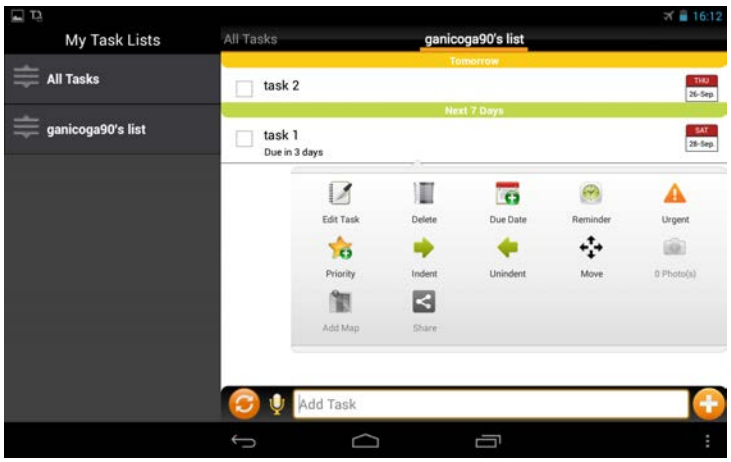


C

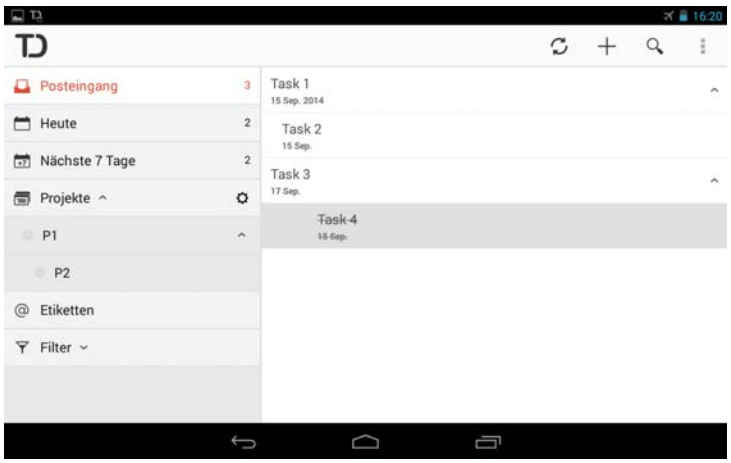
Abbildung 3.3: Die Hauptansicht der Anwendungen Nozbe HD (A), Astrids Tasks & Todo List (B) und Remember the Milk (C)



A



B



C

Abbildung 3.4: Die Hauptansicht der Anwendungen Tasks Free (A), Tasks N Todos (B) und Todoist (C)

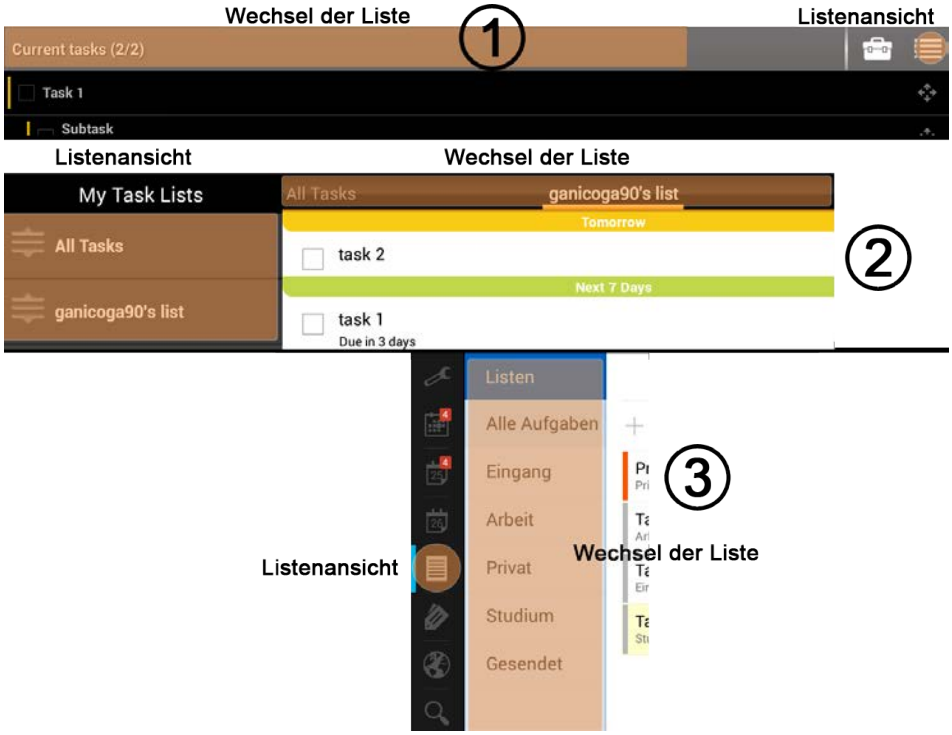


Abbildung 3.5: Ansichts- und Listenwechsel

3.1.4 Organisation

Allgemein: Alle Anwendungen bieten das Konzept der hierarchischen Strukturierung, wobei CLs mehrere CEs enthalten können. Die CEs können auch anderen CEs untergeordnet werden. So entstehen Hierarchien innerhalb der CL (siehe Abbildung 3.6, Nummern 1, 3). Einen übergeordneten OR, wie er in Kapitel 2.4 definiert wurde, gibt es bei keiner der Anwendungen. Die Benutzung der CLs ist außerdem auf einen einzigen Benutzer eingeschränkt.

Ausnahmen: Tasks Free, Noodles, Remember the Milk und Nozbe HD bieten nicht die Möglichkeit CEs oder CLs einander unterzuordnen.
Nozbe HD nennt seine CLs Projekte. Diese dienen alleine der Zusammenfassung von CEs in Kategorien und enthalten keine weitere Information über den Kontext (siehe Abbildung 3.6, Nummer 2). Hier ist es auch möglich einen CE zu einer CL zu transformieren. Es können außerdem zu diesen CLs auch weitere Kontakte eingeladen werden. Nozbe

HD bietet außerdem noch die Möglichkeit CEs mit Kontextinformation zur späteren kategorischen Filterung zu versehen.
Auch Todoist bietet die Möglichkeit sogenannte Projekte und Unterprojekte zu erstellen (siehe Abbildung 3.6, Nummer 1). Diese stellen jedoch einfach eine weitere Möglichkeit der Kategorisierung dar. Es werden keine zusätzlichen Informationen zum Kontext, oder die Möglichkeit der Kooperation gegeben. In einem solchen Projekt angelegte CEs werden nicht in der normalen Hauptansicht angezeigt und sind auch nicht zwischen diesen Ansichten verschiebbar.

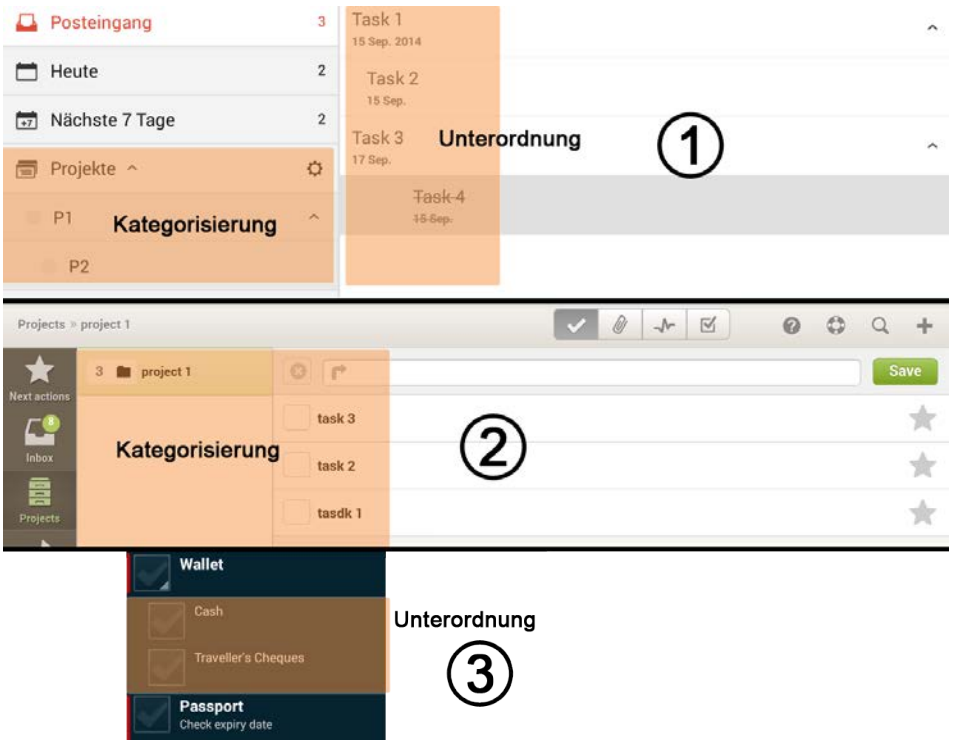


Abbildung 3.6: Organisation von CLs und CEs

3.1.5 Interaktion

Bei der Analyse der Interaktionsmöglichkeiten in den Anwendungen wurden drei Funktionalitäten herausgegriffen und verglichen: Die hierarchische Untergliederung, das Abhaken und die Sortierung von CEs.

Hierarchisieren von CEs

Allgemein: Das Unterordnen von CEs geschieht über eine horizontale Wischgeste nach rechts. Dabei wird der CE dem darüberliegenden untergeordnet. Die Geste nach links schiebt den CE auf die Ebene des Vatereintrags zurück.
Das Verschieben von CEs in andere CL geschieht über einen Dialog, welcher über eine Schaltfläche in einem Menü aufgerufen wird (siehe Abbildung 3.7, Nummern 1, 2). Das Menü wird bei Anwendungen, die keine separate Bearbeitungsansicht besitzen, ebenfalls dazu verwendet, um den Bearbeitungsdialog aufzurufen.

Ausnahmen: Bei Todoist muss die Hierarchisierung in der Bearbeitungsansicht durchgeführt werden.
Tasks+ verwendet ein Inline-Menü mit Schaltflächen zum An- und Unterordnen.
Bei Tasks N Todos findet sich die Funktion in einem Popup-Menü, welches bei Berührung des CE erscheint (Abbildung 3.7, Nummer 2).

Abhaken von CEs

CEs werden über eine Checkbox oder Schaltfläche per Berührung dieser abgehakt (siehe Abbildung 3.7, Nummer 4).
Bei Noodles und alternativ bei Todoist muss zum Abhaken eine horizontale Wischgeste über den CE durchgeführt werden, welche das Durchstreichen repräsentieren soll.

Sortieren von CEs

Eine manuelle Sortierung findet bei den Anwendungen per Drag & Drop des CE statt. Diese muss bei Tasks N Todos, ProDo zunächst über eine Option in den Einstellungen aktiviert werden. Remember the Milk bietet als einzige der betrachteten Anwendungen nicht die Möglichkeit CE zu sortieren.

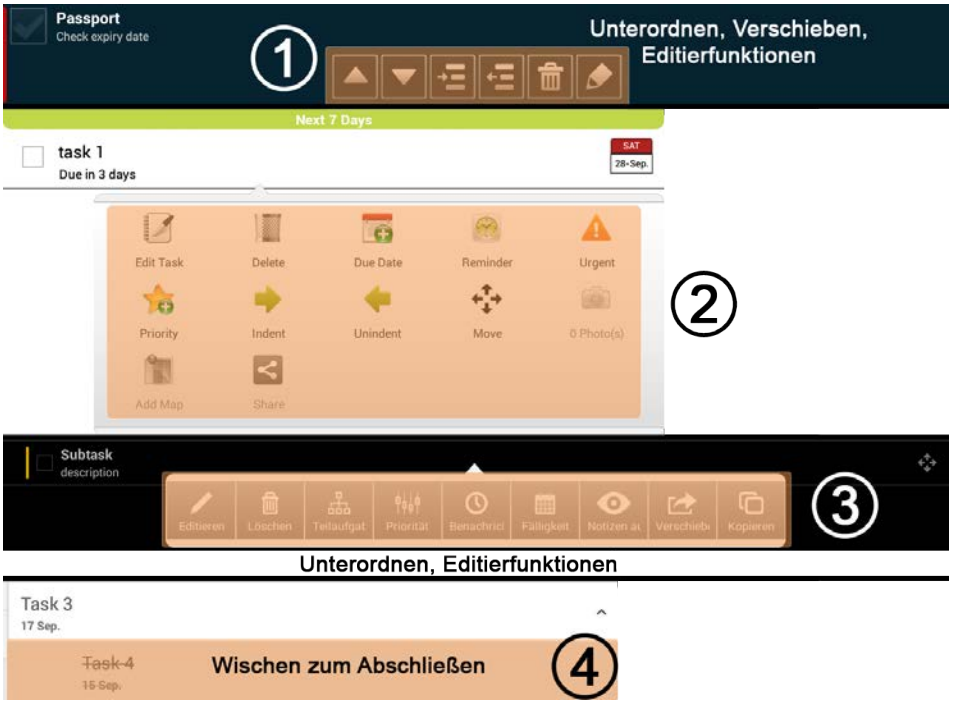


Abbildung 3.7: Interaktion mit CL und CE

3.1.6 Stil

Allgemein: Die Anwendungen halten sich bei der Verwendung von Farben generell zurück. Es überwiegen Weiß und Grautöne als Hintergrundfarbe mit schwarzer Textfarbe.

Ausnahmen: Prodo und Tasks+ setzen auf ein inverses Farbschema mit dunklem Hinter- und weißem Vordergrund. Highlights, wie wichtige Schaltflächen oder Titel, werden vom Großteil der Anwendungen mit einem hellen Blau gesetzt.
Einzig Tasks N Todos verwendet hier ein Orange und Nozbe HD ein helles Grün.

3.2 Kontextanalyse der Anwender

Um den Kontext der späteren Verwendung von pCT zu analysieren, müssen die Anwender von diesem ebenfalls betrachtet werden.
Bei der Benutzerprofilanalyse gilt es demnach herauszubekommen, welche Eigenschaf-

ten die späteren Nutzer des Systems besitzen und welche Auswirkungen diese auf die Bedienung und Gestaltung der Anwendung haben [Nie93].
Im Falle von pCT wird angenommen, dass es sich bei den Anwendern um Personen mittleren Alters handelt, die sich ihrem Fachgebiet auskennen und wissen wie sie ihre Aufgaben zu erfüllen haben. Sie wollen durch die Anwendung, in den ihnen zugeordneten wissensintensiven Prozessen, unterstützt werden und sollen deshalb keine zusätzliche mentale Belastung von dieser erfahren. Demnach sollte keine Einarbeitung erforderlich sein, sondern eine intuitive und schnelle Bedienung erreicht werden.
Ein weiterer wichtiger Faktor ist die Erfahrung der Anwender mit dem Eingabegerät. Bei Tablets handelt es sich um dem Smartphone ähnliche, größere Toucheingabegeräte. Über 40% der über 14-jährigen besitzen ein Smartphone [PS13] und schon mehr als ein Drittel der Unternehmen setzt Tablets ein [TP13]. Darauf aufbauend ist davon auszugehen, dass die meisten Anwender bereits Erfahrungen mit einem solchen Gerät haben.
Der dritte wichtige Faktor stellt die Häufigkeit der späteren Verwendung von pCT dar. Hierbei ist davon auszugehen, dass die Anwender die Anwendung mehrmals am Tag einsetzen und es sich daher eher um eine häufige Nutzung handelt.
Bei den Aufgaben der Anwender zeichnen sich vier grundlegende Benutzerrollen ab, welche in ihrer Abhängigkeit in Abbildung 3.8 dargestellt sind und im Folgenden beschrieben werden.

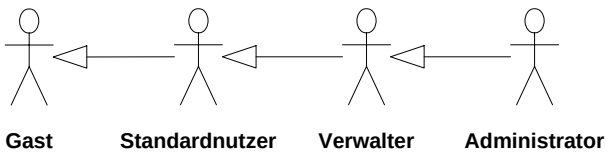


Abbildung 3.8: Vererbungshierarchie der Benutzerrollen

3.2.1 Gast

Eine Person, die keinen Zugriff auf die Inhalte der Anwendung hat, ist ein Gast. Sie hat nur die Möglichkeit sich zu registrieren und anzumelden um dann eine Rolle mit mehr Rechten zu erlangen.

3.2.2 Standardnutzer

Die einfachste Anwenderrolle ist der Standardnutzer. Er arbeitet regelmäßig mit der Anwendung und seine Aufgabe besteht darin innerhalb einer ORI ihm zugeteilte CLIs mit ihren CEIs abzuarbeiten. Er kann dabei auch in mehreren ORIs gleichzeitig tätig sein.

3.2.3 Verwalter

Der Verwalter ist einer oder mehreren ORIs zugeteilt und kann in diesen zusätzlich zu den Aufgaben des Standardnutzers, die ORIs mit ihren CLIs und CEIs verwalten, sowie CLTs und CETs instanziieren.

3.2.4 Administrator

Der Administrator übernimmt zusätzlich zu den Aufgaben des Verwalters die Verwaltung der Benutzer, sowie den generischen ORTs, CLTs und CETs. Diese Aufgaben werden aktuell jedoch nicht im mobilen Frontend unterstützt, sondern sind über die webbasierte Administrationsoberfläche zugänglich [Thi13].

3.3 Kontextanalyse der Aufgaben

Zur Kontextanalyse gehört vor allem auch die Analyse der Aufgaben der einzelnen Benutzerrollen. Diese sind entscheidend für die Anwendung, da sie den nötigen Funktionsumfang widerspiegeln. Die Aufgaben werden im Folgenden kategorisch abgehandelt. Bei den Kategorien handelt es sich um..

- ..die Benutzerverwaltung (BV, siehe Kapitel 3.3.1).
- ..die Verwaltung von organisatorischen Rahmen (ORV, siehe Kapitel 3.3.2).
- ..die Verwaltung von Checklisten (CLV, siehe Kapitel 3.3.3).
- ..die Verwaltung von Checklisteneinträgen (CEV, siehe Kapitel 3.3.4).

- ..übergreifende Verwaltungsmaßnahmen (UV, siehe Kapitel 3.3.5).

Die einzelnen Aufgaben und Abläufe werden dazu mit Hilfe einer Auswahl von Kontextfragen nach [Off13] im Detail analysiert:

- Wann wird die Aufgabe durchgeführt? (Auslöser, Vorbedingung)
- Von wem wird die Aufgabe durchgeführt?
- Warum wird die Aufgabe durchgeführt (Handlungsziel, Nachbedingung)
- Wie oft wird die Aufgabe durchgeführt?
- Was ist bei der Durchführung im einzelnen zu tun?
- Welche Ausnahme-/Sonderfälle zur Normalvorgehensweise gibt es?

Die Schritte der Durchführung sind dabei an Anwendererzählungen angelehnt. Eine Anwendererzählung beschreibt, wie ein Nutzer vorgeht um die entsprechende Aufgabe zu erfüllen und welches Feedback die Benutzeroberfläche ihm zurückgibt. Jeder Ablauf wird außerdem priorisiert und mit einer Flussdiagrammnotation visualisiert. Die Beschreibung der Notation ist in Tabelle 3.1 dargestellt.

3.3.1 Benutzerverwaltung

Die folgenden Abläufe aus dem Kontext der Benutzerverwaltung beschreiben die Möglichkeiten des Benutzers auf die Anwendung zuzugreifen und seine Daten zu verwalten. Sie sind in Abbildung 3.9 in Form einer Übersicht dargestellt.

BV-1 Registrierung

Priorität: hoch

Beschreibung: Ein Gast hat noch kein Benutzerkonto und möchte sich registrieren, um die Anwendung künftig als Standardnutzer nutzen zu können.

Häufigkeit: Er muss sich einmalig mit seinen Daten im System registrieren.

Durchführung: Er den Registrierungsdialog auf und gibt dort seinen Vor- und Nachnamen, E-Mail Adresse, den Namen seiner Organisation, die URL zu deren Website und sein gewünschtes Passwort ein. Zur Bestätigung gibt er ein zweites mal das

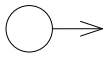
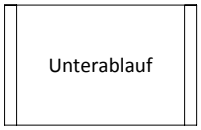
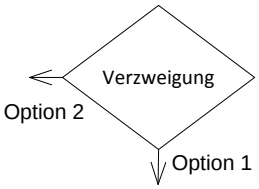
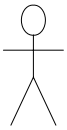
Symbol	Bezeichnung	Bedeutung
	Startpunkt	Stellt den Start des Aktivitätsablaufs dar und hat keinen Eingang.
	Endpunkt	Stellt das Ende des Aktivitätsablaufs dar und hat keinen Ausgang.
	Aktivität	Stellt eine zusammengehörige Aktivitätsfolge dar.
	Unterablauf	Stellt einen Aktivitätsablauf dar, welcher selbst in einer verfeinerten Darstellung vorliegt.
	Verzweigung	Stellt die Aufteilung des Aktivitätsablaufs aufgrund einer Entscheidung dar.
	Benutzerinteraktion	Stellt die Interaktion mit einem Benutzer dar. Dieser kann entweder eine Eingabe tätigen oder eine Ausgabe erhalten.
	Interaktion	Stellt die Ein- und Ausgabe von Informationen dar.
	Aktivitätsfluss	Stellt den Ablauf der Aktivitäten dar.

Tabelle 3.1: Beschreibung der Flussdiagrammnotation

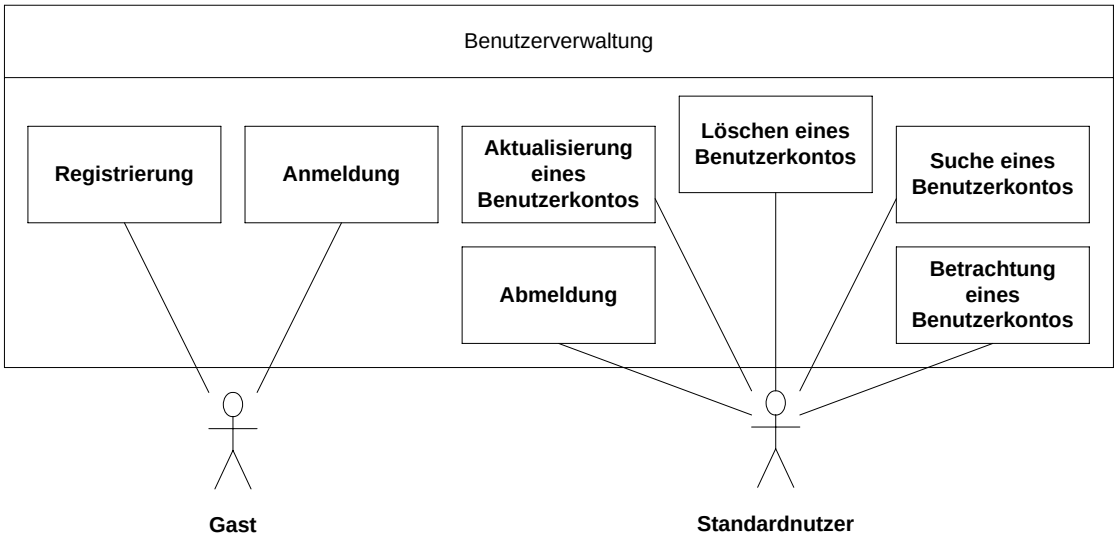


Abbildung 3.9: Anwendungsfälle des Gastes zum Zugriff auf die Anwendung und des Standardnutzers zur Verwaltung seiner Daten

Passwort ein. E-Mail Adresse und Passwort werden während der Eingabe validiert und bei fehlschlagender Validierung markiert und mit einem Hinweis versehen. Nach dem erfolgreichen Abschicken des Formulars erhält der Gast eine Bestätigung und die Aufforderung angezeigt, sein E-Mail Postfach zur Verifikation der E-Mail Adresse und der Aktivierung des Kontos zu prüfen. Nachdem der Gast den Aktivierungslink in der E-Mail angeklickt hat, erhält dieser eine Bestätigung, dass er die Anwendung nun mit seinen Benutzerdaten verwenden kann (siehe Abbildung 3.10).

Ausnahmen: Im Falle eines Fehlers oder noch nicht ausgefüllten Formularfelder wird dem Gast eine Mitteilung mit den Details angezeigt. Ursachen für einen Fehler sind eine schon vorhandene E-Mail Adresse, eine inkorrekte Eingabe der E-Mail Adresse, ein zu unsicheres Passwort oder Sonderzeichen im Namen.

BV-2 Anmeldung

Priorität: sehr hoch

Beschreibung: Ein Gast mit vorhandenem Benutzerkonto möchte sich in der Anwendung anmelden, um seinen Arbeiten nachzugehen.

Häufigkeit: Er muss dies einmalig und nach einer manuellen Abmeldung (siehe BV-6) tun.

Durchführung: Um sich anzumelden, gibt der Gast seine E-Mail Adresse und sein Passwort in die Anmeldemaske ein. Bei erstmaliger erfolgreicher Anmeldung wird ihm die Hauptseite der Anwendung angezeigt. Die Benutzerdaten werden dabei in der Anwendung gespeichert, Wenn der Benutzer zuvor schon einmal angemeldet war, oder er die Anwendung mit gespeicherten Daten startet, wird ihm seine letzte Ansicht wiederhergestellt (siehe Abbildung 3.11).

Ausnahmen: Bei falscher Eingabe der Benutzerdaten wird ein entsprechender Hinweis angezeigt und zur Korrektur der Eingabe aufgefordert. Ein Benutzer, der seine E-Mail Adresse nicht verifiziert hat, kann sich nicht anmelden.

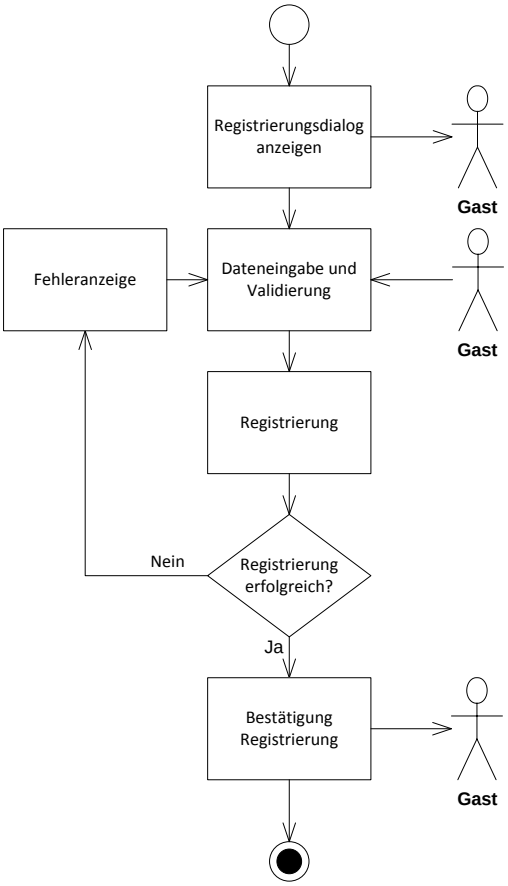


Abbildung 3.10: Registrierung (BV-1)

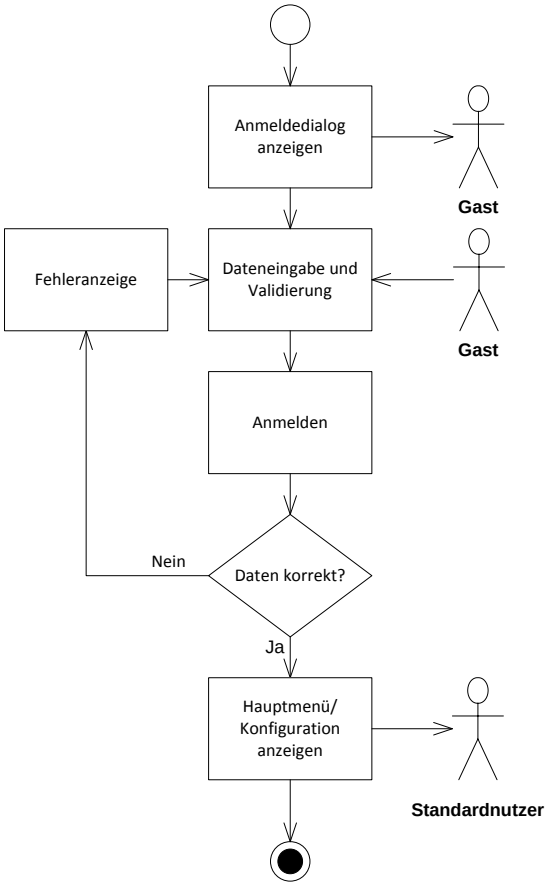


Abbildung 3.11: Anmeldung (BV-2)

BV-3 Suche eines Benutzerkontos

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte das Konto eines anderen Benutzers suchen, um sich die Daten und Rollen dieses Benutzers anzusehen. Alternativ sucht ein Verwalter nach einem Benutzer, um diesen einer ORI oder einer CLI zuzuordnen.

Häufigkeit: Das Zuteilen von Benutzern zu ORI und CLI stellt eine Hauptaufgabe des Verwalters dar. Eine vorausgehende Suche wird deshalb, gerade bei neu angelegten ORIs und CLIs, häufig durchgeführt.

Durchführung: Über ein Suchformular sucht er den entsprechenden Benutzer mit Hilfe der Attribute Vorname, Nachname, E-Mail Adresse und Organisation. Alle auf die Eingabe passenden Ergebnisse werden ihm dabei in einer Ergebnisliste angezeigt (siehe Abbildung 3.12).

Ausnahmen: Sind keine passenden Ergebnisse vorhanden, oder wird eine Suche mit unzulässigen Sonderzeichen getätigt, wird dem Benutzer eine Hinweismeldung dazu angezeigt.

BV-4 Betrachtung eines Benutzerkontos

Priorität: mittel

Beschreibung: Ein Standardnutzer möchte seine oder die Daten eines anderen Benutzers ansehen. Er macht dies, um anschließend beispielsweise seine Daten zu bearbeiten (siehe BV-5), um die Rolle eines Benutzer innerhalb einer ORI festzustellen oder als Verwalter, um den Benutzer einer ORI zuzuweisen (siehe UV-1).

Häufigkeit: Die Aufgabe ist bei jeder Interaktion mit Benutzern relevant und wird entsprechend häufig durchgeführt.

Durchführung: Der Benutzer wählt dazu sein Konto aus oder sucht das Konto eines anderen Benutzers (siehe BV-3). Anschließend erhält er eine Ansicht mit den Daten Vorname, Nachname, E-Mail Adresse, Organisation und der Website der Organisation des Benutzers angezeigt. Er sieht außerdem, in welcher Rolle dieser ORIs und CLIs zugeordnet ist (siehe Abbildung 3.13).

Ausnahmen: Der Spezialfall, sein eigenes Benutzerkonto anzusehen, ist eine Erweiterung des Standardfalls. Hier bieten sich dem Benutzer weitere Möglichkeiten, wie die Aktualisierung (siehe BV-5) oder das Löschen (siehe BV-7) seines Kontos.

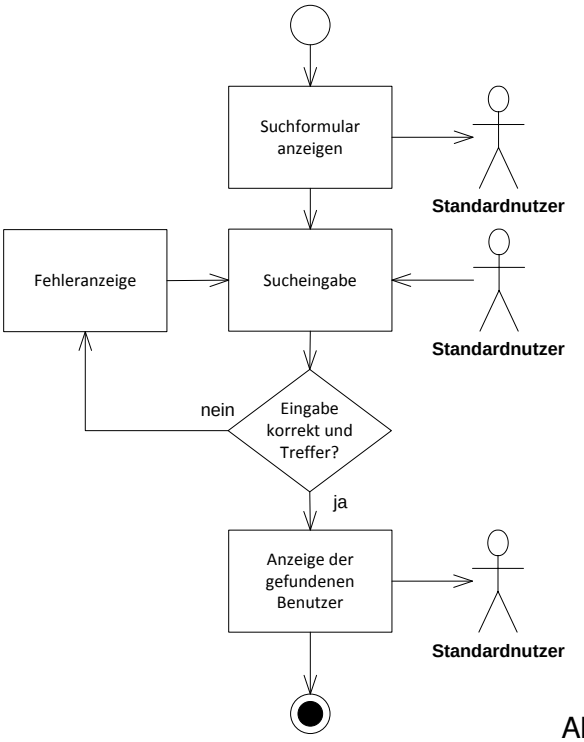


Abbildung 3.12: Suche eines Benutzerkontos (BV-3)

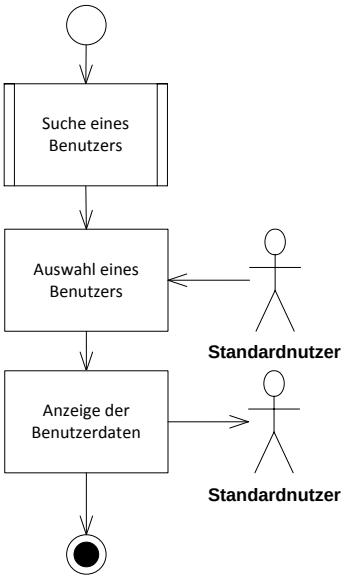


Abbildung 3.13: Betrachtung eines Benutzerkontos (BV-4)

BV-5 Aktualisierung eines Benutzerkontos

Priorität: mittel

Beschreibung: Ein Standardnutzer möchte seine Benutzerdaten ändern, da sich beispielsweise seine E-Mail Adresse geändert hat oder er möchte sein Passwort ändern. Nach der Betrachtung seines Benutzerkontos (siehe BV-4) kann ein Standardnutzer über eine Option die Felder für seine Daten editierbar machen.

Häufigkeit: Die Aktualisierung des Benutzerkontos kommt selten vor, da sich die Daten in der Regel gleich bleiben und sich nur in wenigen Fällen ändern.

Durchführung: Nachdem der Standardnutzer seine Daten editierbar gemacht hat, kann dieser seinen Vor- und Nachnamen, seine E-Mail Adresse, seine Organisation und deren Website URL verändern. Des Weiteren kann der er sein Passwort ändern, indem er einmal das alte und zweimal das neue Passwort eingibt (siehe Abbildung 3.14).

Ausnahmen: Die Eingabefelder werden validiert und bei Falscheingaben markiert und mit einem Hinweis versehen. Beim Übernehmen der Änderungen erhält der Benutzer eine Bestätigung angezeigt. Hat er seine E-Mail Adresse geändert, wird ihm eine E-Mail zur Verifikation zugestellt.

BV-6 Abmeldung

Priorität: mittel

Beschreibung: Ein Standardnutzer möchte sich von der Anwendung abmelden, weil sich beispielsweise ein anderer Benutzer auf dem selben Gerät anmelden möchte.

Häufigkeit: Die Aufgabe kommt selten vor, da jedes mobile Gerät in der Regel von einem einzelnen Nutzer verwendet wird.

Durchführung: Um sich abzumelden, wählt der Benutzer die entsprechende Option im Menü der Anwendung aus. Seine Daten werden aus dem Speicher der Anwendung entfernt und er landet anschließend in der Anmeldemaske (siehe Abbildung 3.15).

BV-7 Löschen eines Benutzerkontos

Priorität: niedrig

Beschreibung: Ein Standardnutzer möchte sein Konto löschen, da er beispielsweise keinen weiteren Bedarf an der Anwendung hat.

Häufigkeit: Dieser Fall kommt sehr selten vor.

Durchführung: Um sein Konto zu löschen wählt der Benutzer sein Konto aus (siehe BV-4) und wählt die dortige Option zum Löschen des Kontos. Er muss eine Warnung bestätigen, woraufhin er abgemeldet wird. Des Weiteren wird ihm eine E-Mail

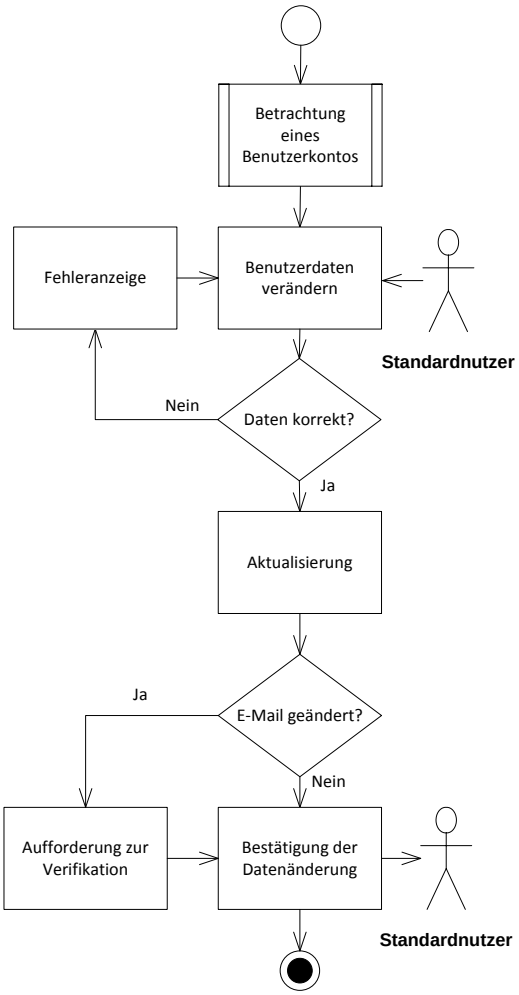


Abbildung 3.14: Aktualisierung eines Benutzerkontos (BV-5)

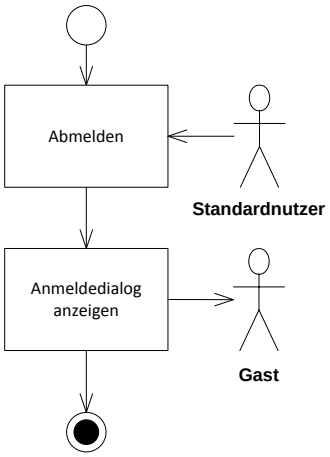


Abbildung 3.15: Abmeldung (BV-6)

zugeschickt, in welcher er das Löschen ein weiteres Mal bestätigen muss (siehe Abbildung 3.16).

Ausnahmen: Beim Löschen muss darauf geachtet werden, dass der Benutzer nicht als einziger die Rolle eines Verwalters in einer ORI innehält, da es sonst neben dem Administrator keinen Benutzer mit Verwaltungsrechten mehr gibt. Tritt dieser Fall auf, wird dem Benutzer eine Meldung dazu ausgegeben, worin er aufgefordert wird, die ORI an einen anderen Benutzer zu übergeben oder diese zuvor zu entfernen (siehe ORV-9).

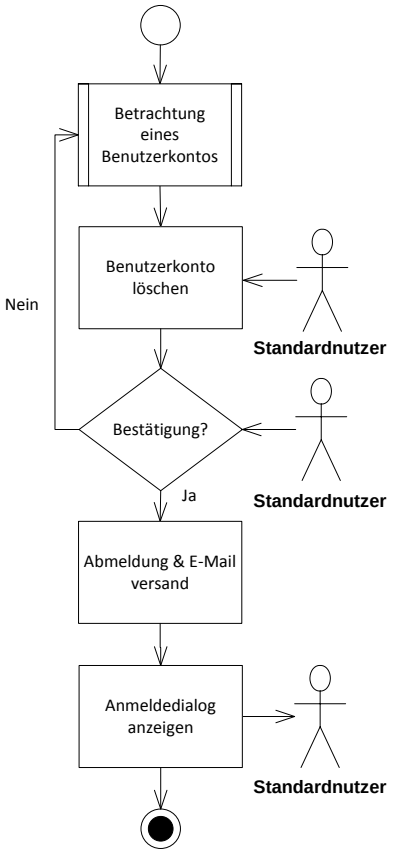


Abbildung 3.16: Löschen eines Benutzerkontos (BV-7)

3.3.2 Verwaltung von organisatorischen Rahmen

Die Aufgaben der Verwaltung von OR sind der Rolle des Verwalters zugeordnet. Aufgaben, die keine bestehenden ORI verändern, können jedoch auch von einem Standardnutzer durchgeführt werden. In Abbildung 3.17 sind diese Aufgaben in der Übersicht dargestellt. Weitere Aufgaben, welche das Erzeugen ORTs einschließt, werden nur vom Administrator durchgeführt.

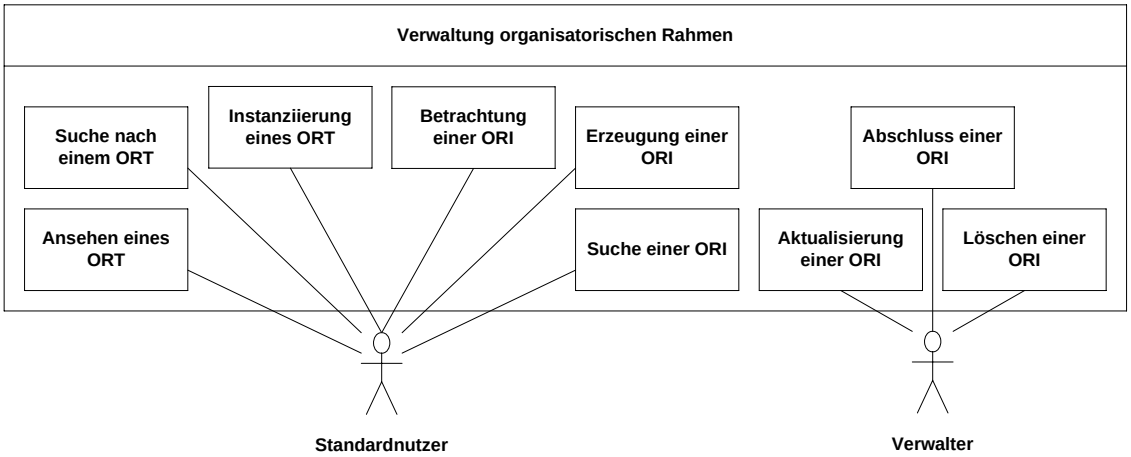


Abbildung 3.17: Anwendungsfälle des Benutzers und Verwalters zur Verwaltung der organisatorischen Rahmen

ORV-1 Suche eines ORT

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte einen ORT instanziieren (siehe ORV-3) oder die Attribute eines ORT ansehen (siehe ORV-2). Dazu muss er zuerst den entsprechenden ORT suchen.

Häufigkeit: Eine Suche findet jedes mal statt, wenn eine Handlung mit einem vordefinierten ORT geschehen soll.

Durchführung: Die Suche wird vom Benutzer durchgeführt, indem er in einer Eingabemaske nach den Eigenschaften Name, Beschreibung oder Ziel eines ORT sucht. Alle auf die Eingabe passenden Ergebnisse werden ihm dabei in einer Ergebnisliste angezeigt (siehe Abbildung 3.18).

Ausnahmen: Sind keine passenden Ergebnisse vorhanden, wird ihm eine Hinweismeldung dazu angezeigt.

ORV-2 Betrachtung eines ORT

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte sich die Details eines ORT anschauen, um diesen beispielsweise anschließend instanziieren zu können. Nach der Suche (siehe ORV-1) eines ORT wählt er diesen aus.

Häufigkeit: Dieser Vorgang wird bei jeder Instanziierung eines ORT durchgeführt.

Durchführung: Nach der Auswahl des ORT erhält der Standardnutzer die Ansicht der Eigenschaften des ORT mit Name, Beschreibung und Ziel angezeigt. Außerdem kann er die enthaltenen CLT ansehen (siehe Abbildung 3.19).

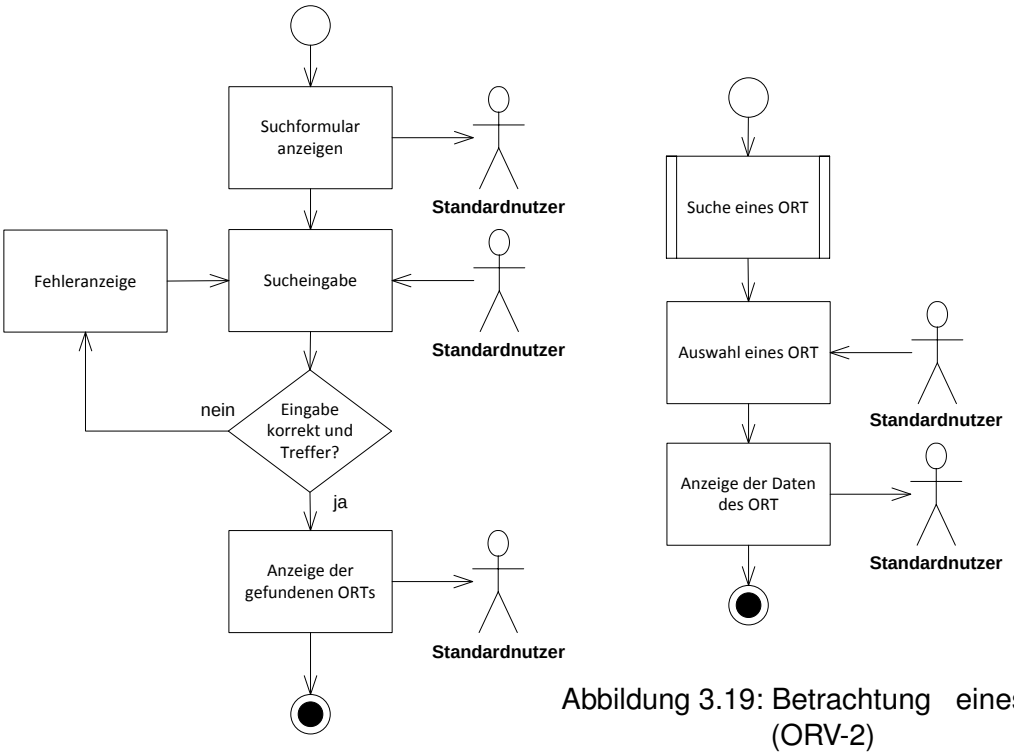


Abbildung 3.19: Betrachtung eines ORT (ORV-2)

Abbildung 3.18: Suche eines ORT (ORV-1)

ORV-3 Instanziierung eines ORT

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte einen bestehenden ORT instanziieren, da beispielsweise ein neues Projekt zur Softwareentwicklung gestartet werden soll. Nach der Betrachtung des ORT (siehe ORV-2), wählt er die Option zur Instanziierung aus.

Häufigkeit: Dieser Vorgang wird immer dann ausgeführt, wenn eine neue ORI eines vordefinierten ORT benötigt wird.

Durchführung: Nach der Betrachtung der ORT und der Auswahl zur Instanziierung wird dem Standardnutzer die Möglichkeit gegeben die ORI zu aktualisieren (siehe ORV-7). Der Standardnutzer nimmt nach der Instanziierung die Rolle des Verwalters für diese ORI ein (siehe Abbildung 3.20).

Ausnahmen: Bei inkorrekten Eingaben oder nicht ausgefüllten Feldern wird dem Benutzer ein Hinweis dazu angezeigt.

ORV-4 Erzeugung einer ORI

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte eine ORI anlegen, zu der es noch keine Vorlage im Sinne eines ORT gibt.

Häufigkeit: Dieser Vorgang wird immer durchgeführt, wenn kein passender ORT vorhanden ist.

Durchführung: Er gibt dazu in einer Eingabemaske Name, Beschreibung und Ziel der neuen ORI ein und erhält anschließend die Möglichkeit weitere Änderungen an dieser vorzunehmen (siehe ORV-7). Nach der Erzeugung der ORI, nimmt der Benutzer die Rolle des Verwalters für diese ORI ein (siehe Abbildung 3.21).

Ausnahmen: Bei inkorrekten Eingaben oder nicht ausgefüllten Feldern wird dem Benutzer ein Hinweis dazu angezeigt.

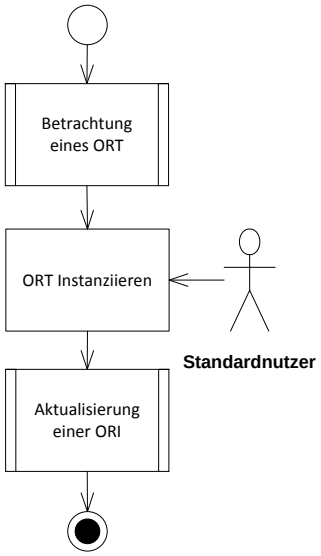


Abbildung 3.20: Instanziierung eines ORT (ORV-3)

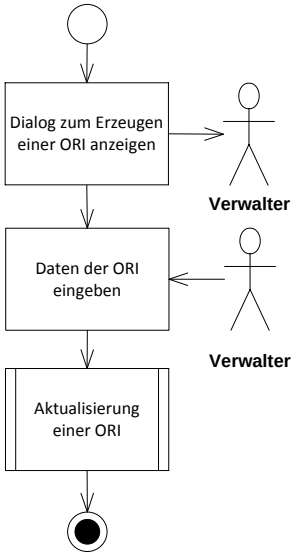


Abbildung 3.21: Erzeugung einer ORI (ORV-4)

ORV-5 Suche einer ORI

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte eine ihm zugeordnete ORI suchen, um an dieser zu arbeiten. Alternativ sucht ein Verwalter nach einer ORI um Änderungen an dieser vorzunehmen (siehe ORV-7).

Häufigkeit: Diese Aufgabe ist Teil vieler weiterer Aufgaben der Benutzer und wird entsprechend sehr häufig durchgeführt.

Durchführung: Um die Aufgabe durchzuführen sucht der Benutzer über eine Eingabemaske nach der entsprechenden ORI mithilfe deren Eigenschaften Name, Beschreibung und Ziel. Alle auf die Eingabe passenden Ergebnisse werden ihm dabei in einer Ergebnisliste angezeigt (siehe Abbildung 3.22).

Ausnahmen: Sind keine passenden Ergebnisse vorhanden, wird ihm eine Hinweismeldung dazu angezeigt.

ORV-6 Betrachtung einer ORI

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte sich die Eigenschaften einer ihm zugeordneten ORI anschauen, um beispielsweise das Enddatum zu prüfen. Das Betrachten einer ORI geschieht in der Regel nach der Suche (siehe ORV-5), woraufhin der Standardnutzer diese auswählt.

Häufigkeit: Die Aufgabe wird vom Standardnutzer sehr häufig durchgeführt, da sie Teil vieler weiterer Aufgaben ist.

Durchführung: Er erhält die Ansicht ihrer Eigenschaften mit Name, Beschreibung, Ziel, Startdatum und Enddatum oder verbleibende Tage angezeigt. Er sieht außerdem untergeordnete CLIs dieser ORI und zugeteilte Benutzer in ihren Rollen (siehe Abbildung 3.23).

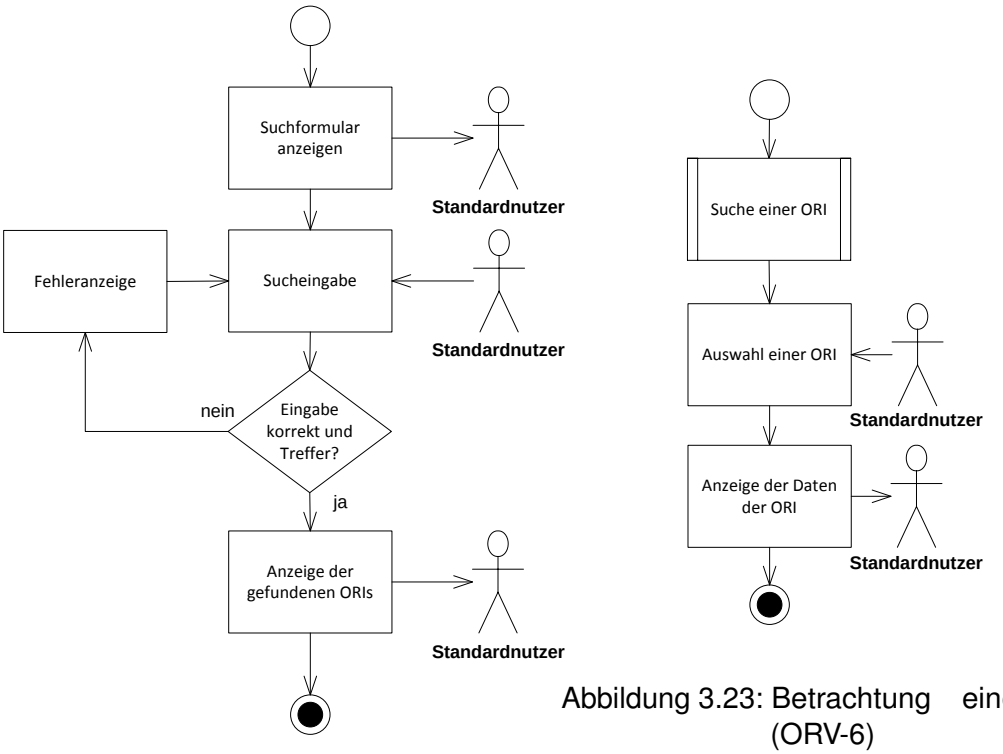


Abbildung 3.23: Betrachtung einer ORI (ORV-6)

Abbildung 3.22: Suche einer ORI (ORV-5)

ORV-7 Aktualisierung einer ORI

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte die Eigenschaften einer ORI anpassen, da sich zum Beispiel das Enddatum verschoben hat oder eine neue CLI hinzugefügt werden soll. Nach der Betrachtung einer ORI (siehe ORV-6) wählt er die Option zur Bearbeitung der ORI aus.

Häufigkeit: Das Aktualisieren einer ORI wird immer dann durchgeführt, wenn ein Datum geändert, neue CLI hinzugefügt oder ein Benutzer zum ORI zugeordnet werden soll.

Durchführung: Die Eigenschaften einer ORI werden mit Eingabefeldern editierbar dargestellt. Zu diesen zählen Name, Beschreibung, Ziel, Startdatum und die Anzahl der Tage oder das Enddatum, nach denen die ORI abgeschlossen sein muss. Nach dem Speichern der Änderungen ist die ORI aktualisiert (siehe Abbildung 3.24).
Der Verwalter hat außerdem die Möglichkeit die CLIs der ORI anzusehen, anzupassen (siehe CLV-7) und zu erweitern (siehe CLV-3 und CLV-4).

Ausnahmen: Bei inkorrekten Eingaben oder nicht ausgefüllten Feldern wird dem Verwalter ein Hinweis dazu angezeigt.

ORV-8 Abschluss einer ORI

Priorität: hoch

Beschreibung: Ein Verwalter möchte, nach Erfüllung aller Aufgaben der enthaltenen CLI, die zugehörige ORI als abgeschlossen markieren.

Häufigkeit: Dies wird immer durchgeführt, wenn eine ORI fertig bearbeitet wurde.

Durchführung: In der Betrachtung der ORI (siehe ORV-6) ändert der Verwalter dazu den Status der ORI auf abgeschlossen, woraufhin dieser archiviert wird (siehe Abbildung 3.25).

ORV-9 Löschen einer ORI

Priorität: sehr hoch

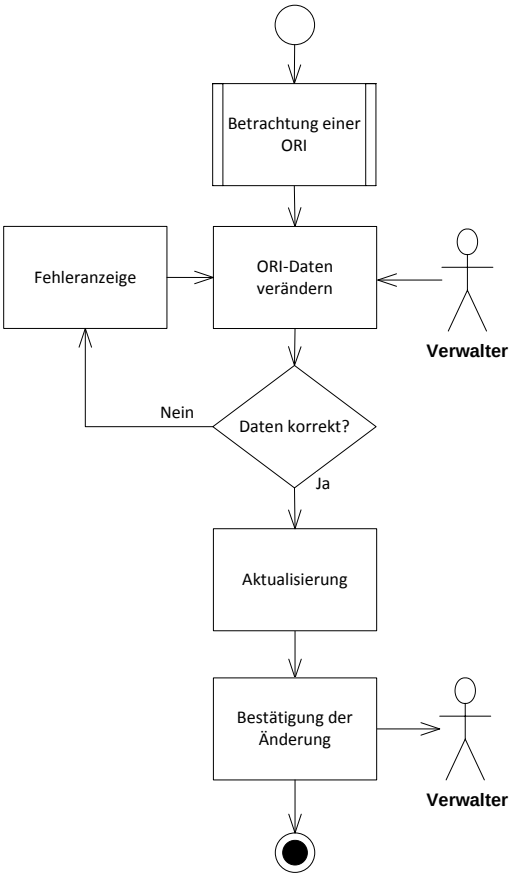


Abbildung 3.24: Aktualisierung einer ORI (ORV-7)

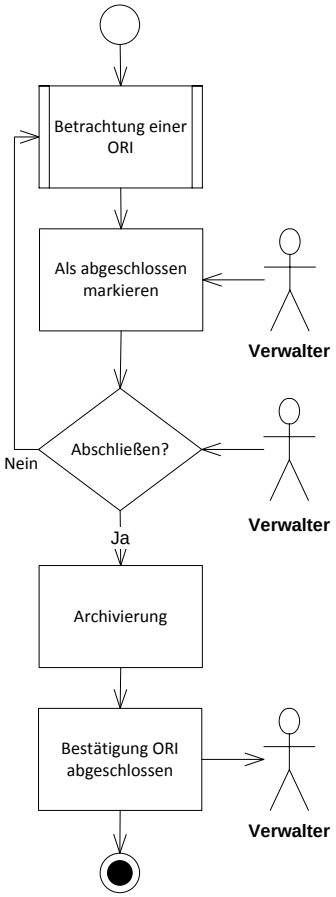


Abbildung 3.25: Abschluss einer ORI (ORV-8)

Beschreibung: Ein Verwalter möchte eine ihm zugeordnete ORI löschen, da sie nicht länger benötigt wird.

Häufigkeit: Diese Aufgabe kommt vor, wenn eine ORI nicht mehr genutzt werden soll.

Durchführung: Zur Durchführung der Aufgabe sucht der Verwalter die ORI (siehe ORV-5) aus. Anschließend wählt der die Option zum Löschen der ORI und bestätigt die darauf folgende Warnung (siehe Abbildung 3.26).

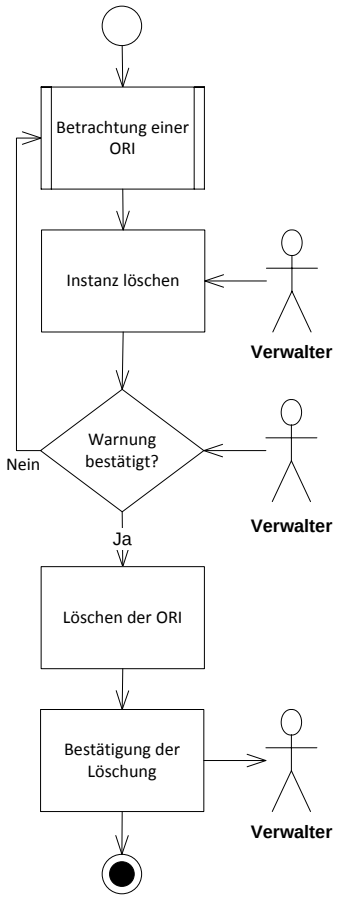


Abbildung 3.26: Löschen einer ORI (ORV-9)

3.3.3 Verwaltung von Checklisten

Die folgenden Aufgaben werden durch einen Verwalter durchgeführt und beschreiben Interaktionen mit den CLTs und CLIs. Weitere Aufgaben, welche das Erzeugen der CLT einschließt, werden nur vom Administrator unterstützt.

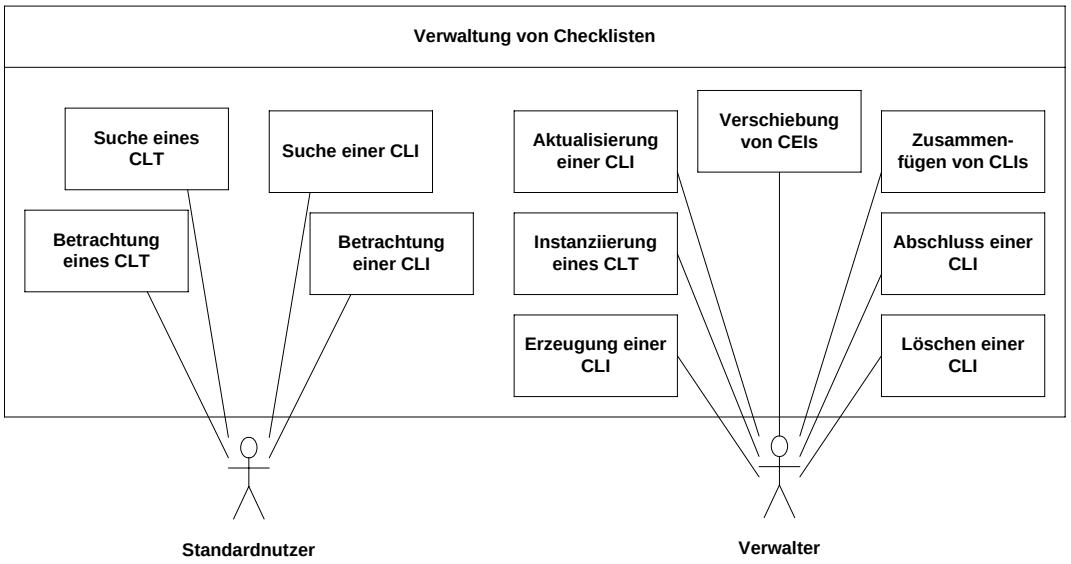


Abbildung 3.27: Anwendungsfälle des Standardnutzers und des Verwalters zur Verwaltung der Checklisten

CLV-1 Suche eines CLT

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte einen CLT instanziieren (siehe CLV-3) oder die Attribute des CLT betrachten (siehe CLV-2). Dazu muss er zuerst den entsprechenden CLT suchen.

Häufigkeit: Eine Suche findet jedes mal statt, wenn eine Handlung mit einem vordefinierten CLT stattfinden soll.

Durchführung: Sie wird von einem Verwalter durchgeführt, indem er in einer Eingabemaske nach den Eigenschaften Name und Beschreibung des CLT sucht. Alle auf die Eingabe passenden Ergebnisse werden ihm dabei in einer Ergebnisliste angezeigt (siehe Abbildung 3.28).

Ausnahmen: Sind keine passenden Ergebnisse vorhanden, wird ihm eine Hinweismeldung dazu angezeigt.

CLV-2 Betrachtung eines CLT

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte sich die Details eines CLT ansehen, um diesen beispielsweise anschließend zu instanziieren (siehe ORV-3). Nach der Suche des CLT (siehe ORV-1) oder in einer Betrachtung der Details eines ORT (siehe ORV-2) wählt er diesen CLT aus.

Häufigkeit: Dieser Vorgang wird bei jeder Instanziierung durchgeführt.

Durchführung: Er bekommt die Eigenschaften des CLT angezeigt. Dazu zählen Name, Beschreibung, Status und die enthaltenen Einträge (siehe Abbildung 3.29).

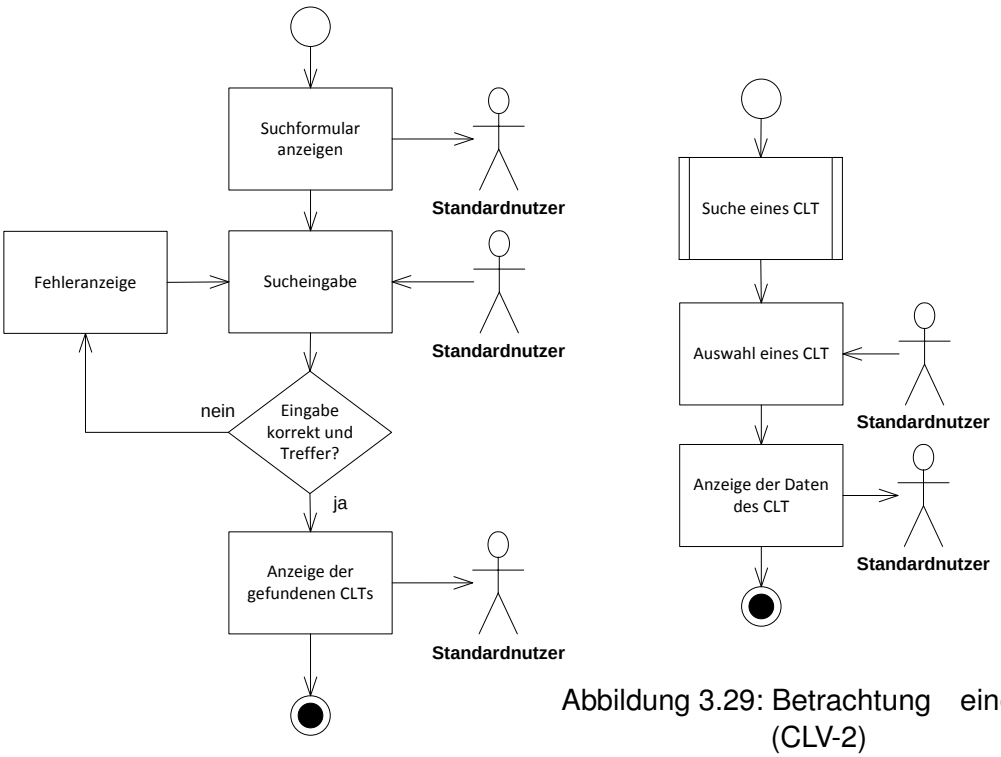


Abbildung 3.29: Betrachtung eines CLT (CLV-2)

Abbildung 3.28: Suche eines CLT (CLV-1)

CLV-3 Instanziierung eines CLT

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte einen CLT instanziieren, um diesen einer ORI zuzuordnen. Nach der Betrachtung des CLT (siehe CLV-2), wählt der Verwalter die Option zur Instanziierung aus.

Häufigkeit: Dieser Vorgang wird immer dann ausgeführt, wenn vordefinierte CLTs als CLIs zu neuen ORIs hinzugefügt werden sollen.

Durchführung: Dem Verwalter wird nach Betrachtung der CLT und Auswahl der Option zur Instanziierung die Möglichkeit gegeben, die CLI zu aktualisieren (siehe CLV-7, Abbildung 3.30).

CLV-4 Erzeugung einer CLI

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte bei der Erzeugung einer neuen ORI (siehe ORV-4) oder der Instanziierung eines ORT (siehe ORV-3), eine CLI einer ORI hinzufügen, für die es keinen passenden CLT gibt.

Häufigkeit: Dieser Vorgang wird durchgeführt, wenn es keinen passenden CLT gibt.

Durchführung: Der Verwalter gibt dazu in einer Eingabemaske Name, Beschreibung und Zustand der neuen CLI ein und erhält anschließend die Möglichkeit weitere Änderungen an der CLI vorzunehmen (siehe CLV-7, Abbildung 3.31).

CLV-5 Suche einer CLI

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte eine ihm zugeordnete CLI suchen, um an dieser zu arbeiten. Alternativ sucht ein Verwalter nach einer CLI um Änderungen an dieser vorzunehmen (siehe CLV-7).

Häufigkeit: Diese Aufgabe ist Teil weiterer Aufgaben der Benutzer und wird entsprechend häufig durchgeführt.

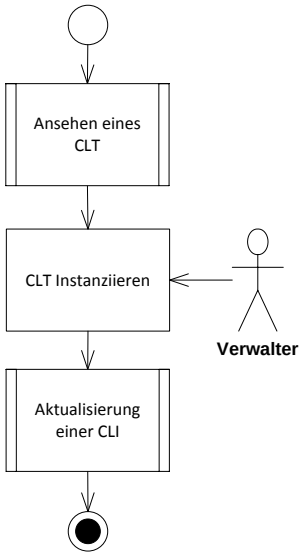


Abbildung 3.30: Instanziierung eines CLT (CLV-3)

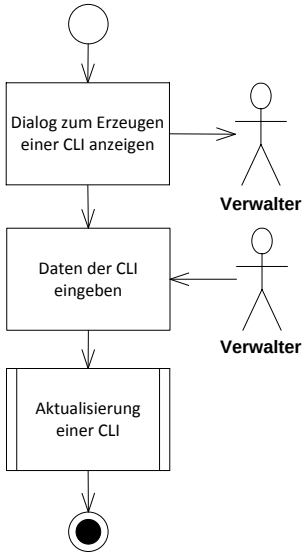


Abbildung 3.31: Erzeugung einer CLI (CLV-4)

Durchführung: Um die Aufgabe durchzuführen, sucht der Benutzer über eine Eingabemaske nach der entsprechenden CLI mithilfe deren Eigenschaften Name und Beschreibung. Alle auf die Eingabe passenden Ergebnisse werden ihm dabei in einer Ergebnisliste angezeigt (siehe Abbildung 3.32).

Ausnahmen: Sind keine passenden Ergebnisse vorhanden, wird ihm eine Hinweismeldung dazu angezeigt.

CLV-6 Betrachtung einer CLI

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte sich die Eigenschaften einer ihm zugeordneten CLI anschauen. Nach der Suche (siehe CLV-5) einer CEI oder in einer Ansicht der ORI-Details (siehe ORV-6) wählt der Benutzer die CLI aus.

Häufigkeit: Die Aufgabe wird sehr häufig durchgeführt, da sie Teil weiterer Aufgaben ist, zum Beispiel dem Abschließen einer CEI (siehe CEV-8).

Durchführung: Er erhält die Ansicht ihrer Eigenschaften mit Name, Beschreibung und Zustand angezeigt. Er sieht außerdem die ORI und untergeordnete CLIs wie auch CEIs der CLI (siehe Abbildung 3.33).

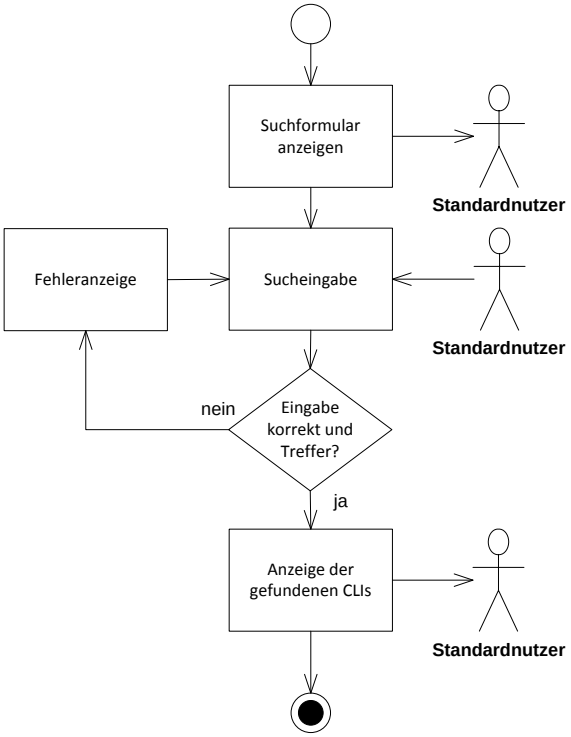


Abbildung 3.32: Suche einer CLI (CLV-5)

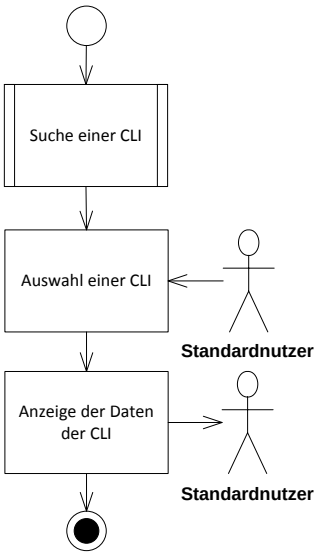


Abbildung 3.33: Betrachtung einer CLI (CLV-6)

CLV-7 Aktualisierung einer CLI

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte eine CLI anpassen, da sich deren Eigenschaften geändert haben oder neue Einträge hinzugefügt werden sollen (siehe CEV-3 und CEV-4). Nach der Betrachtung einer CLI (siehe CLV-6) wählt der Verwalter die Option zur Bearbeitung der CLI aus.

Häufigkeit: Die Aktualisierung einer CLI wird unregelmäßig durchgeführt und hängt individuell von der CLI ab.

Durchführung: Die Eigenschaften der CLI werden mit Eingabefeldern editierbar dargestellt. Nach dem Speichern der Änderungen ist die CLI aktualisiert. Er hat außerdem die Möglichkeit die CEIs der CLI anzusehen, anzupassen (siehe CEV-7) und zu erweitern (siehe CEV-3 und CEV-4, Abbildung 3.34).

Ausnahmen: Bei inkorrekten Eingaben oder nicht ausgefüllten Feldern, wird dem Verwalter ein Hinweis dazu angezeigt.

CLV-8 Abschluss einer CLI

Priorität: mittel

Beschreibung: Wenn alle CEIs einer CLI abgearbeitet sind oder der Verwalter der CLI der Meinung ist, dass diese ausreichend abgearbeitet wurde, so kann dieser die CLI abschließen. Eine abgeschlossene CLI gilt als fertiggestellt und treibt den Fortschritt der übergeordneten CLI oder der zugehörigen ORI voran.

Häufigkeit: Der Abschluss einer CLI stellt eine zentrale Aufgabe dar. Diese wird bei jeder CLI genau einmal durchgeführt, insofern diese zuvor nicht gelöscht wird.

Durchführung: Um die Aufgabe durchzuführen, ändert der Verwalter in der Betrachtung der CLI (siehe CLV-6) den Status der CLI auf abgeschlossen, woraufhin er einen Warndialog bestätigen muss. Die abgeschlossene CLI wird anschließend visuell entsprechend markiert und vom System archiviert (siehe Abbildung 3.35).

Ausnahmen: Wenn der Warndialog nicht bestätigt wird, schließt sich dieser, ohne dass eine Aktion durchgeführt wurde.

CLV-9 Löschen einer CLI

Priorität: sehr hoch

Beschreibung: Wenn eine CLI in einer ORI nicht länger benötigt wird, wird diese von dem Verwalter der CLI gelöscht. Die enthaltenen CEIs werden dabei ebenfalls mitgelöscht.

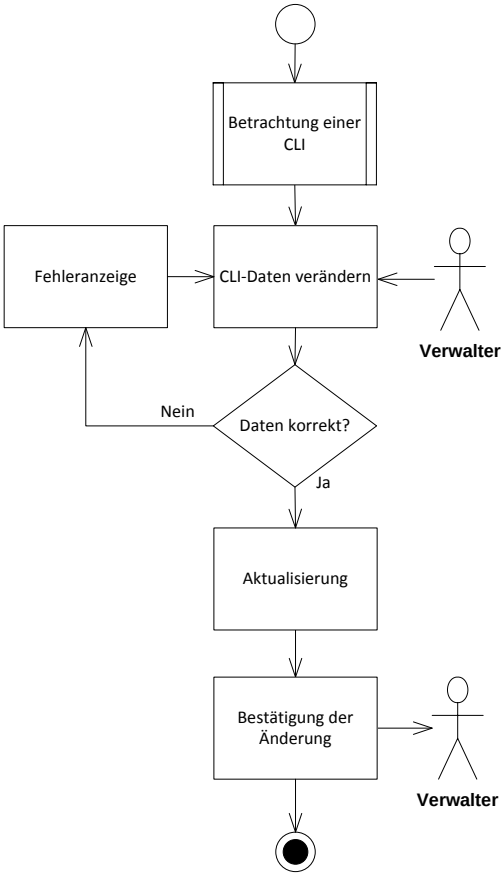


Abbildung 3.34: Aktualisierung einer CLI (CLV-7)

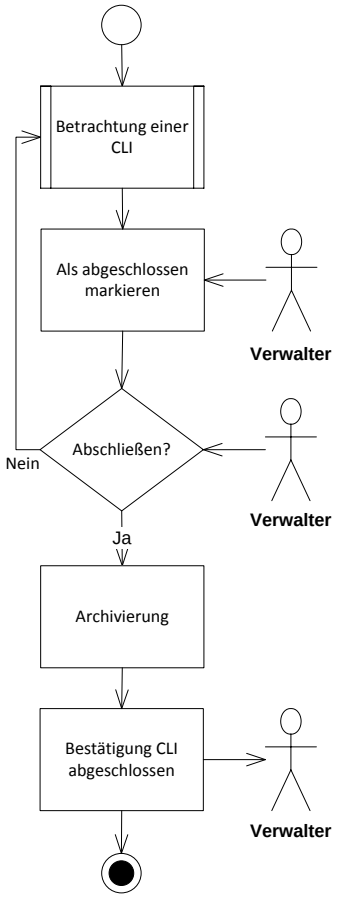


Abbildung 3.35: Abschluss einer CLI (CLV-8)

Häufigkeit: Diese Aufgabe wird selten durchgeführt, da nicht benötigte CEI (siehe CEV-9) einzeln gelöscht und die CLI auch den geänderten Umständen angepasst werden kann (siehe CLV-7).

Durchführung: Der Verwalter sucht dazu die CLI (siehe CLV-5) oder wählt diese aus der übergeordneten ORI (siehe ORV-6) aus. Anschließend wählt der die Option zum Löschen der CLI und bestätigt die darauf folgende Warnung (siehe Abbildung 3.36).

Ausnahmen: Wird der Warndialog nicht bestätigt, schließt sich dieser und keine Aktion findet statt.

CLV-10 Verschiebung von CEIs

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte eine oder mehrere CEIs aus einer CLI auf eine andere Position verschieben oder diese einer anderen CLI unterordnen. Er macht dies um Reihenfolgen anzupassen oder diese anders zu gliedern. Auch beim Einfügen von neuen CEIs (siehe CEV-3 und CEV-4) müssen diese häufig noch an die richtige Position verschoben werden.

Häufigkeit: Die Aufgabe wird häufig vom Verwalter durchgeführt.

Durchführung: Um einen Verschiebevorgang durchführen zu können, macht er die CLI editierbar (siehe CLV-7). Anschließend kann er CEIs auswählen und per Drag & Drop auf eine andere Position verschieben. Ihm wird dabei visuell angezeigt, wohin die CEI verschoben wird (siehe Abbildung 3.37).

CLV-11 Zusammenfügen von CLIs

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte eine oder mehrere CLIs zu einer einzigen zusammenfügen.

Häufigkeit: Dies wird immer dann durchgeführt, wenn ähnliche Abläufe vereinheitlicht werden oder eine Vereinfachung einer Gliederung stattfindet.

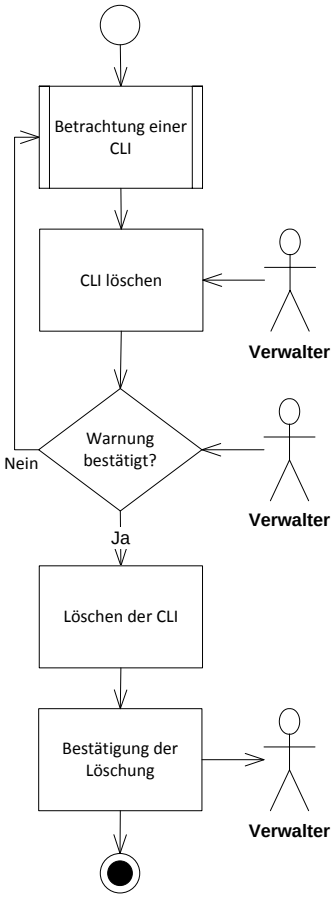


Abbildung 3.36: Löschen einer CLI (CLV-9)

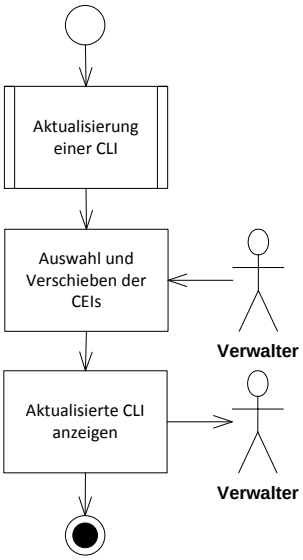


Abbildung 3.37: Verschiebung von CEIs (CLV-10)

Durchführung: Dazu macht der Verwalter die übergeordnete ORI oder die übergeordnete CLI editierbar (siehe ORV-7 und CLV-7). Anschließend wählt er eine der beiden CLIs aus, und zieht diese per Drag & Drop auf die andere CLI. Ihm wird ein Dialog angezeigt, bei dem er sich zum Zusammenfügen der CLI entscheiden muss. Er hat dabei die Möglichkeit die Elemente der ersten zur zweiten CLI hinzuzufügen oder den umgekehrten Fall (siehe Abbildung 3.38).

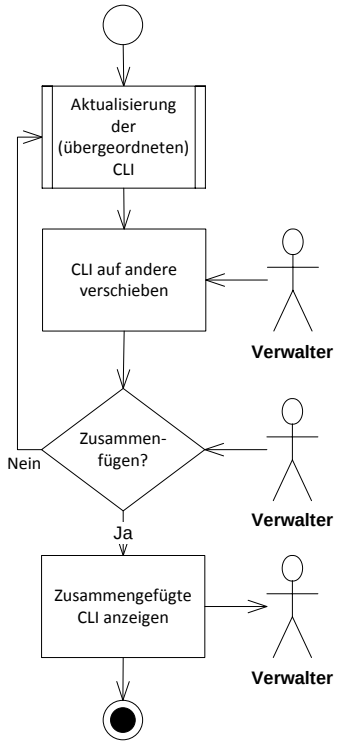


Abbildung 3.38: Zusammenfügen von CLIs (CLV-11)

3.3.4 Verwaltung von Checklisteneinträgen

Die folgenden Aufgaben und Abläufe beziehen sich auf die Benutzung und Verwaltung der CETs und CEIs, die die Blattebene in der hierarchischen Struktur bilden. Weitere Aufgaben, welche die Erzeugung der CET einschließt, werden nur vom Administrator unterstützt.

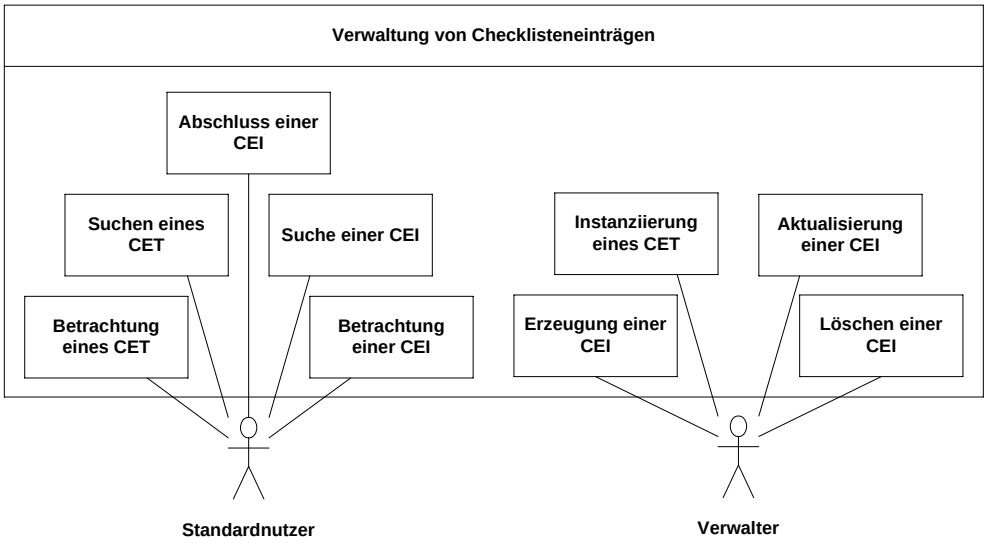


Abbildung 3.39: Anwendungsfälle des Standardnutzers und Verwalters zur Verwaltung der Einträge

CEV-1 Suche eines CET

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte einen CET instanziieren (siehe CEV-3) oder die Attribute (siehe CEV-2) eines CET ansehen. Dazu muss er den entsprechenden CET zuvor suchen.

Häufigkeit: Eine Suche findet jedes mal statt, wenn eine Instanziierung eines vordefinierten CET geschehen soll.

Durchführung: Sie wird von einem Standardnutzer durchgeführt, indem er in einer Eingabemaske nach den Eigenschaften Name, Beschreibung eines CET sucht. Alle auf die Eingabe passenden Ergebnisse werden ihm dabei in einer Ergebnisliste angezeigt (siehe Abbildung 3.40).

Ausnahmen: Sind keine passenden Ergebnisse vorhanden, wird ihm eine Hinweismeldung dazu angezeigt.

CEV-2 Betrachtung eines CET

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte sich die Details eines CET anschauen, um diesen beispielsweise anschließend zu instanziieren. Er kann diesen nach der Suche des CET (siehe CEV-1) oder in einer Detailansicht von CLTs (siehe CLV-6) auswählen.

Häufigkeit: Dieser Vorgang wird vor jeder Instanziierung eines CET (siehe CEV-3) durchgeführt.

Durchführung: Er erhält nach der Auswahl eines CET eine Ansicht der Eigenschaften mit Name und Beschreibung angezeigt (siehe Abbildung 3.41).

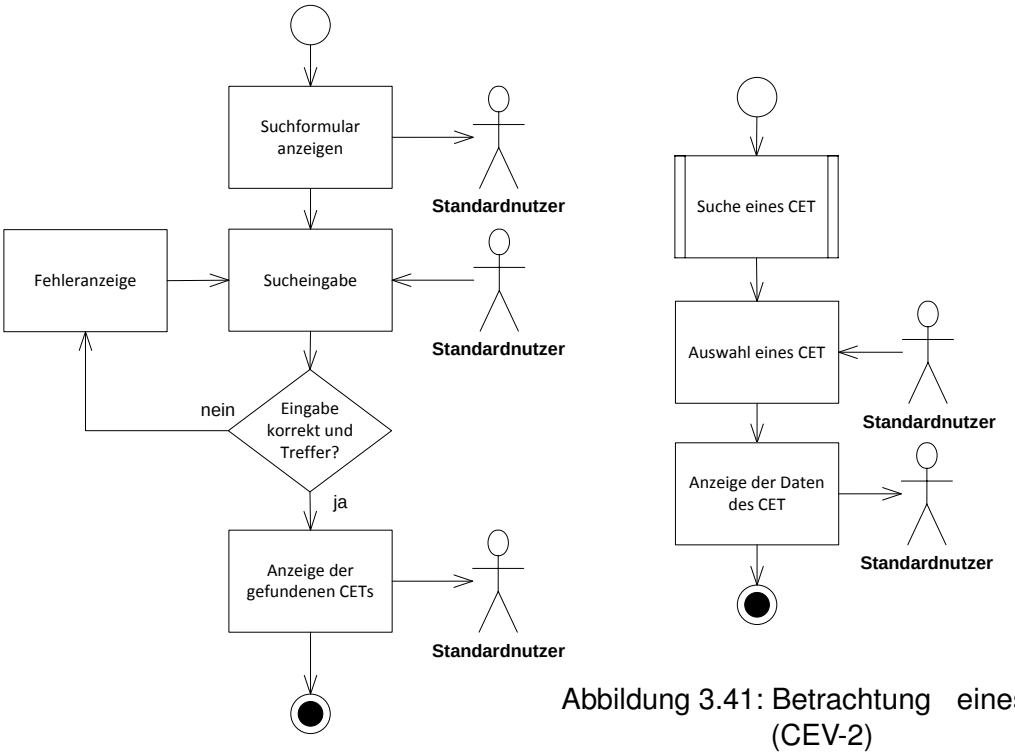


Abbildung 3.41: Betrachtung eines CET (CEV-2)

Abbildung 3.40: Suche eines CET (CEV-1)

CEV-3 Instanziierung eines CET

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte einen CET instanziieren, um diesen einer CLI zuzuordnen.

Häufigkeit: Dieser Vorgang wird immer dann ausgeführt, wenn eine neue CEI eines vordefinierten CET benötigt wird.

Durchführung: Nach der Betrachtung des CET (siehe CEV-2), wählt er die Option zur Instanziierung aus. Ihm wird anschließend die Möglichkeit die erstellte CEI zu aktualisieren (siehe CEV-7), um Beispielsweise die Beschreibung anzupassen (siehe Abbildung 3.42).

Ausnahmen: Bei inkorrekten Eingaben oder nicht ausgefüllten Feldern wird dem Verwalter ein Hinweis dazu angezeigt.

CEV-4 Erzeugung einer CEI

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte bei der Erzeugung einer neuen CLI (siehe CLV-4) oder der Instanziierung eines CLT (siehe CLV-3), einen noch nicht als CET vorhandenen CE dieser CLI hinzufügen.

Häufigkeit: Dieser Vorgang wird immer durchgeführt, wenn ein CE noch nicht als CET definiert ist und als CEI erstellt werden soll.

Durchführung: Er gibt dazu in einer Eingabemaske Name, Beschreibung und Zustand des neuen Items ein und erhält anschließend die Möglichkeit weitere Änderungen an diesem vorzunehmen (siehe CEV-7, Abbildung 3.43).

Ausnahmen: Bei inkorrekten Eingaben oder nicht ausgefüllten Feldern wird dem Verwalter eine Hinweismeldung dazu angezeigt.

CEV-5 Suche einer CEI

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte eine CEI aus denen ihm zugeordneten CLI suchen, um diese beispielsweise abzuschließen (siehe CEV-8).

Häufigkeit: Diese Aufgabe wird sehr häufig durchgeführt, da sie Teil der Hauptaufgaben des Benutzers ist.

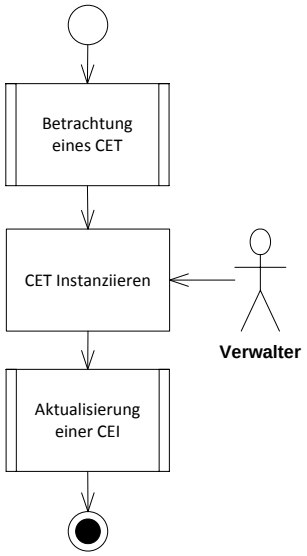


Abbildung 3.42: Instanziierung eines CET (CEV-3)

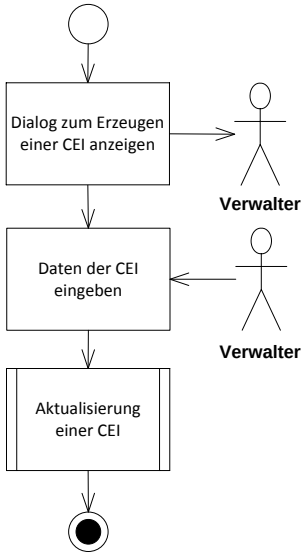


Abbildung 3.43: Erzeugung einer CEI (CEV-4)

Durchführung: Um die Aufgabe durchzuführen sucht der Benutzer über eine Eingabemaske nach der entsprechenden CEI über die Eigenschaften Name, Text und Zustand. Alle auf die Eingabe passenden Ergebnisse werden ihm dabei in einer Ergebnisliste angezeigt (siehe Abbildung 3.44).

Ausnahmen: Sind keine passenden Ergebnisse vorhanden, wird ihm eine Hinweismeldung dazu angezeigt.

CEV-6 Betrachtung einer CEI

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte sich die Eigenschaften einer ihm zugeordneten CEI anschauen. Nach der Suche (siehe CEV-5) einer CEI oder in einer Ansicht der CLI-Details (siehe CLV-6) wählt der Benutzer diese CEI aus.

Häufigkeit: Dies wird sehr häufig von ihm getan, um beispielsweise den Zustand einer CEI zu prüfen.

Durchführung: Nach der Auswahl der CEI erhält er die Ansicht ihrer Eigenschaften mit Name, Beschreibung und Zustand angezeigt (siehe Abbildung 3.45).

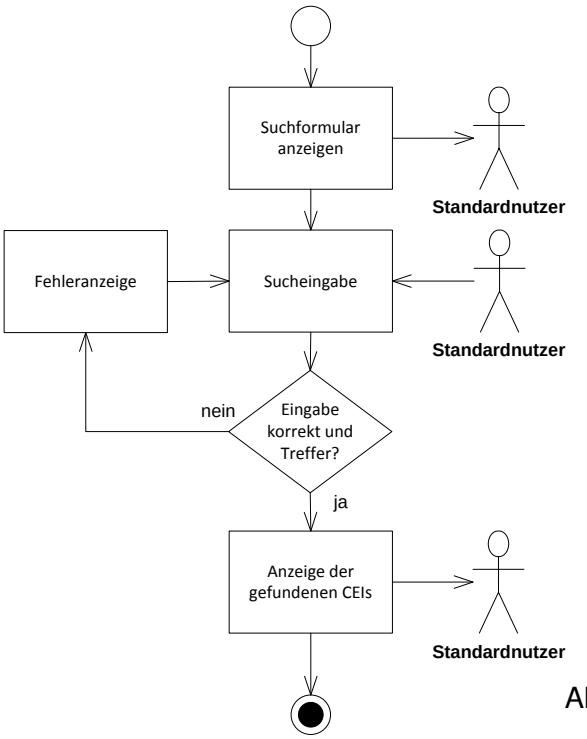


Abbildung 3.44: Suche einer CEI (CEV-5)

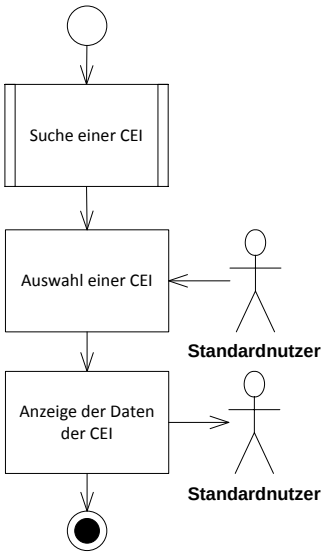


Abbildung 3.45: Betrachtung einer CEI (CEV-6)

CEV-7 Aktualisierung einer CEI

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte die Eigenschaften einer CEI anpassen. Nach der Betrachtung einer CEI (siehe CEV-6) wählt er die Option zur Bearbeitung der CEI aus.

Häufigkeit: Das Aktualisieren einer CEI wird selten durchgeführt. Mindestens jedoch einmalig, wenn die CEI abgeschlossen wird (siehe CEV-8).

Durchführung: Die Eigenschaften werden mit Eingabefeldern editierbar dargestellt. Zu diesen zählen Name, Text und Zustand. Nach dem Speichern der Änderungen ist die CEI aktualisiert (siehe Abbildung 3.46).

CEV-8 Abschluss einer CEI

Priorität: sehr hoch

Beschreibung: Ein Standardnutzer möchte, nach erfüllter Aufgabe, die zugehörige CEI als abgeschlossen markieren.

Häufigkeit: Eine abgeschlossene CEI treibt den Fortschritt der übergeordneten CLI voran

Durchführung: In der Betrachtung der CEI (siehe CEV-6) ändert er dazu den Status dieser auf abgeschlossen (siehe Abbildung 3.47).

CEV-9 Löschen einer CEI

Priorität: sehr hoch

Beschreibung: Ein Verwalter möchte eine CEI aus einer ihm zugeordneten CLI löschen.

Häufigkeit: Diese Aufgabe wird selten durchgeführt.

Durchführung: Er sucht dazu die CEI (siehe CEV-5) oder wählt diese aus der Übersicht in der CLI (siehe CLV-6) aus. Anschließend wählt der die Option zum Löschen der CEI und bestätigt die folgende Warnung (siehe Abbildung 3.48).

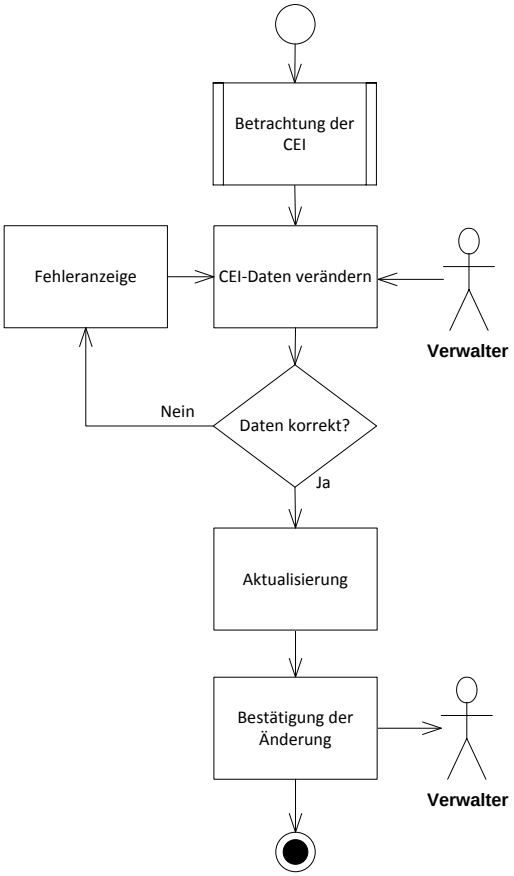


Abbildung 3.46: Aktualisierung einer CEI (CEV-7)

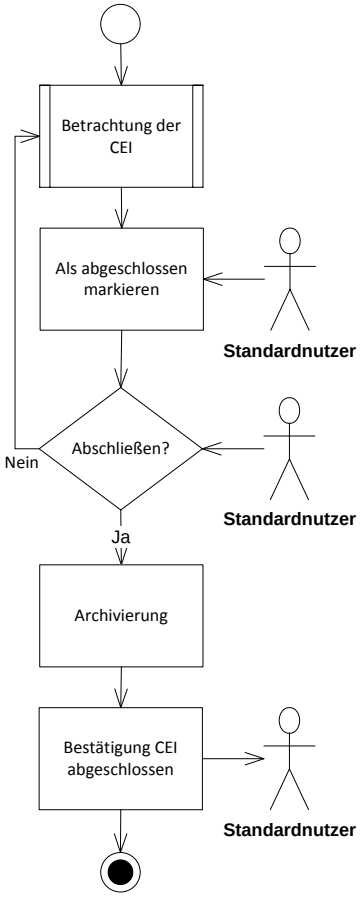


Abbildung 3.47: Abschluss einer CEI (CEV-8)

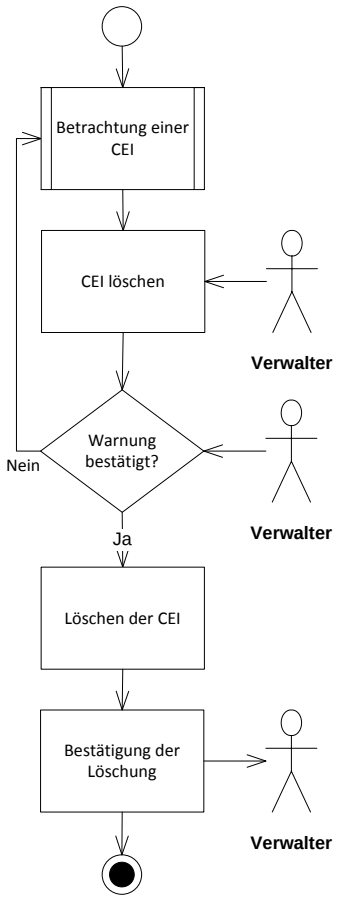


Abbildung 3.48: Löschen einer CEI (CEV-9)

3.3.5 Übergreifende Verwaltungsmaßnahmen

Übergreifende Verwaltungsmaßnahmen beschreiben Aufgaben, die zwischen BV, ORV, CLV und CEV stattfinden. Diese sind in Übersicht in Abbildung 3.49 zu sehen.

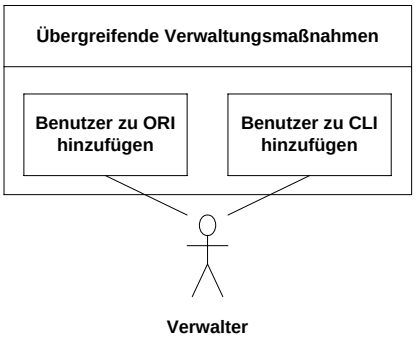


Abbildung 3.49: Anwendungsfälle von übergreifenden Verwaltungsmaßnahmen zwischen Benutzern, Rahmen und Listen

UV-1 Benutzer zu ORI hinzufügen

Priorität: sehr hoch

Beschreibung: Der Verwalter einer ORI möchte weitere Verwalter oder Standardnutzer zu einer ihm zugehörigen ORI hinzufügen.

Häufigkeit: Das Benutzermanagement innerhalb von ORIs stellt einen wichtigen Bestandteil der Aufgaben des Verwalters dar. Diese Aufgabe wird dementsprechend häufig durchgeführt.

Durchführung: Um einen Benutzer der ORI hinzuzufügen, muss der Verwalter diese editierbar machen (siehe ORV-7). Anschließend sucht er nach dem entsprechenden Benutzer (siehe BV-3). Aus der Ergebnisliste kann er nun per Drag & Drop den gewünschten Benutzer in die Nutzerliste der ORI einfügen. Nach dem Hinzufügen kann er die gewünschte Rolle auswählen (siehe Abbildung 3.50).

Ausnahmen: Der selbe Benutzer sollte nicht mehrfach in verschiedenen Rollen einer ORI zugeordnet werden können.

UV-2 Benutzer zu CLI hinzufügen

Priorität: hoch

Beschreibung: Der Verwalter einer CLI möchte Benutzer zu einer ihm zugehörigen CLI hinzufügen.

Häufigkeit: Das Benutzermanagement innerhalb von CLI stellt einen wichtigen Bestandteil der Aufgaben des Verwalters dar. Diese Aufgabe wird dementsprechend häufig durchgeführt.

Durchführung: Um einen Benutzer der CLI hinzuzufügen, muss der Verwalter diese editierbar machen (siehe CLV-7). Anschließend sucht er nach dem entsprechenden Benutzer (siehe BV-3). Aus der Ergebnisliste kann er nun per Drag & Drop den gewünschten Benutzer in die Nutzerliste der CLI einfügen (siehe Abbildung 3.51).

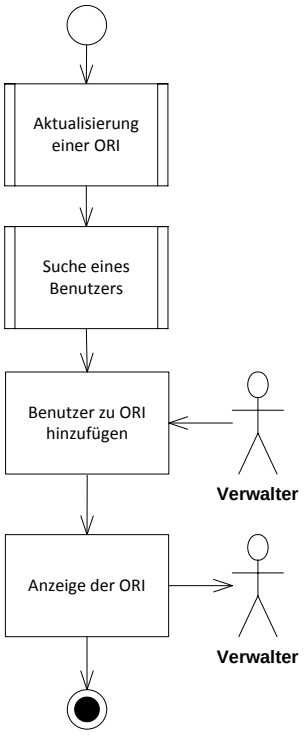


Abbildung 3.50: Benutzer zu ORI hinzufügen (UV-1)

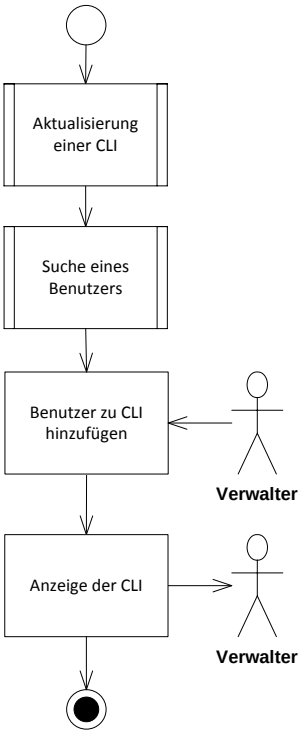


Abbildung 3.51: Benutzer zu CLI hinzufügen (UV-2)

3.4 Kontextanalyse der Umgebungsbedingungen

Die Umgebung, in welcher die Anwendung genutzt werden soll, ist für das visuelle Design und die Interaktion mit der Anwendung wichtig.
Für pCT wird angenommen, dass dieses hauptsächlich innerhalb von Gebäuden verwendet wird. Dabei ist davon auszugehen, dass im Arbeitsumfeld eine gute Abdeckung mit WLAN[2] gegeben ist und konstante Lichtverhältnisse herrschen. Über die Geräuschkulisse kann keine Angabe gemacht werden, deshalb wird die Möglichkeit des auditiven Feedbacks im Folgenden ignoriert.

3.5 Software- und Hardware-Randbedingungen

Die Randbedingungen unter denen die Anwendung entwickelt wird, sind von entscheidender Bedeutung für die Entwurfs- und Implementierungsphase. Zu diesen Randbedingungen zählen softwareseitige Bedingungen, wie das Betriebssystem, aber auch die Vorgaben der Hardware. Zunächst soll jedoch die Hardware des Tablets an sich definiert werden.

3.5.1 Tablets

Die IDC[3] definiert ein Tablet folgendermaßen (aus dem Englischen):

> „Die IDC betrachtet alle LCD[4]-basierten flachen Geräte mit Bildschirmen zwischen 7 und 16 Zoll als Tabelts, egal ob sie mit einer entfernbaren Tastatur ausgestattet sind (wie das Surface RT). Umwandelbare Geräte mit fest verbauten Tastaturen (wie das Lenovo Yoga) werden nicht als Tablets gezählt.“ [MRS13a]

Tablets werden außerdem in der Regel über Stift oder Berührungsgesten bedient. Sie sind des Weiteren, ähnlich dem Smartphone, mit einer großen Anzahl von Features

[2] Wireless Local Area Network
[3] International Data Corporation: `http://www.idc.com`
[4] Liquid Crystal Display - Flüssigkristallbildschirm

versehen, zu denen zum Beispiel eine umfangreiche sensorische Ausstattung gehört. Dazu zählen Gyroskop (Lage), Accelerometer (Beschleunigung), Magnetometer (Kompassfunktion), Näherungs- , Helligkeitssensoren und digitale Kameras.
Sie sind handlich und mobil, was eine Verwendung ohne die Einschränkungen der Größe und des Gewichts eines Laptops ermöglicht. Sie bieten, im Gegensatz zu den kleineren Smartphones, hingegen genügend Platz, Informationen anschaulich und übersichtlich zu präsentieren.
Mit diesen Eigenschaften stellen Tablets einen Ersatz für Notizblöcke dar. Aber auch in Bereichen wie Krankenhäusern können Tablets verwendet werden, um während Visiten Patientenakten abzurufen. Hierdurch müssen diese nicht mehr handschriftlich gepflegt werden und Probleme, wie das Vertauschen von Akten, können vermieden werden.

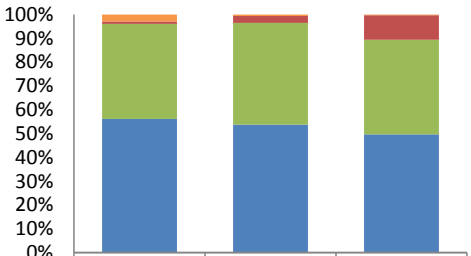
3.5.2 Software-Randbedingungen

Zu den Software-Randbedingungen zählt vor allem das Betriebssystem, unter dem die Anwendung laufen soll. Auf dem Markt für Tablets sind eine Vielzahl von Betriebssysteme vorhanden. Deshalb ist es nötig, auf Marktanalysen und Trends zurückzugreifen um eine Entscheidung für die Wahl zu treffen.
Auf Basis der Marktanalyse von [MRS12], dargestellt in Abbildung 3.52, kommen zwei unterschiedliche Betriebssysteme in Frage, unter denen die Anwendung laufen muss. Eine mögliche Erweiterung sind Geräte, die mit Windows RT oder Windows 8 laufen.

- Google Android
- Apple iOS
- Microsoft Windows RT/8

3.5.3 Hardware-Randbedingungen

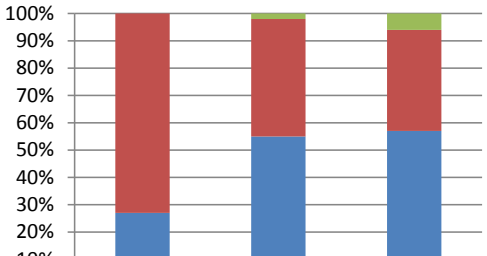
Im Bereich der Apple iOS Geräte, die wie in Abschnitt 3.5.2 beschrieben die Hälfte des Marktes ausmachen, zählen die zehn Zoll großen iPad und iPad 2, sowie das sieben Zoll große iPad Mini. Bei den Androidgeräten unter acht Zoll handelt es sich zum Großteil um sieben Zoll Geräte, wie das Samsung Galaxy Tab, Samsung Galaxy Tab 2 7.0, das



	2011	2012	2016
Andere	2,90%	0,60%	0,30%
Windows	1%	2,90%	10,30%
Android	39,80%	42,70%	39,70%
iOS	56%	53,80%	49,70%

Abbildung 3.52: Entwicklung und Prognose der Marktverteilung von Tabletbetriebssystemen

Google Nexus 7 und die Amazon Kindle Fire Geräte [Ani13]. Die Androidgeräte über acht Zoll werden von den zehn Zoll Geräten dominiert, zu denen das Samsung Galaxy Tab 2 10.1, Samsung Galaxy Note 10.1 und das Asus Transformer zählen. Die Statistik zu den Androidgeräten ist in Abbildung 3.53 dargestellt.



	2011	2013	2017
11"+	0%	2%	6%
8" - 11"	73%	43%	37%
< 8"	27%	55%	57%

Abbildung 3.53: Entwicklung und Prognose der Marktverteilung von Tabletgrößen

Ein Ziel für pCT ist demnach eine übersichtliche Darstellung und eine gute Bedienung, speziell auf diesen Größenformaten.

4

Entwurf

Basierend auf der Analyse der Anforderungen in Kapitel 3 wird im Folgenden der Entwurf für die Anwendung vorgestellt. Dabei werden zuerst die Komponenten des Systems erläutert, das Datenmodell erstellt, Entwurfsentscheidungen getroffen und abschließend ein Gestaltungskonzept der Anwendung erarbeitet. Als grafische Übersicht ist dieses Kapitel in Abbildung 4.1 dargestellt.

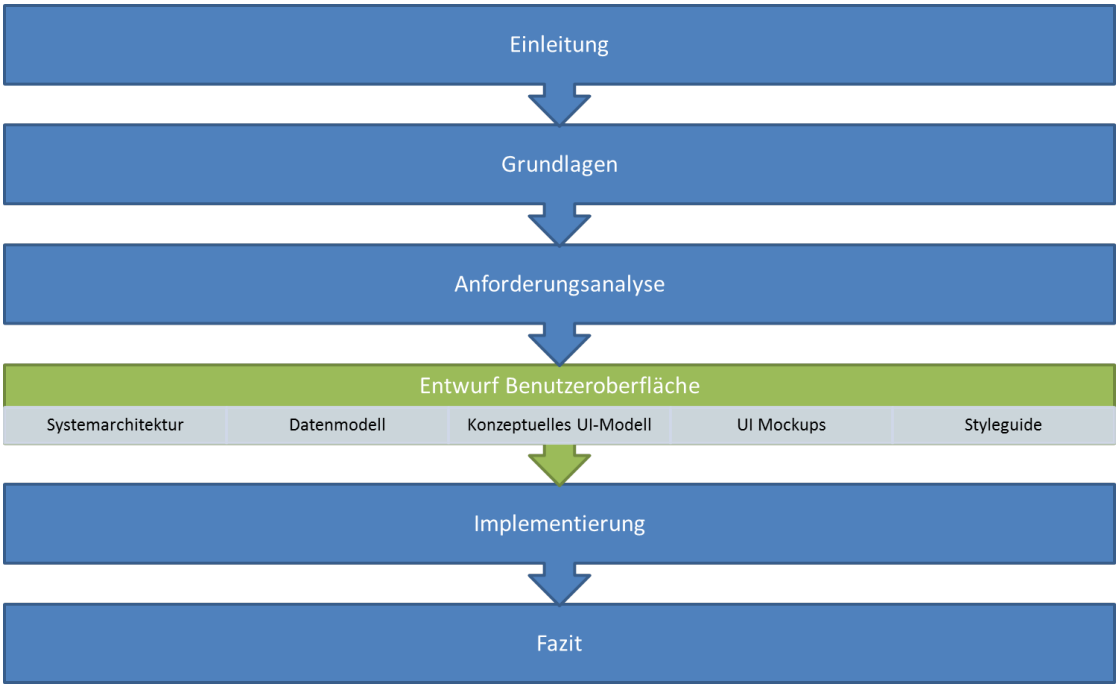


Abbildung 4.1: Aufbau Kapitel 4 — Entwurf

4.1 Architektur

Bei der Architektur des Systems wird im Folgenden zwischen der Gesamtarchitektur des proCollab Prototyps und der Architektur von pCT im Speziellen unterschieden.

4.1.1 proCollab Gesamtarchitektur

Der proCollab Prototyp besteht insgesamt aus einem mehrschichtigen System, deren Komponenten miteinander interagieren (siehe Abbildung 4.2). Auf der untersten Schicht geht es um die Persistenz der Daten, also das Abspeichern mittels einer Datenbanklösung [Rei13].
Die Schicht darüber regelt die Logik der Verwaltungsmechanismen. Hier werden zum Beispiel Änderungsoperationen auf den Daten durchgeführt und an die Persistenzschicht weitergegeben [Zie13].
Über der Anwendungsschicht liegt die Schnittstellenschicht. Hier werden Dienste für

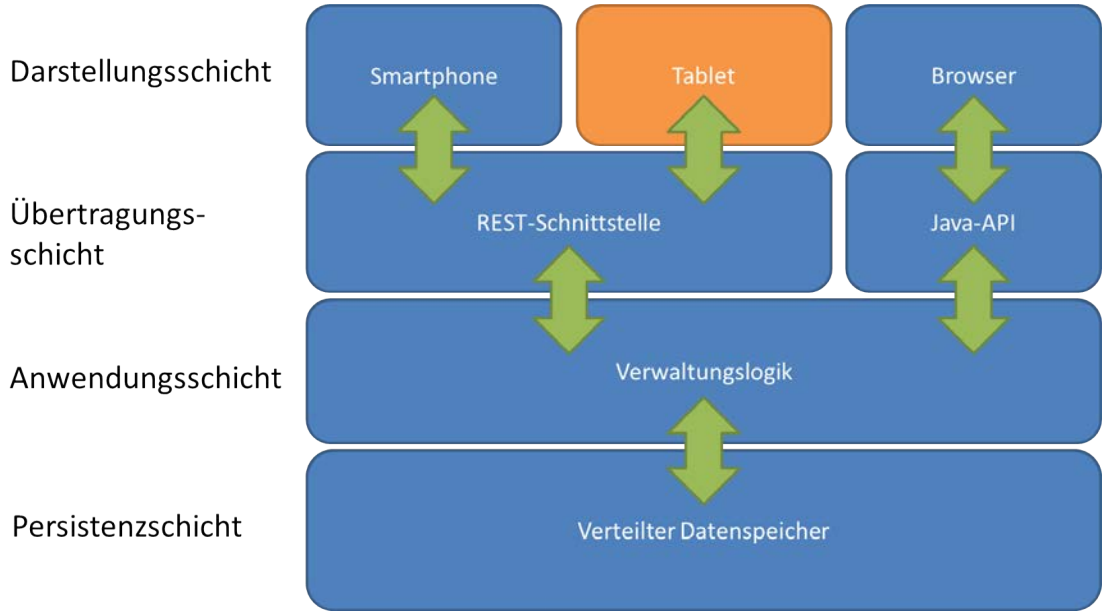


Abbildung 4.2: Architektur des gesamten Prototyps

die Darstellungsschicht bereitgestellt, um Daten empfangen und senden zu können. Diese stellt demnach die Schnittstelle zwischen den darstellenden Clients und dem verwaltenden Server dar [Zie13].

Auf der obersten Schicht, der Darstellungsschicht, wird dem Benutzer eine grafische Oberfläche angeboten, über die dieser Aktionen mit dem System durchführen kann. Die Schicht ist in drei Bereiche unterteilt, die jeweils eine andere Art der Darstellung bieten: Eine webbasierte Administrationsoberfläche [Thi13], sowie zwei Anwendungen für mobile Endgeräte. Während in dieser Arbeit eine Anwendung für Tablets (pCT) entwickelt wird, wird in einer verwandten Arbeit eine Anwendung für Smartphones konzipiert [Gei13].

Die Darstellungsschicht ist über eine REST-Schnittstelle (siehe Kapitel 5.4.1) mit der Schnittstellenschicht und über diese mit der Anwendungslogik und der Persistenzschicht verknüpft.

4.1.2 pCT Architektur

Die Architektur von pCT ist in die drei Hauptkomponenten View, Controller und Core aufgeteilt. Damit ist sie an das klassische Model-View-Controller-Muster (MVC) aus der Softwareentwicklung angelehnt [LR01]. Die Funktion der einzelnen Komponenten wird im Folgenden erläutert. Die Vorstellung des Datenmodells folgt in Kapitel 4.2.

View: Die *View* enthält die Information über den Aufbau und das Aussehen der Benutzeroberfläche auf dem Tablet-Bildschirm.

Controller: Der *Controller* enthält die Logik zum Zugriff auf die View, um in dieser Daten darzustellen, Daten vom Benutzer einzulesen und auf Ereignisse zu reagieren, die von diesem ausgelöst werden.

Core: Der *Core* enthält Unterkomponenten, wie die REST-Schnittstelle, lokale Datenspeicher und Objekte, die vom Controller verwendet werden. Zu letzteren zählt zum Beispiel ein Listenobjekt, das Funktionalitäten wie den Aufbau der View einer Liste oder das Sortieren dieser bereitstellt.

Diese Architektur ermöglicht eine klare Aufteilung der Funktionalitäten und reduziert die Gesamtkomplexität der Anwendung. Die Anbindung an das Backend des procollab Prototyps geschieht über die zuvor erwähnte REST-Schnittstelle. Der Zusammenhang der Komponenten und die Anbindung aus Clientsicht ist in Abbildung 4.3 dargestellt.

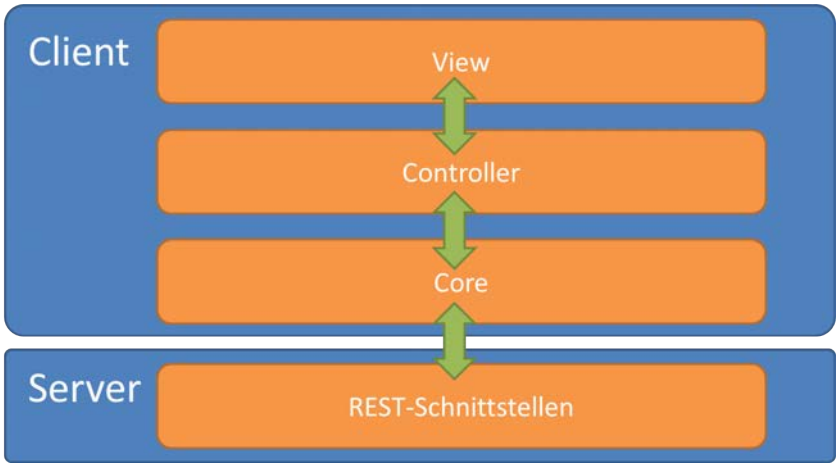


Abbildung 4.3: Architektur von pCT

4.2 Datenmodell

Aus der Anforderungsanalyse (siehe Kapitel 3), sowie den Eigenschaften der einzelnen Elemente (siehe Kapitel 2) ergeben sich die Beziehungen und Datenobjekte wie in Abbildung 4.4 dargestellt.

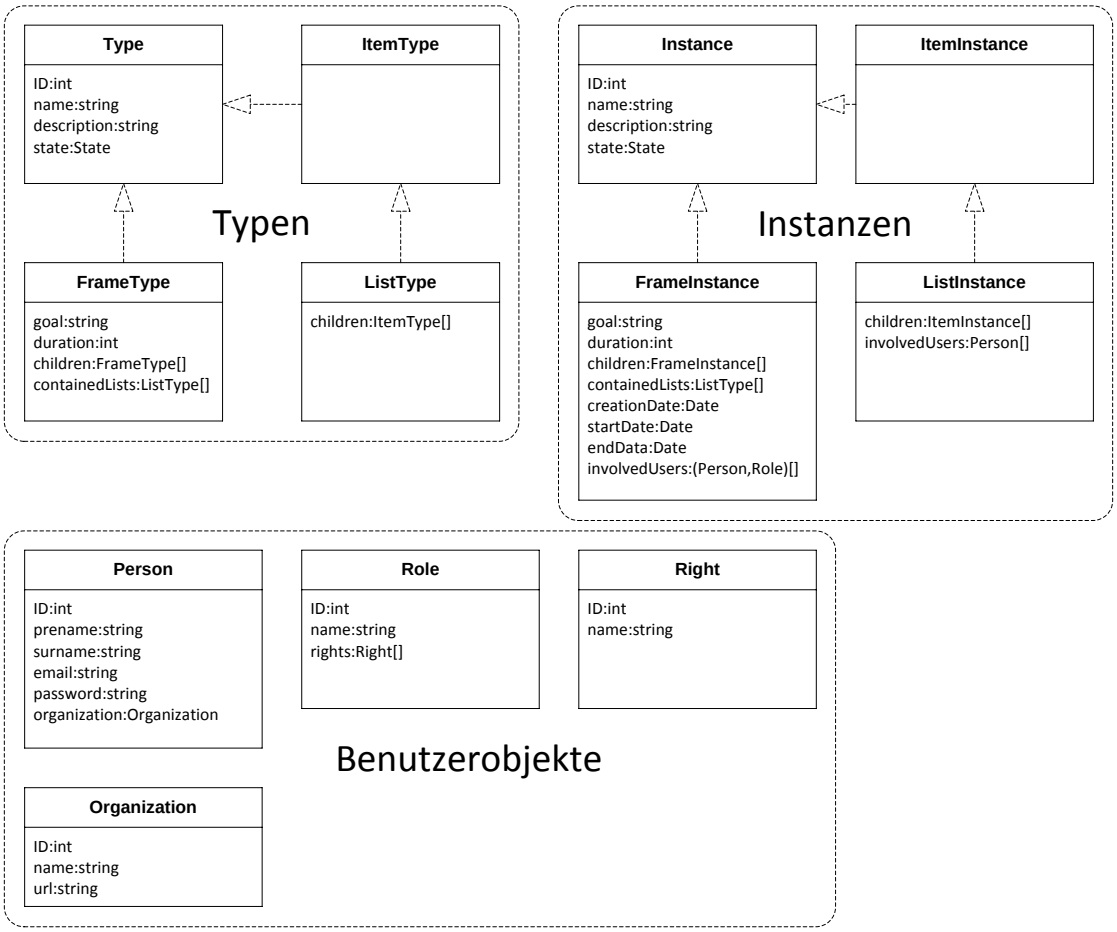


Abbildung 4.4: Datenmodell von pCT

Es wurde eine hierarchische Beziehung zwischen OR, CL und CE aufgebaut, zwischen deren ihren Typen und Instanzen unterschieden, sowie die Rechte einer Person in der Anwendung an eine Rollengruppe geknüpft.

4.2.1 Typen

In Kapitel 3 wurden die drei Typen ORT, CLT und CET eingeführt. Genau diese sind auch im Datenmodell durch *FrameType*, *ListType* und *ItemType* abgebildet. Hierbei wurden die gemeinsamen Attribute einer eindeutigen ID, Name, Beschreibung und Zustand auf eine Superklasse *Type* ausgelagert. Außerdem müssen hierarchische Einordnungen beachtet werden. So kann der vom Type abgeleitete *FrameType* nur FrameTypes als Kinder haben. Er kann jedoch auch *ListTypes* enthalten. Weitere Eigenschaften des FrameType sind das Ziel und seine Dauer.
Ein ListType kann sowohl ListTypes, als auch ItemTypes enthalten. Deshalb wird der ListType von der Superklasse ItemType abgeleitet, der ListType ist demnach eine mit Kindern spezialisierte Version des ItemTypes.

4.2.2 Instanzen

Die in Kapitel 3 vorgestellten ORI, CLI und CEI werden im Datenmodell von den Klassen *FrameInstance*, *ListInstance* und *ItemInstance* abgebildet. Die Vererbungshierarchie ist dabei identisch zu den Typen (siehe Kapitel 4.2.1) aufgebaut. Auch die Grundmenge an Attributen ist identisch. So hat die Superklasse *Instance*, die gleichen Attribute wie ein Type. Zu Unterschieden kommt es hauptsächlich in der *FrameInstance*, welche sich durch Start-, End-, und Erstellungsdatum von einem FrameType unterscheidet.
FrameInstance und ListInstance besitzen außerdem das Attribut *involvedUsers*. Bei der FrameInstance ermöglicht dies die Zuordnung von beteiligten Personen in einer Rolle wie dem Verwalter oder Standardnutzer.
Bei den *involvedUsers* einer ListInstances ist keine Rolle nötig, denn diese ist immer ein Bearbeiter aus der Menge der beteiligten Personen der übergeordneten FrameInstance.

4.2.3 Benutzerobjekte

Die Benutzerobjekte bestehen aus dem Personenobjekt, der Organisation, Rollen und Rechten. Die Person ist eindeutig über ihre ID referenzierbar. Sie besitzt Vor- und

Nachnamen, eine E-Mail Adresse und Passwort zur Anmeldung, und gehört einer Organisation an. Die Organisation wird ebenfalls über eine ID referenziert und besteht aus dem Namen der Organisation und der URL der Website. Dies macht es möglich, dass mehrere Personen einer Organisation zugehörig sein können.
Die Rollen und Rechte enthalten auch eine ID und einen Namen. Dabei sind jeder Rolle Rechte zugeordnet, die bestimmen, was eine Person in der jeweiligen Rollen darf.

4.3 Konzeptuelles Modell der Benutzerschnittstelle

Im konzeptuellen Modell der Benutzerschnittstelle geht es darum, die Grundlagen für die Gestaltungsrichtlinien zu finden. Es sollen die benötigten Ansichten identifiziert und Navigationspfade zwischen diesen festgelegt werden.
Dazu werden im Folgenden die in Kapitel 3.3 identifizierten Aufgaben zu hierarchischen Abläufen verknüpft. Außerdem können aus den Aufgaben generelle Ansichten abgeleitet werden. Aus diesen Ergebnissen lassen sich Navigationsstrukturen erstellen welche in den Dialogen der Benutzeroberflächen vereinigt werden.

4.3.1 Aufgabenhierarchie

Die Zusammenhänge der in Kapitel 3.3 analysierten Aufgaben lassen sich hierarchisch wie in Abbildung 4.5 darstellen. Dabei bilden die vier Arten der Verwaltung die Kategorien der Aufgaben. Die Aufgaben selbst sind so strukturiert, dass zuerst übergeordnete Aufgaben erfüllt sein müssen, um die daruntergelegenen Aufgaben erfüllen zu können.

4.3.2 Wesentliche Ansichten und Darstellungsinhalte

Aus der Aufgabenanalyse (siehe Kapitel 3.3) ergeben sich die folgenden Typen von Ansichten:

Anmeldung: Eine Ansicht, die der Anmeldung in die Anwendung dient.
Registrierung: Eine Ansicht, die der Registrierung für die Anwendung dient.

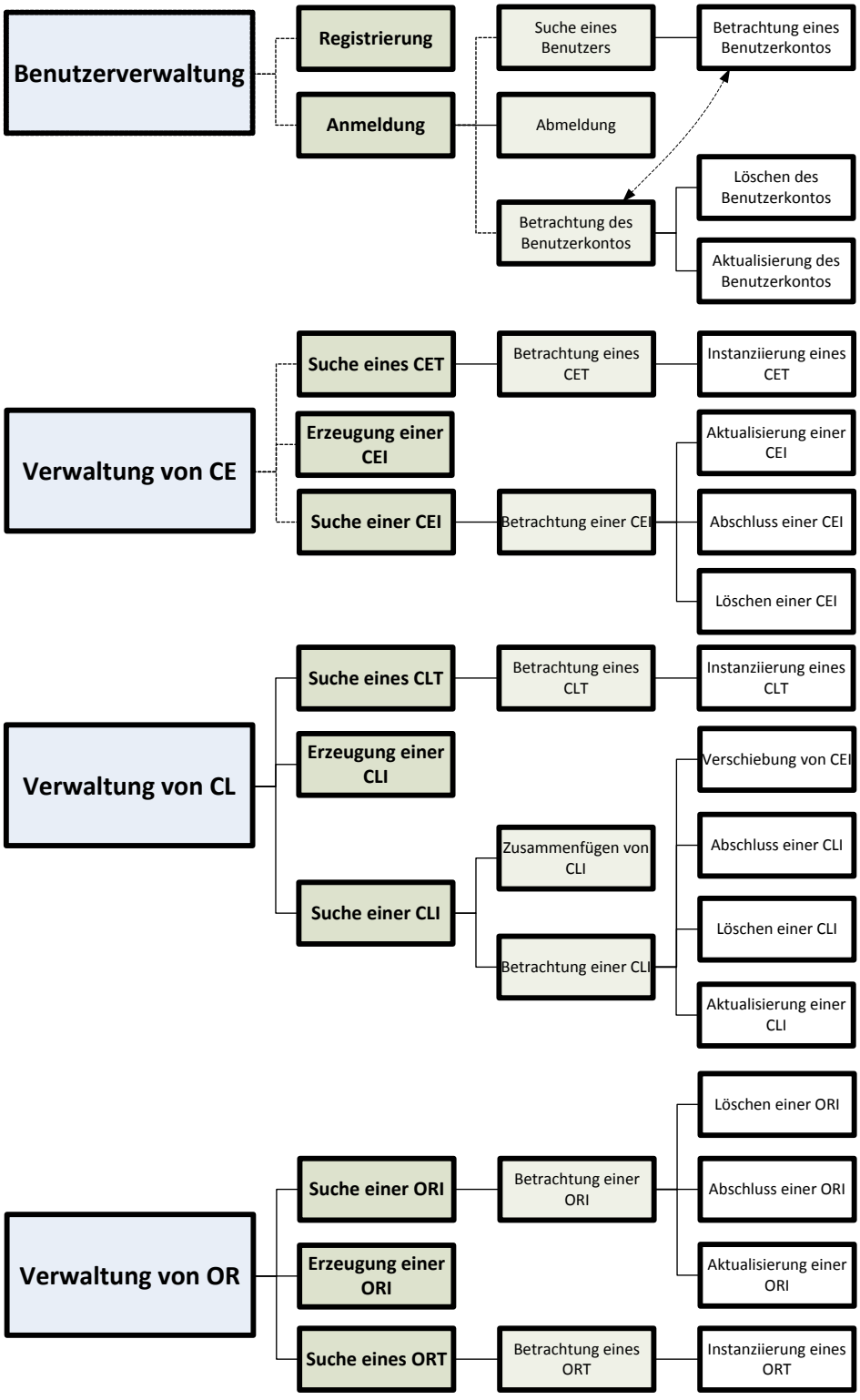


Abbildung 4.5: Die Aufgaben der Anwender des Systems in hierarchischer Darstellung

Übersicht: Eine Ansicht, die der Übersicht von Benutzern, OR, CL oder CE dient und eine Liste mehrerer dieser anzeigt.

Detailansicht: Eine Ansicht, die innerhalb der Übersicht aufgerufen werden kann und die Details eines einzelnen Elements anzeigt.

Bearbeitung: Eine Ansicht innerhalb der Übersicht, bei der die Elemente der Detailansicht bearbeitet werden können.

Suche: Eine Ansicht innerhalb der Übersicht, in der Suchfilter angewendet und Ergebnisse angezeigt werden können.

Erstellung: Eine Ansicht innerhalb der Übersicht, in der neue Instanzen (ORIs, CLIs, CEIs) ohne entsprechende Typzuordnung (ORTs, CLTs, CETs) erstellt werden können.

4.3.3 Hauptnavigationspfade

Die zuvor definierten Typen von Ansichten können nun mit Hilfe der Aufgabenanalyse und der Aufgabenhierarchie in Form einer Dialogstruktur (siehe Abbildung 4.6) dargestellt werden, welche die Navigationspfade zwischen den Ansichten beschreibt. In eine weitere Aufteilung in Instanzen und Typen wurde hierbei aus Gründen der Übersicht verzichtet.

Auffallend ist dabei die Redundanz der Funktionalitäten. Sowohl OR und CL, als auch CE und Benutzer können angesehen, bearbeitet und gesucht werden. OR, CL und CE können zusätzlich erstellt werden. Sie werden jeweils auch zur Anzeige in einer Übersicht benötigt.
Um Konsistenz durch die Anwendung hindurch zu wahren, sollte die grafische Repräsentation dieser Funktionalitäten einheitlich aufgebaut sein. Eine mögliche Gruppierung der Ansichten zu Dialogen ist in Abbildung 4.6 farblich dargestellt.

4.3.4 Konzepte und Skizzen

Da die Navigation anhand einer hierarchischen Baumstruktur stattfindet und auf jeder der Ebenen verschiedene Operationen durchgeführt werden können, bietet sich, angelehnt an die Mehrheit der in der Ist-Stand Analyse betrachteten Anwendungen (siehe Kapitel

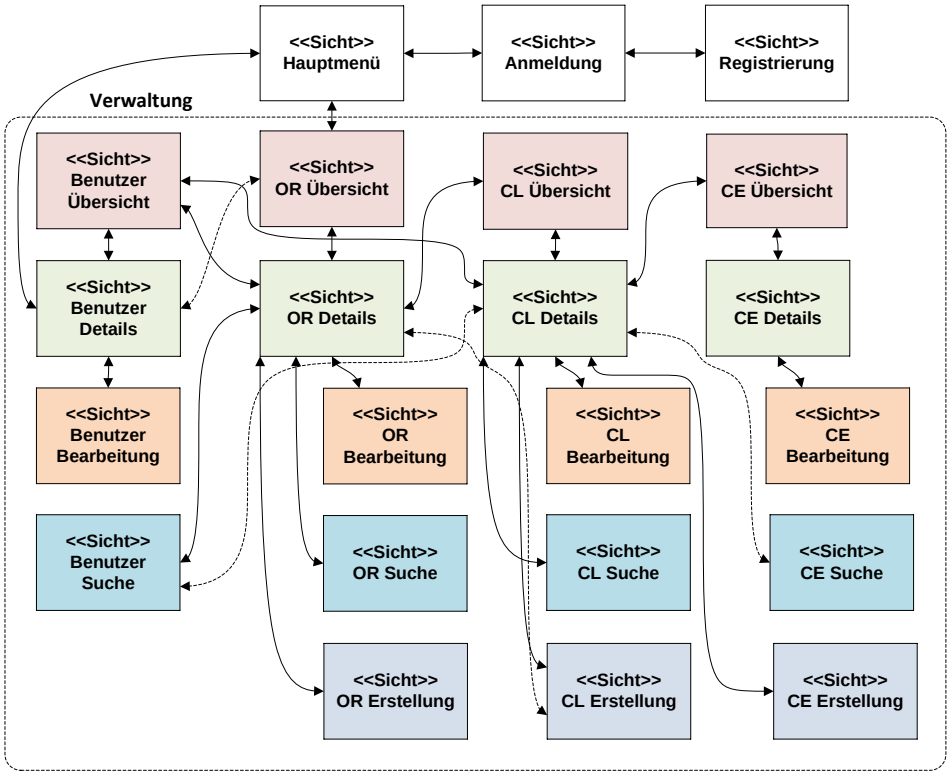


Abbildung 4.6: Die Navigationspfade der Hauptdialoge

3.1), ein zweispaltiges Gesamtlayout an (siehe Abbildung 4.7). In der linken Spalte

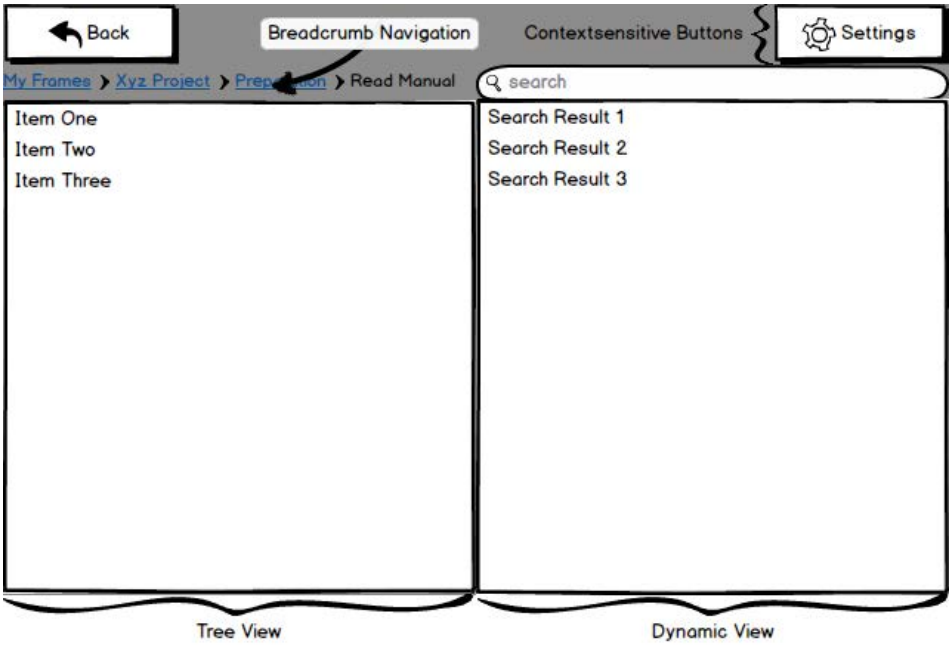


Abbildung 4.7: Mockup des generellen Layouts von pCT

befindet sich der Baum aus OR, CL und CE, also den zuvor beschriebenen Übersichten. Die rechte Spalte stellt eine dynamische Ansicht dar und enthält die Ansichten zur Suche, den Details und Bearbeitungsmöglichkeiten.
Die Navigation in Richtung Blattebene (CE) in der Baumstruktur erfolgt per Berührung eines Listenelements. Zu höheren Ebenen wechselt man über eine Breadcrumb[1] Menü oberhalb der Ansichten.
In der Kopfzeile werden Schaltflächen für die Navigation innerhalb der Anwendung, sowie allgemeine und kontextabhängige Funktionalitäten bereitgestellt.

4.4 Mockups der Benutzerschnittstelle

Auf Basis des konzeptionellen Modells der Benutzerschnittstelle (siehe Kapitel 4.3) werden im Folgenden Mockups der tatsächlichen Dialoge konzipiert.

[1] Brotkrümel: Eine Reihung von Links, welche die Navigationsreihenfolge in der Tiefe widerspiegelt.

4.4.1 Mockups Anmeldung und Registrierung

Der Anmeldedialog stellt die initiale Benutzeroberfläche nach dem Start der Anwendung dar (siehe Abbildung 4.8). Auf den ersten Blick sollten hier die gewünschten Aktionen sichtbar sein. Diese sind nach dem zuvor vorgestellten zweispaltigen Layout aufgebaut. Auf der linken Seite ist somit der Wechsel in den Registrierungsdialog und auf der rechten Seite die eigentliche Anmeldung platziert.
Der Anmelde- und Registrierungsdialog benötigen dabei ähnliche Elemente in der Darstellung (siehe Abbildung 4.9). Zum einen müssen Formulardaten eingegeben werden, zum anderen werden Schaltflächen für die Durchführung der Aktion der Anmeldung und Registrierung gebraucht.

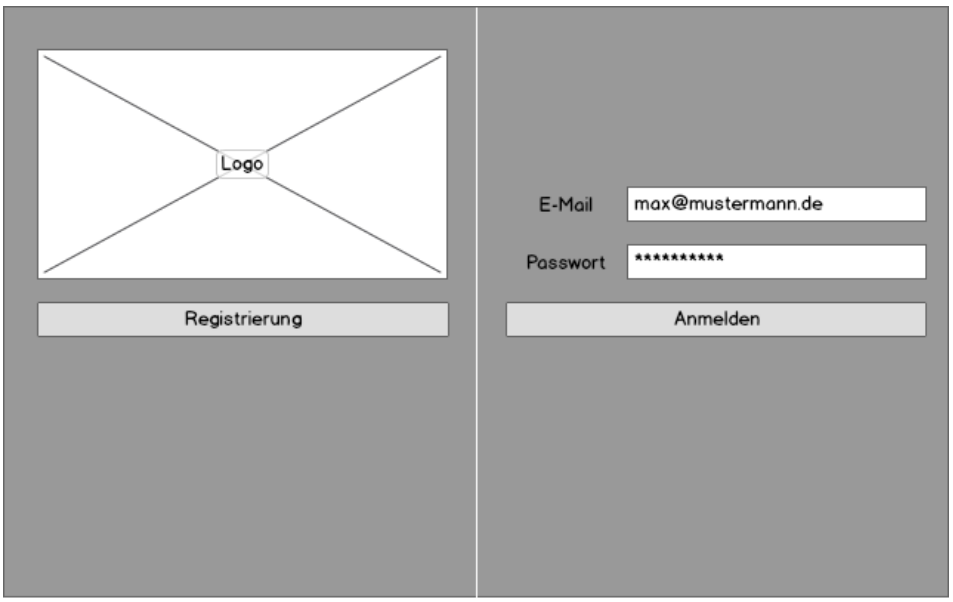


Abbildung 4.8: Mockup des Anmeldedialogs

4.4.2 Mockup Hauptmenü

Das Hauptmenü (siehe Abbildung 4.10) dient dem Einstieg in die Checklistenverwaltung. Es ist direkt nach dem Anmelden zu sehen und bietet direkt Zugriff auf alle nötigen Funktionalitäten. Dazu zählen Suchfunktionen für ORT, ORI, CLT, CLI, CET, CEI und

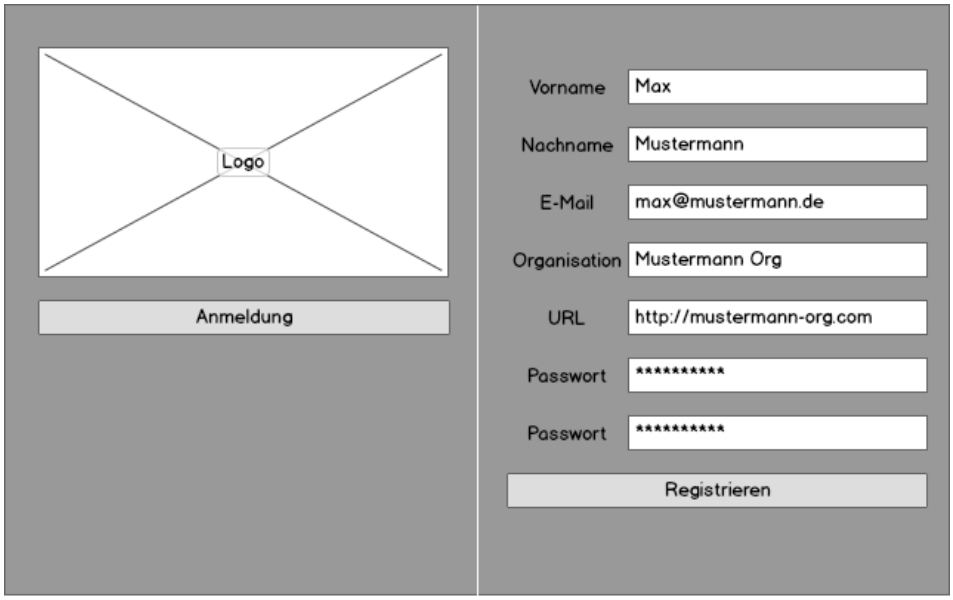


Abbildung 4.9: Mockup des Registrierungsdialogs

Benutzer. Auch der direkte Zugriff auf die Erstellung von neuen Instanzen der Objekte und die Anzeige der ORIs, in die der aktuelle Benutzer involviert ist, gehören dazu. Der Nutzer hat außerdem die Möglichkeit auf seine Benutzerkontodaten zuzugreifen.

4.4.3 Mockup Verwaltungsansicht

Die Verwaltungsansicht stellt jene Ansicht dar, in der sich der Benutzer die meiste Zeit befindet, um seinen Aufgaben nachzugehen (siehe Abbildung 4.11). Sie bietet ihm genau die im konzeptuellen Modell der Benutzerschnittstelle (siehe Kapitel 4.3) vorgestellte Aufteilung an. Auf der linken Seite ist die Navigation durch die Baumstruktur der OR, CL und CE platziert, auf der rechten Seite die dynamische Ansicht, die die Ansichten Details, Bearbeitung, Erstellung und Suche anbietet.
Zwischen den Ansichten in der dynamischen Ansichten kann über die Schaltflächen in der Kopfzeile gewechselt werden. Bei ORIs und CLIs kann die Ansichten der Struktur außerdem umgeschaltet werden, um die beteiligten Benutzer zu verwalten.
Die einzelnen Ansichten in der dynamischen Ansichten werden im Folgenden im Detail vorgestellt.

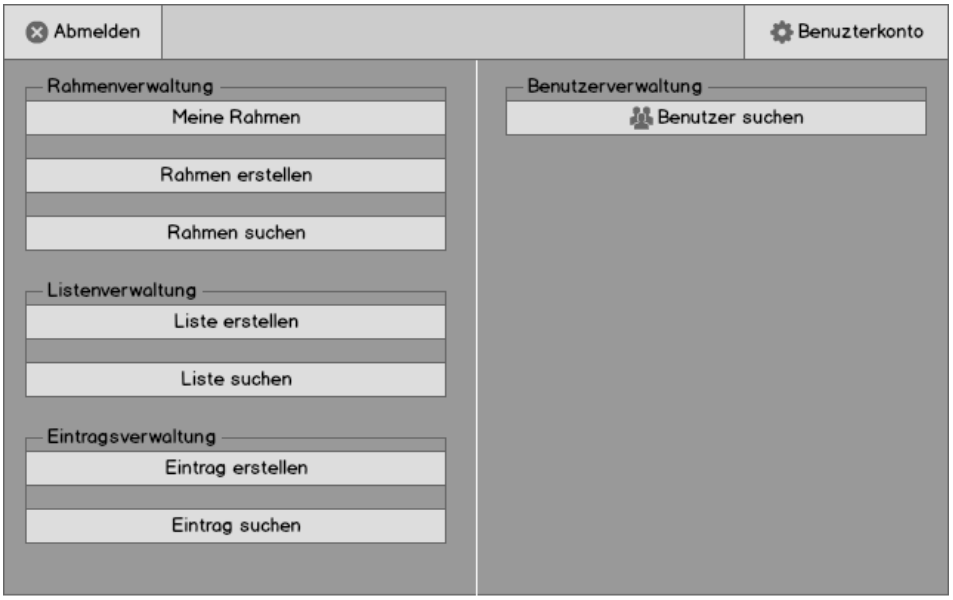


Abbildung 4.10: Mockup des Hauptmenüs mit Zugriff auf alle grundlegenden Funktionalitäten

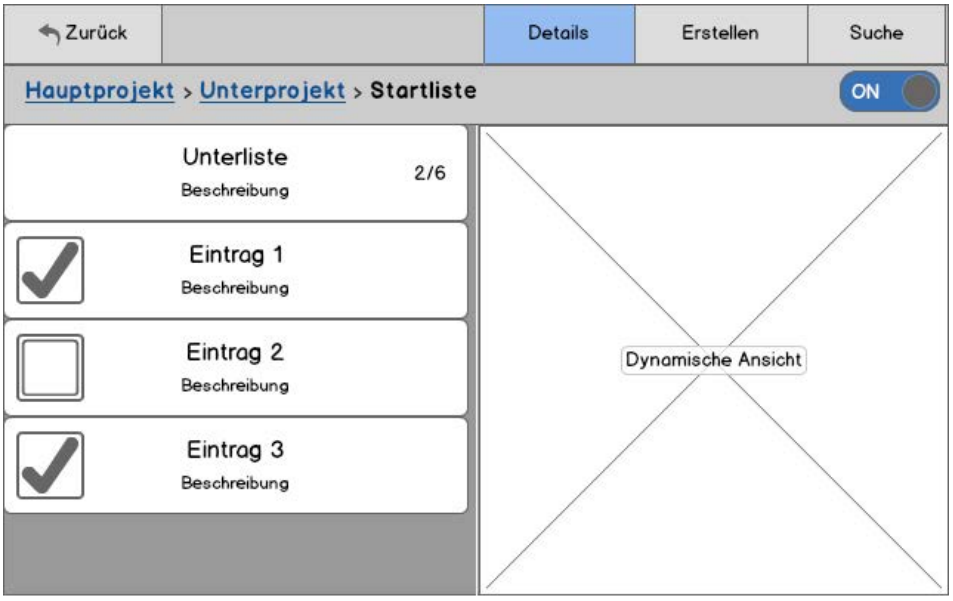


Abbildung 4.11: Mockup der Verwaltungsansicht

Detailansicht

Die Detailansicht zeigt die Details von Benutzern, ORTs, ORIs, CLTs, CLIs, CETs oder CEIs an und beinhaltet zusätzlich die Funktionalität, in einen Editiermodus zur Bearbeitung dieser umzuschalten. Sie ist formularähnlich aufgebaut, besitzt demnach für jedes Attribut eines Datenobjekts eine Bezeichnung und den jeweiligen Wert selbst.
Die Editierfunktion kann beim jeweiligen Datenobjekt auf der rechten Seite der Breadcrumb-Navigation aktiviert werden. Diese und die Schaltflächen zum Übernehmen der Änderungen sind jedoch nur von berechtigten Benutzerrollen zu sehen.
Ein Beispiel einer Detailansicht ist in Abbildung 4.12 zu sehen.

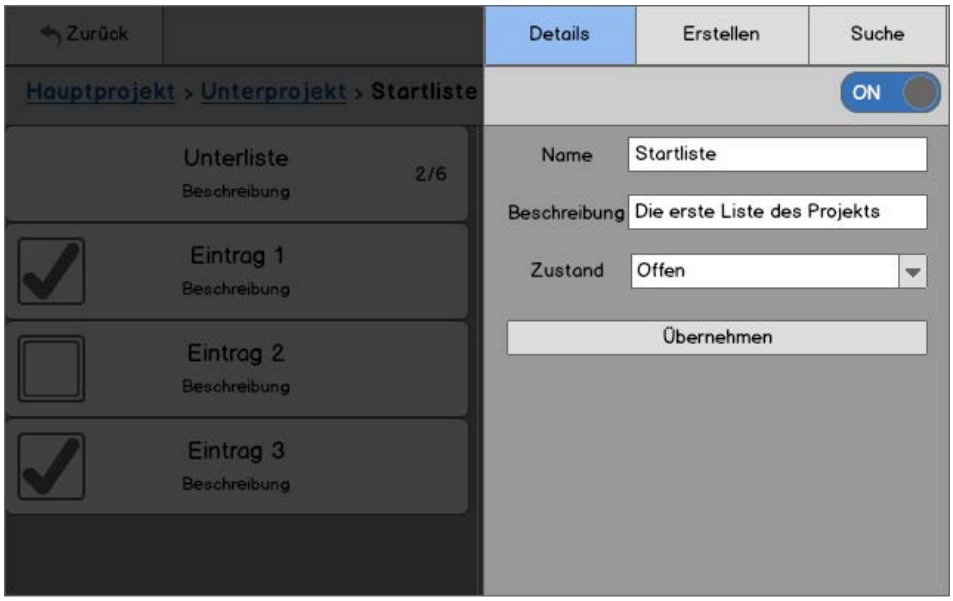


Abbildung 4.12: Mockup der Detailansicht

Erstellansicht

In der Erstellansicht können ORI, CLI und CEI erstellt werden, die nicht von einem entsprechenden Typen abgeleitet werden. Hierbei werden für jede Instanz die nötigen Felder in Form eines Formulars angeboten.
Ein Beispiel einer Detailansicht ist in Abbildung 4.13 zu sehen.

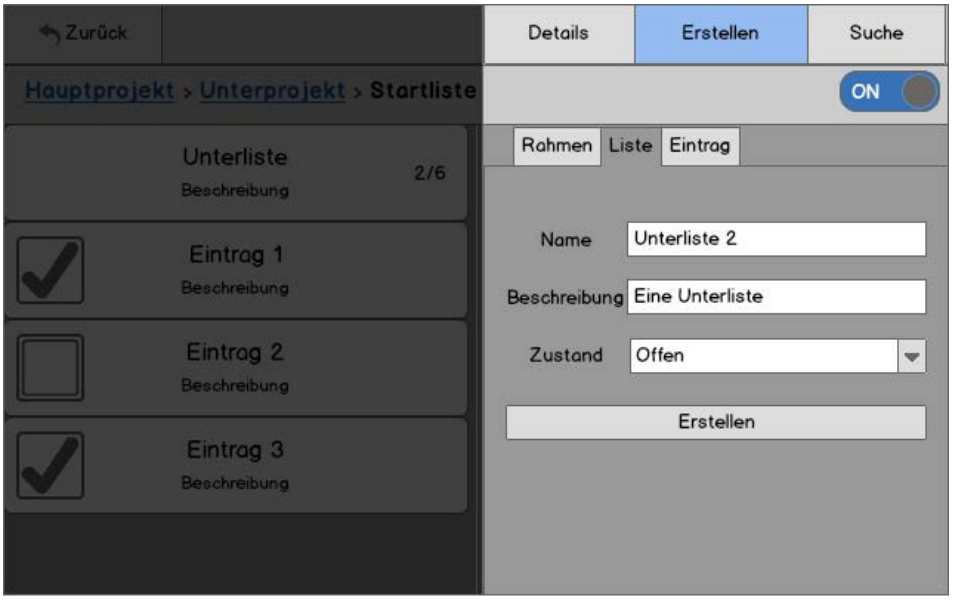


Abbildung 4.13: Mockup der Erstellungsansicht

Suchansicht

Die Suchansicht bietet, je nach Berechtigung des aktuellen Benutzers, die Möglichkeit nach vorhandenen ORTs, ORIs, CLTs, CLIs, CETs, CEIs, und Benutzern im System zu suchen.
Die Ergebnisse der Suchanfrage werden in einer Liste unterhalb dargestellt.
Eine beispielhafte Suche ist in Abbildung 4.14 zu sehen.

4.4.4 Mockup Benutzerkonto

Die Ansicht für das Benutzerkonto muss die Attribute des Benutzers und die organisatorischen Rahmen und Listen desselben anzeigen. Für den aktuell angemeldeten Benutzer muss es außerdem die Möglichkeit bieten, das Konto zu bearbeiten und zu löschen.
Dazu lehnt sich die Ansicht des Benutzerkontos wieder stark an das oben vorgestellte Layout an. In der linken Spalte befinden sich die ORIs und CLIs, denen der Benutzer zugehörig ist, in der rechten Spalte die Details mit den Benutzerattributen und der Mög-

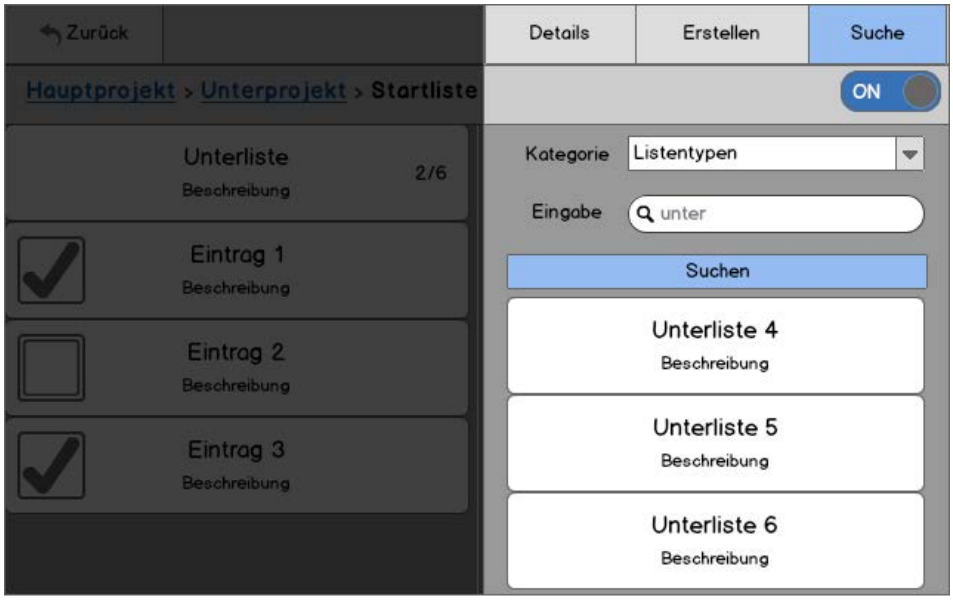


Abbildung 4.14: Mockup der Suchansicht

lichkeit diese zu bearbeiten.
Das Mockup für das Benutzerkonto ist in Abbildung 4.15 dargestellt.

4.4.5 Interaktionsmockups

Um eine benutzerfreundliche Bedienung umzusetzen, die es eben nicht erfordert, sich durch viele Untermenüs zu klicken, wird bei den Interaktionen auf *Drag & Drop* Funktionalität und *Wischgesten* gesetzt.
Drag & Drop unterstützt dabei in ersten Linie die Verwaltung von Benutzern, OR, CL und CE. Dabei werden Elemente zwischen zwischen Übersichts- und Suchergebnislisten verschoben oder innerhalb einer sortiert.
Zum Beispiel kann zur Instanziierung eines CLT in einer ORI einfach der CLT an die gewünschte Position in einer ORI gezogen werden. So muss sich der Benutzer nicht mit dem eigentlichen Konzept der Instantiierung auseinandersetzen, sondern kann intuitiv vorgehen. Das gleiche Konzept gilt auch für die anderen Datentypen. Die Instanziierung eines CLT als untegeordnete CLI in eine bestehende CLI ist in Abbildung 4.16 dargestellt.

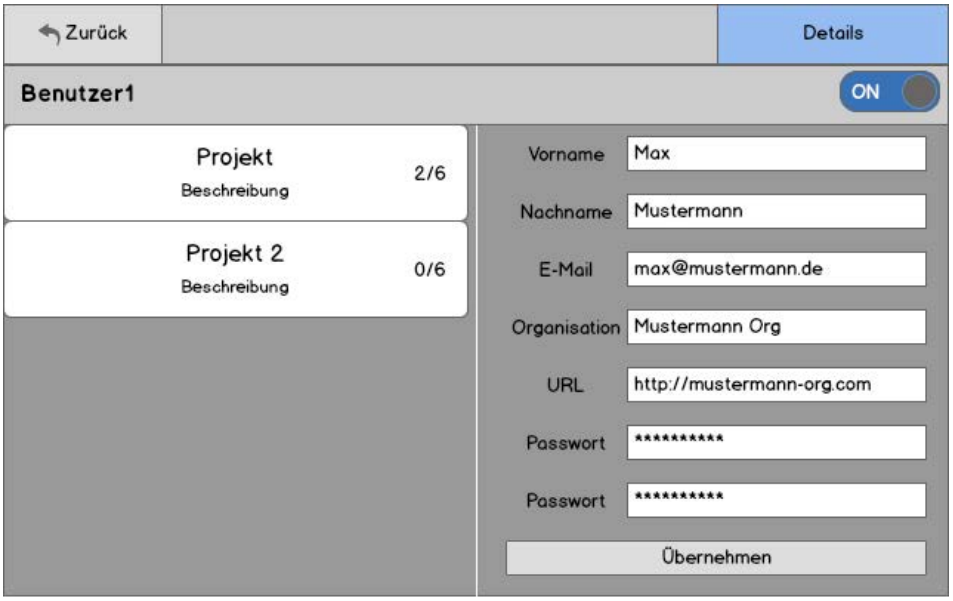


Abbildung 4.15: Mockup der Ansicht des Benutzerkontos

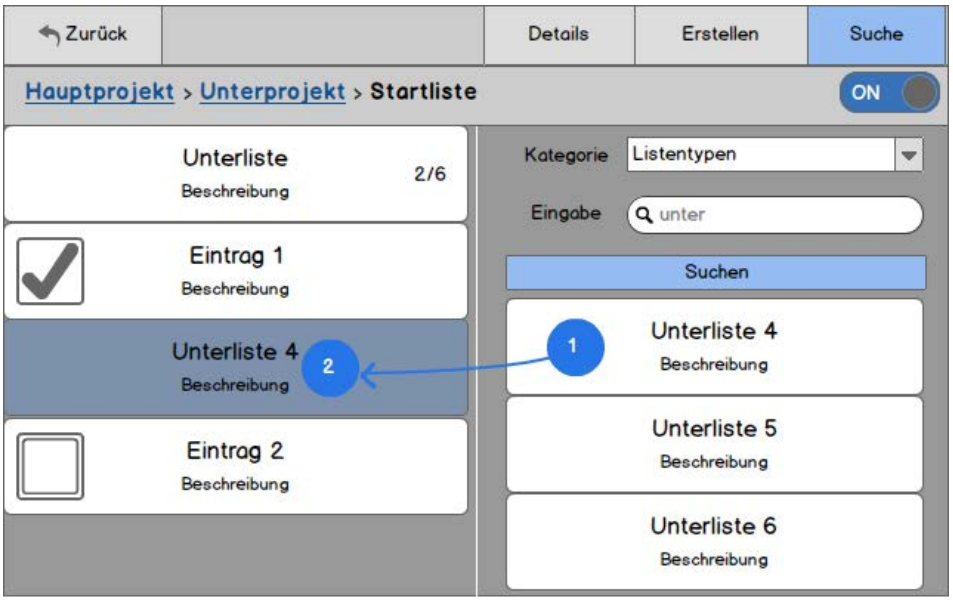


Abbildung 4.16: Mockup der Instanziierung eines CLT in einer bestehenden CLI

4.5 Styleguide

Der Styleguide dient der Definition von Gestaltungsregel, die beim Entwurf der Anwendung eingehalten werden sollen. Hiermit soll dafür gesorgt werden, dass alle Dialoge einheitlich und konsistent aufgebaut sind [Off13].
Da der Fokus auf den Betriebssystemen Apple iOS und Google Android liegt, sollen auch die Styleguides dieser Plattformen nicht außer Acht gelassen werden. Bei Apple sind dies die iOS Guidelines [App13b], bei Google das Android Design [Goo13].

4.5.1 Texte und Schreibstil

Da der Platz auf einem mobilen Gerät begrenzt ist und der Anwender nicht mit langen Texten von seiner eigentlichen Arbeit abgelenkt werden soll, sollen Bezeichnungen von Feldern und Schaltflächen möglichst kurz und prägnant gestaltet werden.
Im Android Design sind hierzu folgende Ratschläge für einen Text aufgeführt [Goo13]:

- Halte ihn kurz.
- Halte ihn einfach.
- Sei freundlich.
- Fange mit dem Wichtigsten zuerst an.
- Beschreibe nur was nötig ist, und nicht mehr.
- Vermeide Wiederholung.

Schaltflächen, die eine Aktion auslösen, werden demnach mit dem entsprechenden Verb versehen, die die Aktion beschreibt. Beispiele dafür sind „Registrieren“ und „Instanziieren“. Andere Schaltflächen, die dem Wechsel zwischen Ansichten dienen, werden mit einem Substantiv oder substantiviertem Verb versehen, welches den Namen der Ansicht repräsentiert. Konkret wären das zum Beispiel „Registrierung“ oder „Suche“.

4.5.2 Farbe

Farbe hat einen Wiedererkennungswert und kann als Informationsträger dienen. Hierbei sollte jedoch beachtet werden, dass nicht mehr als fünf bis sieben Farben verwendet

werden. Der Grund dafür ist, dass die Augen bei jeder Farbe neu fokussieren müssen um sie scharf zu sehen, und durch einen zu starken Einsatz von Farbe können die Augen so schnell ermüden [Off13]. Farbe soll stattdessen primär der Hervorhebung dienen. Mit ihr kann der Benutzer auf für ihn aktuell wichtige Bereiche der Anwendung aufmerksam gemacht werden.
Jedoch muss auch bei der Wahl der Farben einiges beachtet werden. Zum einen die mögliche Farbblindheit von Anwendern, die Grün- und Rottöne schwer unterscheidbar macht, zum anderen können zu hohe Kontraste zu Flimmern und Nachbildern führen. Außerdem sollte blau nur für große Flächen und Hintergründe und nicht für dünne Linien und Schrift verwendet werden, da das menschliche Auge weniger Rezeptoren für diese Farbe hat als für rot und grün, und die blauen Strukturen so etwas schwerer wahrzunehmen sind [Off13].
Als Grundlage für die Farbwahl der Anwendung dient das bereits bestehende Logo des Projekts proCollab (siehe Abbildung 4.17). Als Hintergrundfarbe bietet sich ein heller

Abbildung 4.17: Das Logo des Projekts proCollab

Grauton an. Dieser erlaubt es eine große Farbpalette in Kombination zu verwenden, führt zu keinen starken Spiegelungen in hellen Räumen, wie es zum Beispiel schwarz machen würde, und bietet noch den nötigen Kontrast auch hellere Farben zu verwenden. Hierbei wird die in Abbildung 4.18 dargestellte papierähnliche Struktur verwendet, die einen mittleren Grauwert von 98% weiß (Hex: #F9F9F9) besitzt. Sie dient dem Auge als

Abbildung 4.18: Papierstruktur des Hintergrunds (Kontrast für bessere Darstellung erhöht)

möglicher Fixpunkt, sodass es beim Blick auf größere Hintergrundflächen nicht durch ständige unbewusste Bewegung ermüdet. In Anlehnung an das Logo und die oben aufgeführten Farbregeln, werden größere Flächen, wie die Kopfzeile und Schaltflächen, in einem hellen Blauton (Hex: #1397DF) gestaltet. Die Schrift auf diesen Flächen ist weiß, um eine gute Lesbarkeit und einen möglichst hohen Kontrast zu erzeugen.
Als Farbe zur Hervorhebung von Elementen oder um den Fokus einer Schaltfläche darzustellen wird ein analoger grüner Farbton (Hex: #0EA652) mit ähnlicher Helligkeit wie der blaue verwendet. Als Farbpalette sind die gewählten Farben auf Abbildung 4.19 dargestellt.

Abbildung 4.19: Farbpalette der Farben für pCT

5

Implementierung

Dieses Kapitel führt in die Technologien ein, die verwendet wurden um die Anwendung zu realisieren und beschreibt, warum diese jeweils gewählt wurden. Außerdem werden Aspekte aus der realisierten Implementierung aufgegriffen und erläutert. Als grafische Übersicht und im Zusammenhanf zu den anderen Kapiteln ist Kapitel 5 in Abbildung 5.1 dargestellt.

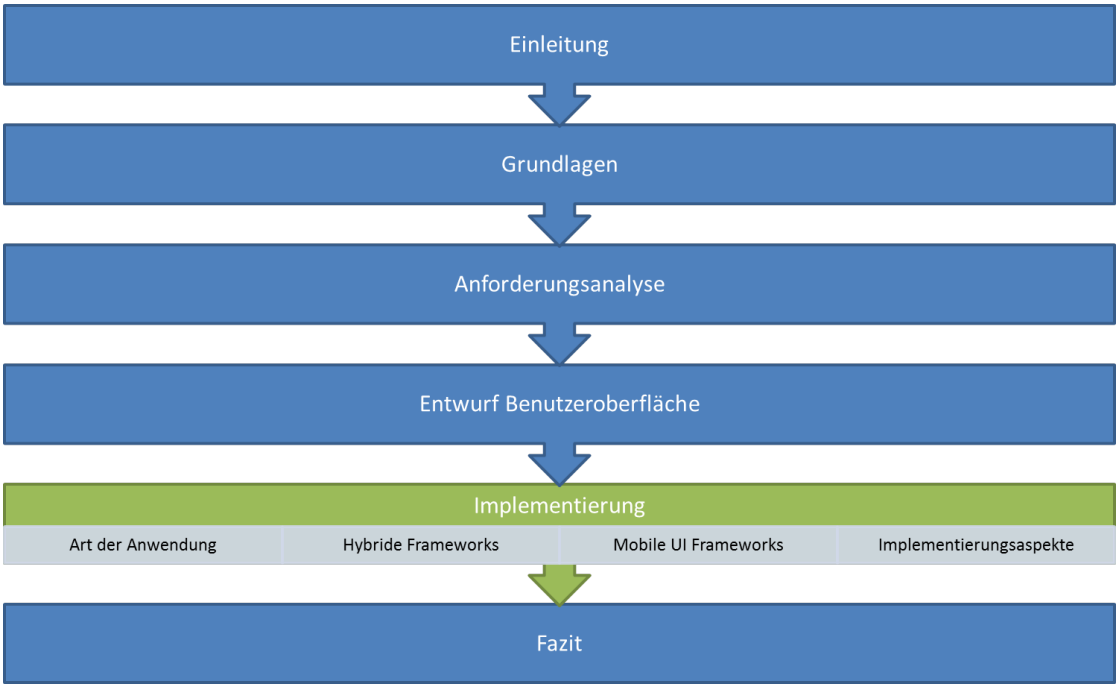


Abbildung 5.1: Aufbau Kapitel 5 — Implementierung

5.1 Art der Anwendung

Im Allgemeinen gibt es eine Vielzahl an Möglichkeiten eine Anwendung für ein mobiles Gerät umzusetzen. Die Möglichkeiten werden im Folgenden erläutert.

5.1.1 Native Anwendung

Eine native Anwendung bezeichnet eine Anwendung, die für eine spezifische Plattform kompiliert[1] wurde und so eigenständig und nicht im Browser des mobilen Geräts läuft [HS10]. So werden zum Beispiel Anwendungen für Android in der Programmiersprache Java und Anwendungen für iOS in der Programmiersprache Objective-C geschrieben. Hierbei kann über API[2]-Aufrufe des Betriebssystems direkt auf Ressourcen und Sensorik des mobilen Geräts zurückgegriffen werden (siehe Abbildung 5.2). Mit einer nativen

[1] in Maschinensprache übersetzen
[2] Application Programming Interface

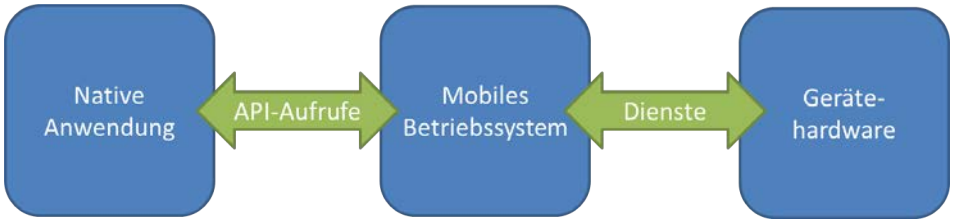


Abbildung 5.2: Interaktion der nativen Anwendung mit dem mobilen Gerät

Anwendung können Rechenoperationen schneller ausgeführt und die Möglichkeiten der Hardware besser ausgenutzt werden [HS10]. Ein weiterer Vorteil ist, dass native Anwendungen in die App-Stores der Plattformanbieter eingestellt werden können und Besitzer von Tablets und Smartphones auf diesem Weg einen einfachen Zugang zu diesen bekommen.
Der Nachteil von nativen Anwendungen besteht im Aufwand der Entwicklung, da für jede Plattform eine individuelle Anwendung in der jeweiligen Programmiersprache entwickelt werden muss [HS10].

5.1.2 Webanwendung

Eine Webanwendung läuft stets in einem Browser. Damit stellt eine Webanwendung nichts anderes dar als eine auf mobile Geräte optimierte Website, die über den Browser aufgerufen wird. Sie ist demnach nicht von der Plattform des Geräts abhängig, und da die Unterstützung von Web-Standards weit fortgeschritten ist, sind Webanwendungen in ihrer Optik wie auch Fiktionalität auf verschiedenen Plattformen gleich nutzbar. Dies führt zu einem großen Vorteil einer Webanwendung: Webseiten sind einfacher und günstiger zu entwickeln als native Anwendungen [HS10].
Eine Webanwendung versucht in der Regel das ’‘Look and Feel“ einer nativen Anwendung nachzubilden. Hierbei kann sie jedoch nur soweit gehen, wie die Browser und auch die Web-Standards Unterstützung anbieten. Ein Zugriff auf die Hardware des Geräts, wie zum Beispiel Kamera oder Sensoren, ist oft noch nicht möglich [HS10].

5.1.3 Hybride Anwendung

Die sogenannte hybride Anwendung versucht die Vorteile der nativen Anwendung mit denen der Webanwendung zu kombinieren. Die Entwicklung hierbei setzt auf Webtechnologien wie HTML5, CSS3 und JavaScript um Funktionalität und Benutzeroberfläche zu entwickeln. Diese ist standardisiert und wird von allen gängigen Browsern aller Betriebssysteme unterstützt. Als Schnittstelle zum mobilen Betriebssystem ist ein Teil der Anwendung in nativem Code geschrieben. Dieser stellt eine Art Browser bereit, in dem die Webtechnologien laufen. Er ist von Betriebssystem zu Betriebssystem unterschiedlich implementiert und bietet zusätzlich die Möglichkeit, auf jene Funktionalitäten zurückzugreifen, die auch von einer nativen Anwendung genutzt werden [HS10]. Dies ist in Abbildung 5.3 veranschaulicht.

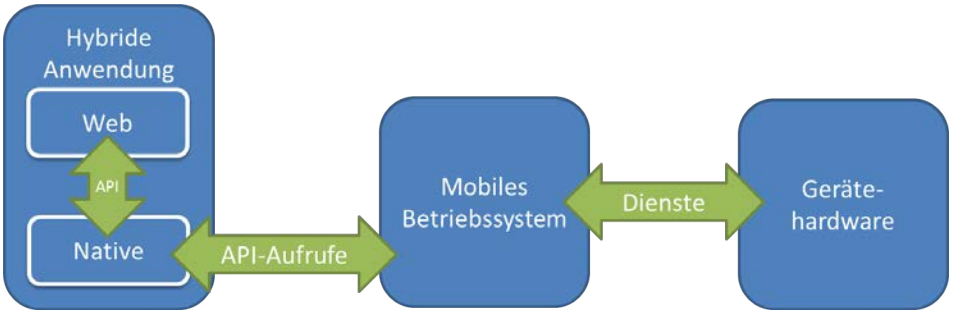


Abbildung 5.3: Interaktion der hybriden Anwendung mit dem mobilen Gerät

Da pCT einen Prototypen darstellt, wird auf die einfach und schnell zu entwickelnde Webtechnologie gesetzt. Um dem Nutzer jedoch ein möglichst natives Erlebnis und einfache Distribution zu ermöglichen, soll die Anwendung nicht rein im Browser laufen. Es bietet sich also eine hybride Anwendung an.

5.2 Framework für hybride Anwendung

Zur effizienten Umsetzung von einer hybriden Anwendung gibt es verschiedene Frameworks. Die zwei populärsten sind Titanium Mobile von Appcelerator [App13a] und PhoneGap von Adobe/Nitobi [AN13], die in den folgenden Abschnitten vorgestellt wer-

den [HS10]. Den beiden Frameworks ist gemeinsam, dass HTML, CSS und JavaScript für die Entwicklung einer mobilen Anwendung verwendet wird. Die Frameworks bieten Zugriffe auf Gerätefunktionen, die nicht über einen Webbrowser erreichbar wären, und durch diese realisierte Anwendungen können über die jeweiligen App- Stores vertrieben werden [HS10].

5.2.1 PhoneGap

PhoneGap verwendet ein natives Browserfenster, in welchem eine reine Webanwendung läuft. Es bietet eine sehr große Unterstützung von mobiler Plattformen. Dazu zählen iOS, Android, Windows Phone, Blackberry, Symbian und Bada [AN13].
Hierbei generiert PhoneGap hauptsächlich nativen Code für die Komponente der Plattform, welche die Webanwendung anzeigt. Bei Android ist dies beispielsweise die *WebView*. Die Anwendung selbst besteht weiterhin aus HTML, CSS und JavaScript und verhält sich wie in einem normalen Webbrowser [HS10].
Die Architektur des Frameworks wird ist in Abbildung 5.4 dargestellt.

5.2.2 Titanium Mobile

Bei Titanium Mobile ist die Anwendung auf dem Gerät keine reine Webanwendung mehr. Stattdessen verwendet Titanium Mobile spezielle JavaScript Klassen, die in den nativen Code der Zielplattform übersetzt werden. Hardwarezugriffe sind wie bei PhoneGap ebenfalls möglich [HS10]. In Abbildung 5.5 ist die Architektur von Titanium Mobile dargestellt.
Unterstützt werden von Titanium die Plattformen iOS und Android. Die Unterstützung weiterer Plattformen wie Windows Phone und Blackberry ist in Entwicklung [App13a].

5.2.3 Vergleich der Frameworks

Der Unterschied der beiden Plattformen kann folgendermaßen ausgedrückt werden: Bei PhoneGap wird eine mobile Webseite entwickelt und anschließend in eine mobile

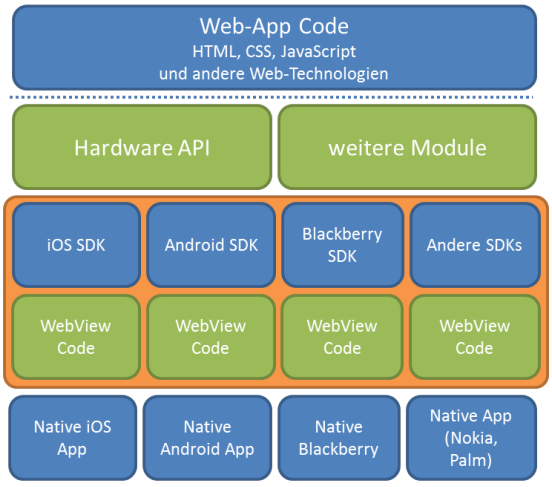


Abbildung 5.4: Darstellung des PhoneGap-Modells nach [HS10]

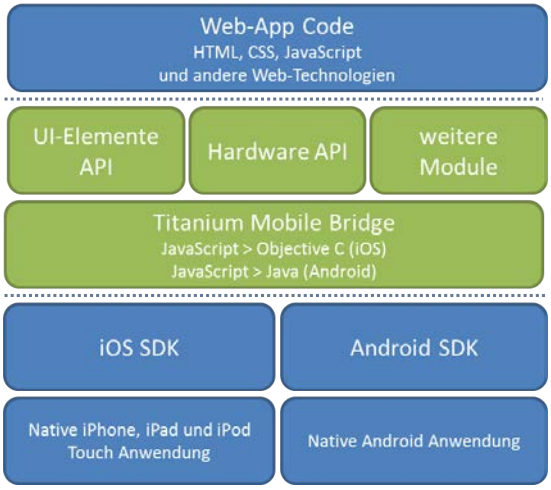


Abbildung 5.5: Darstellung des Titanium-Modells nach [HS10]

Anwendung transformiert. Bei Titanium wird eine mobile Anwendung entwickelt und dabei Webtechnologie verwendet. Die fertige Anwendung kann sich jedoch nicht als Webanwendung nutzen lassen [HS10].
Da es für den zu entwickelnden Prototypen von Vorteil ist, schnelle Änderungen und Debugging durchzuführen, ist in diesem Fall die Entwicklung einer reinen Webanwendung mit einem gewöhnlichen Webbrowser und die anschließende Transformation in eine mobile Anwendung mittels PhoneGap gewählt worden.

5.3 Mobile UI Frameworks

Um die Probleme von Inkompatibilitäten zwischen verschieden Webbrowsern zu umgehen, und auf speziell für den Einsatz auf mobilen Plattformen optimierte Benutzeroberflächen zu setzen, bietet sich die Verwendung von Mobile UI Frameworks an. Diese bieten optimierte und benutzerfreundliche Elemente an, die auf allen gängigen Browsern funktionieren und auf die neusten Webtechnologien setzen. Es kommen hierbei die beiden populärsten Anbieter Sencha Touch [Sen13] und JQuery Mobile [jqu13a] in Frage.

5.3.1 Sencha Touch

Sencha Touch bietet ein eigenes Framework für JavaScript, wobei dieses als High-Level-Syntax verwendet wird, um das Erstellen von komplexen HTML Elementen zu vereinfachen. In diesem JavaScript-zentrierten Ansatz werden zum Beispiel Listenobjekte an Panelobjekte angehängt, statt direkt in HTML bzw. auf dem DOM[3] zu arbeiten.

5.3.2 JQuery Mobile

JQuery Mobile baut auf die sehr weit verbreitete JavaScript Bibliothek JQuery auf. JQuery vereinfacht hierbei das Durchlaufen des DOMs, die Manipulation desselben, die Behandlung von Events, Animationen und die Verwendung von AJAX [jqu13b]. JQuery Mobile erweitert dies mit einer grafischen Oberfläche, die auf dem neusten Standard HTML5 basiert und für den Formfaktor mobiler Geräte optimiert ist.
JQuery Mobile ist HTML-zentriert. Es werden Elemente direkt in HTML geschrieben und so das DOM aufgebaut. Das Framework erweitert diese dann automatisch mit Funktionalität und Styling für die mobile Verwendung. Das ermöglicht es jedem beliebigen Browser diese anzuzeigen.
Aufgrund der großen Reife von JQuery Mobile und der sehr guten Verwendbarkeit von JQuery, wird dieses Framework zur Entwicklung von Funktionalität und Benutzeroberfläche des Prototyps verwendet.

5.4 Implementierunsaspekte

In diesem Abschnitt geht es um die konkrete Implementierung und Umsetzung ausgewählter Konzepte. Zunächst wird auf ein paar grundlegende Webtechnologien eingegangen und erklärt, wozu diese verwendet werden. Anschließend wird die Umsetzung einiger Konzepte aus Kapitel 4 aufgegriffen und geschildert.

[3] Data Object Model

5.4.1 JSON, REST und AJAX

Als Datenformat der Anwendung wird die *JavaScript Object Notation* (JSON) verwendet. Dies ist ein, auch für Menschen, leicht lesbares Format, welches boolsch'e Werte, Zahlen, Zeichenfolgen, Arrays, Objekte und Key-Value-Paare unterstützt. Alle Daten, die an den Server geschickt oder von diesem erhalten werden, sind in dieser Notation verfasst. JSON hat im Vergleich zu dem anderen populären Datenformat *XML*[4] auch weniger Overhead, der durch die Notation der Daten erzeugt wird. Dies ermöglicht kleinere Datenpakete, was bei einer schlechten Verbindungsqualität eines mobilen Geräts von Vorteil ist.
Ein Beispiel einer JSON-Datei, wie sie in pCT eingesetzt wird, ist in Listing 5.1 dargestellt. Es zeigt ein Personenobjekt, das als Antwort vom Server auf eine erfolgreiche Anmeldung gesendet wird.

```
1  {
2    "prename" : "Andreas",
3    "surname" : "Koell",
4    "email" : "andreas.koell@uni-ulm.de",
5    "organization":
6    {
7      "name" : "Universitaet Ulm",
8      "url" : "http://uni-ulm.de",
9      "id" : 13
10   },
11   "id" : 4
12 }
```

Listing 5.1: Das Personenobjekt in Form eines JSON-Strings

Für den Austausch von Daten stellt der Server eine REST-Schnittstelle bereit. REST steht für *Representational State Transfer*, wobei das HTTP[5] verwendet wird, um auf verschiedene Ressourcen oder Dienste des Servers zuzugreifen. Diese werden auf

[4] Extensible Markup Language
[5] Hyper Text Transfer Protocol [FGM+99]

verschiedenen Adressen (URLs[6]) bereitgestellt und können über die Operationen GET, POST, PUT und DELETE des HTTP zugegriffen und modifiziert werden.

Damit der Benutzer nach einer Eingabe nicht warten muss, bis die Daten vom Server bereitgestellt wurden, wird das *Asynchronous JavaScript and XML* (AJAX) Konzept verwendet [HS10]. Hierbei findet der Aufruf der Ressource asynchron statt, was dem Benutzer ermöglicht, weiter mit der Anwendung interagieren zu können, auch wenn die Antwort vom Server noch nicht zurückgekommen ist. Jeder AJAX-Anfrage wird hierbei mitgegeben, welche Funktion (Callback) nach erfolgreich erhaltener Antwort aufgerufen werden soll um das Resultat zu verarbeiten.

In Vereinfachte Form ist der Vorgang der Anmeldung mit JSON, AJAX und REST in Abbildung 5.6 dargestellt. Die Daten (hellblau) werden per JavaScript in Form eines JSON-Strings aus dem Anmeldeformular ausgelesen, an die Login-Methode weitergeleitet, welche dann einen AJAX-POST Request absendet. Dabei wird die REST-Schnittstelle für die Anmeldung des Servers verwendet. Die Daten der Antwort von diesem (orange) gehen über eine POST-Response zurück an AJAX, welches sie an die mitgegebene Callback-Methode weiterleitet. Diese ruft in diesem Fall das Hauptmenü auf.

5.4.2 Layout und dynamisches Laden von Inhalten

Da ein spezielles Layout für Tablets erst für eine spätere Version (aktuell 1.4) von JQuery Mobile geplant ist, und es somit keine direkte Unterstützung für Subpages gibt, wurde für pCT ein eigener Ansatz verwendet. Die Verwaltungsseite ist in drei Abschnitte (DIV-Container) gegliedert. Der erste nimmt die ganze Breite der Anwendung ein und ist unter der Kopfzeile platziert. Er dient der Darstellung des Breadcrumbmenüs. Die beiden Abschnitte darunter liegen nebeneinander, nehmen jeweils die halbe Anwendungsbreite ein und sind in sich nochmals in *Tabs* unterteilt. Im linken Abschnitt geschieht die Anzeige der Baumstruktur und die Navigation in derselben (siehe Kapitel 5.4.3). Im rechten Abschnitt werden die Tabs für die kontextabhängige Detailansicht, das Erstellen von Instanzen und die Suche verwendet. Die Anordnung der Abschnitte ist in Abbildung 5.7 zu sehen.

[6] Uniform Resource Locator: Eine Webadresse, zum Beispiel http://domain.com/resource/

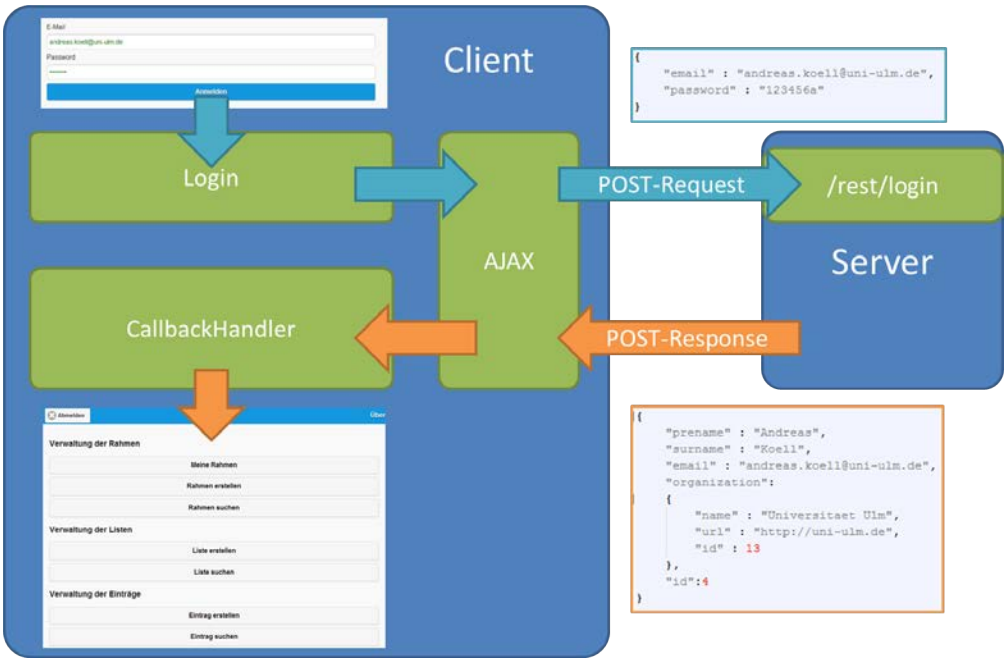


Abbildung 5.6: Modell des Zusammenhangs der Technologien am Beispiel der Anmeldung

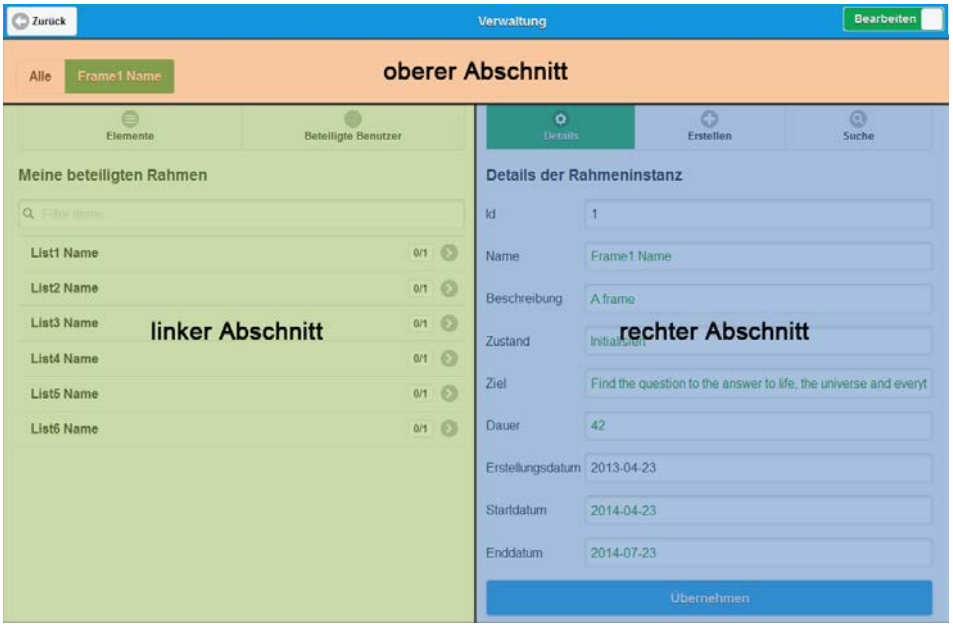


Abbildung 5.7: Das Layout der Verwaltungsansicht

Die Inhalte der Tabs werden dynamisch geladen. Der Grund dafür ist, dass jedes Datenobjekt eine andere Detailansicht benötigt, und auch nicht jede Instanz in jeder übergeordneten Instanz erstellt werden kann. Zum Beispiel kann in einer CLI keine neue ORI erstellt werden. Um dies möglich zu machen, besitzt jedes Datenobjekt ein Klassenattribut, das angibt um was es sich handelt. Aufgrund dieses Attributs wird die passende Ansicht in den jeweiligen Tab geladen. Zwei unterschiedliche Detailansichten sind in Abbildung 5.8 dargestellt. Ein Ausschnitt an JavaScript-Code, der diese Funktionalität implementiert, ist in Listing 5.2 zu sehen.

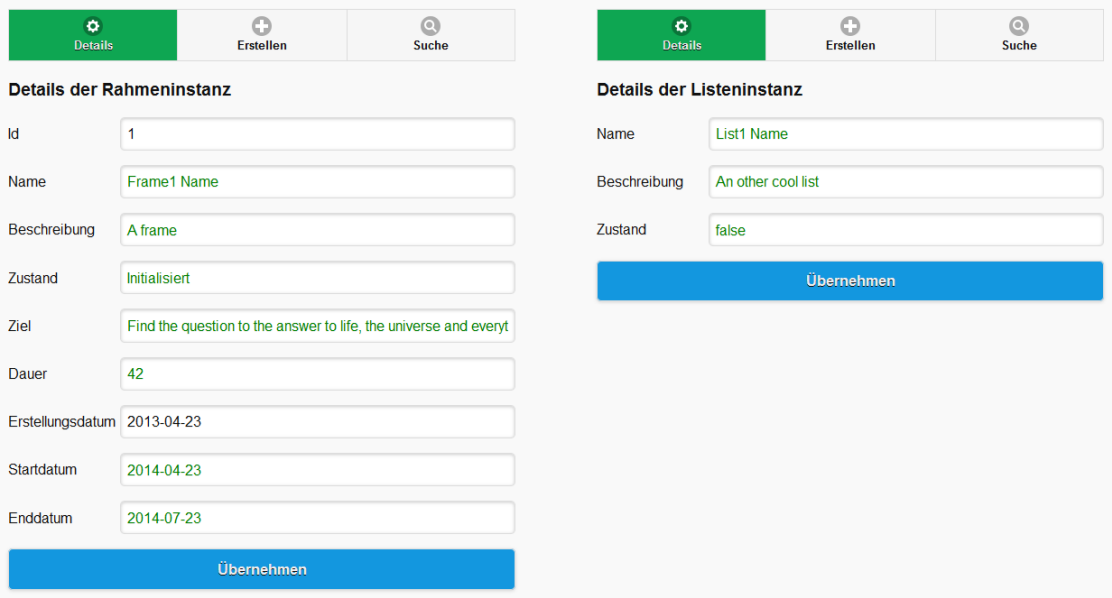


Abbildung 5.8: Unterschiedliche dynamisch geladene Detailansichten von Datenobjekten

```
1 // loads the data-corresponding detail view into the parent
    frame
2 function loadDetailView (jsonData, $frame, requestURL){
3   var $parentElement = $frame.find("#detailTab");
4   if(jsonData.classAttr == 'frameInstace'){
5     requestURL += "#frameInstanceDetails";
6   }
7   else if (jsonData.classAttr=='listType'){
```

```
8       requestURL += "#listTypeDetails";
9     }
10    ..
11    // load the view form the HTML file
12    $parentElement.load(requestURL +' > *', function(response,
          statusText, xhr){
13      // let JQM enhance the raw HTML
14      $parentElement.trigger('create');
15      var form = $parentElement.find('form');
16      if(form[0]) {
17        // if there is a form, fill its content with the jsonData
18        formhandler.setContent(form,jsonData);
19        // if user is in edit mode make all fields editable
20        var enabled = $.mobile.activePage.find(".editToggle")[0].
              value == 'on';
21        formhandler.toggleInputFields(form,enabled);
22      }
23    });
24  }
```

Listing 5.2: Code für das dynamische Laden der Detailansicht

5.4.3 Breadcrumb und Navigation an Baumstrukturen

Wie in Kapitel 4 beschrieben, sind OR, CL und CE in Form einer Baumstruktur organisiert. Die Wurzel ist immer einOR, ein Blatt immer ein einzelner CE.
In der Übersicht aller ORIs, in die der Benutzer involviert ist, können diese Wurzelknoten direkt angezeigt werden (siehe Abbildung 5.9).
Bei der Auswahl einer ORI wird in dessen Detailansicht gewechselt. Außerdem wird ein neues Element im Breadcrumbmenü angehängt (siehe Abbildung 5.10). In diesem sind die Attribute der ORI abgelegt, um eine einfache Navigation zurück zu ermöglichen. Die

Übersichtsliste an ORIs wird dabei im gleichen Schritt mit der Ansicht aller Kindobjekte der ausgewählten ORI überschrieben (siehe Abbildung 5.10). Der Code für diesen Vorgang ist in Listing 5.3 aufgeführt.

```
1  // Clicking a Frame/List/Item/User in a list
2  $(document).on("click",".listElement", function(){
3    //fetch data from element
4    var $jsonData = $(this).data("data");
5
6    // populate list with childen of selected element
7    var $list1 = $.mobile.activePage.find('#elementsTab ul');
8    $list1.data("List").populate($jsonData);
9    loadDynamicContent($jsonData, true); // load detailView etc.
10   addToBreadCrumb($jsonData); // new breadcrumb entry
11
12   // populate involved users list
13   var $list2 = $.mobile.activePage.find('#usersTab ul');
14   $list2.data("List").populate($jsonData.involvedUsers);
15 });
```

Listing 5.3: Code zum Laden der Kindobjekte und Erweitern des Breadcrumbmenüs

Eine weitere Navigation in die Tiefe durch Auswahl einer untergeordneten ORI oder einer CLI führt die gleichen Schritte wie oben aus. Möchte der Benutzer wieder nach oben navigieren, kann er dies einfach tun, indem er auf die Schaltflächen des Breadcrumbmenüs drückt. Die hinterlegten Daten werden dabei wieder angezeigt. Dies ist in Listing 5.4 dargestellt.

```
1  // Clicking a breadcrumb menu element
2  $(document).on("click",".breadCrumbElement",function(){
3    //fetch data from element
4    var $jsonData = $(this).data("data");
5    var $dynamicData = $(this).data("contentData");
```

```
6
7   // populate list with childen of selected element
8   var $list1 = $.mobile.activePage.find('#elementsTab ul');
9   $list1.data("List").populate($jsonData);
10  loadDynamicContent($dynamicData, true);
11
12  // populate involved users list
13  var $list2 = $.mobile.activePage.find('#usersTab ul');
14  $list2.data("List").populate($jsonData.involvedUsers);
15
16  // Update BreadCrumb, remove all subordinate elements
17  $(this).nextAll('a').remove();
18  $(this).addClass("ui-last-child ui-btn-active");
19 });
```

Listing 5.4: Code zum Laden der Daten aus einem Breadcrumb-Element

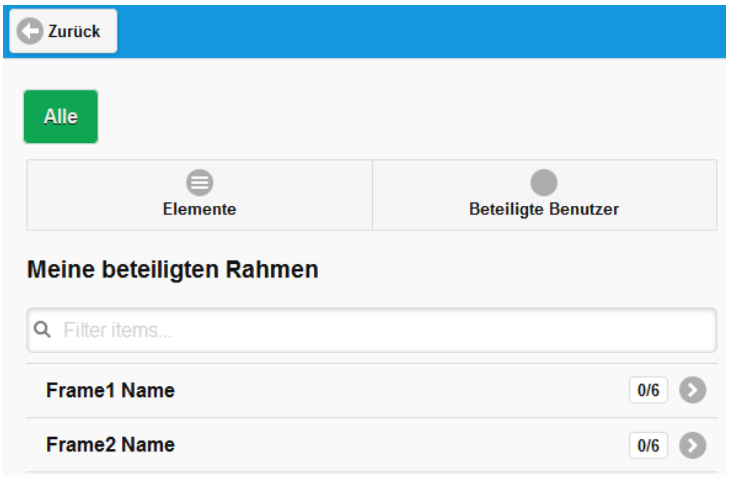


Abbildung 5.9: Darstellung eines ORI als Wurzelknoten

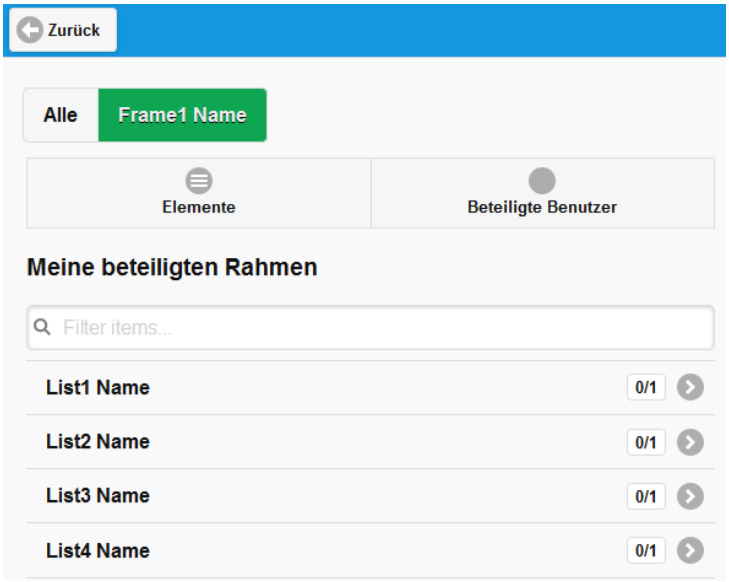


Abbildung 5.10: Erweiterung des Breadcrumbmenüs und Laden der Kindobjekte

5.4.4 Bearbeitung von CLIs

Zur Bearbeitung von CLIs gehört auch die Bearbeitung von CEIs einer CLI. Hierzu muss sich der Anwender im *Bearbeiten*-Modus befinden, welchen er, bei entsprechender Berechtigung, über einem Schalter in der Kopfzeile aktivieren kann. Die CEI können anschließend mittels vertikalem Drag & Drop in ihrer Reihenfolge neu organisiert werden (siehe Abbildung 5.11). Jeder Drag & Drop Vorgang wird dabei folgendermaßen an den Server übermittelt: Der Server muss wissen, welche übergeordnete CLI verändert wird, um welche CEI es sich handelt, und an welche Position diese verschoben werden soll. Der Client übermittelt dieses Tripel an Daten an die passende REST-Schnittstelle des Servers und wartet auf eine Rückmeldung. Ist diese nicht erfolgreich, wird die Operation zurückgesetzt und eine Fehlermeldung angezeigt.

Die kombinierte Bearbeitung zweier CL kommt bei der Instanziierung von CLT vor. Diese werden nach einer Suche in einer Ergebnisliste angezeigt. Der Anwender kann nun per Drag & Drop den CLT von der Ergebnisliste in die Baumstruktur einpflegen. Beim Server wird hier wie oben die passende Schnittstelle zur Instanziierung des CLT aufgerufen. Übermittelt werden die Daten, um welche CLT es sich handelt, in welcher

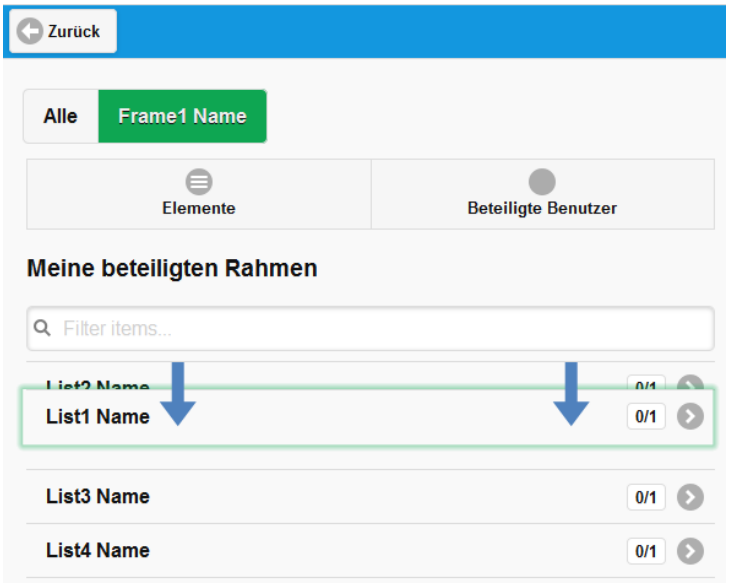


Abbildung 5.11: Sortieren von Listenobjekten mittels Drag & Drop

übergeordneten ORI oder CLI und an welcher Position derselben die CLI erstellt werden soll.

6

Fazit

Kapitel 6 beinhaltet die Zusammenfassung und das Fazit der Arbeit. Es gibt außerdem einen Ausblick auf künftige Anwendung und Erweiterungen von pCT und dem gesamten Protoypen des Projekts proCollab. Die inhaltliche Aufteilung des Kapitels ist in Abbildung 6.1 dargestellt.

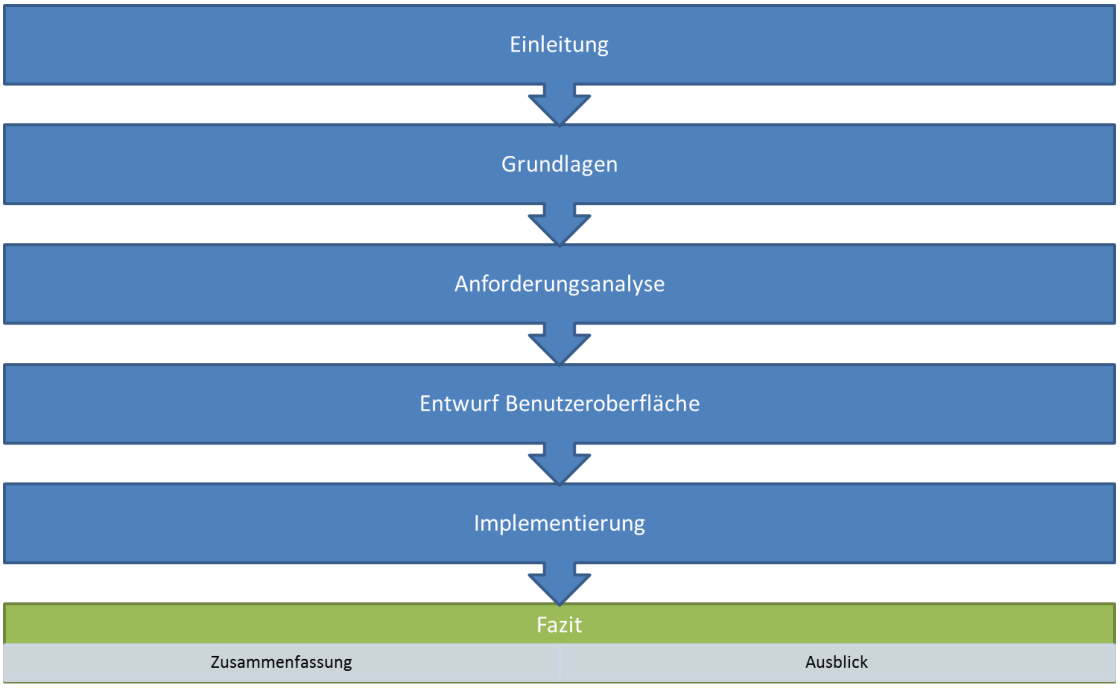


Abbildung 6.1: Aufbau Kapitel 6 — Fazit

6.1 Zusammenfassung

Die Arbeit beginnt mit der Einführung in Wissensarbeit und die damit verbundenen wissensintensiven Prozesse. Sie stellt das Projekt proCollab vor, in dessen Kontext diese Arbeit verfasst wurde und geht anschließend auf Anwendung und Begrifflichkeiten des Checklistenmanagements ein. Auf die Grundlagen aufbauend wurde anschließend eine umfangreiche Anforderungsanalyse für die entwickelte Anwendung durchgeführt. In dieser wurden die Benutzer, Aufgaben und Rahmenbedingungen analysiert, die Einfluss auf Funktionalität, Bedienung und Gestaltung der Anwendung haben. Im Entwurf wurden die Ergebnisse der Anforderungsanalyse aufgegriffen, die Systemarchitektur vorgestellt und die Benutzerschnittstelle in mehreren Schritten aufgebaut. Über die Definition eines Styleguides wurde anschließend ein Einblick in die konkrete Implementierung gegeben. Hierbei wurden Technologien und Frameworks vorgestellt und die Umsetzung einiger Aspekte der Anwendung erläutert.

6.2 Ausblick

Der im Rahmen dieser Arbeit entwickelte Prototyp stellt die Basis für eine künftige Evaluation, um herauszufinden, wie Checklistenmanagement auf einem mobilen Gerät funktioniert. Viele Aspekte konnten für diesen Prototypen noch nicht berücksichtigt werden, um diesen einfach zu halten. Zum Beispiel sollte ein feingranulares Rollenmanagement möglich sein, um OR den speziellen Bedürfnissen von Unternehmen anpassen zu können. Zu der Anpassbarkeit zählt auch ein Templating System, über welches ein Unternehmen das Aussehen und den Umfang an Eigenschaften von ORTs, ORIs, CLTs, CLIs, CETs und CEIs individuell gestalten kann. Hier könnten CEI zum Beispiel auf digitale Handbücher oder Pläne verweisen, die die Aufgabe umfassend beschreiben. Aber auch sogenannte *Constraints*[1] sollten bei einer vollständigen Anwendung unterstützt werden. Ein Constraint stellt zum Beispiel eine Abhängigkeit zwischen CLI oder CEI dar. Hier könnte das Abschließen einer Aufgabe die Handlungen einer anderen CLI nicht mehr nötig oder möglich machen. In einem anderen Fall könnten jedoch neue Aufgaben dynamisch hinzukommen.

Aber auch aus technischer Sicht ist ein Ausbau des Prototypen sinnvoll. So muss der Fall behandelt werden, wenn die Datenverbindung zum Server wegfällt. Hier sollten die Anwendungen die Handlungen des Anwenders in einem Zwischenspeicher halten und bei Wiederherstellung der Verbindung als Paket an den Server schicken. Das ermöglicht dem Anwender auch offline weiterzuarbeiten. Eine solche Funktionalität birgt jedoch viele neue Probleme auf Client- und Serverseite, die es zuvor zu lösen gilt.

Der hiermit erarbeitete Prototyp pCT stellt einen ersten Schritt in Richtung Lösung der Probleme wissensintensiver Prozesse mit Hilfe des Checklistenmanagments dar. Ob und wie gut dieses funktioniert, wird sich in der weiteren Entwicklung des Projekts proCollab zeigen.

[1] engl. für eine Einschränkung oder Abhängigkeit

Abbildungsverzeichnis

Listings

Literaturverzeichnis

[AN13] ADOBE ; NITOBI: *PhoneGap*. http://phonegap.com/. Version: 2013, Abruf: 10.10.2013

[Ani13] ANIMOCA: *Animoca Data: Most Popular Android Tablets Worldwide (Feb 18 – Mar 20, 2013)*. http://www.animoca.com/en/?p=5153. Version: 2013, Abruf: 30.05.2013

[App13a] APPCELERATOR INC.: *Titanium Mobile*. http://www.appcelerator.com/titanium/. Version: 2013, Abruf: 10.10.2013

[App13b] APPLE INC.: *iOS Human Interface Guidelines*. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/index.html. Version: 18.09.2013, Abruf: 09.10.2013

[Bit13] BITSPIN: *Tasks Free*. https://play.google.com/store/apps/details?id=ch.teamtasks.tasks. Version: 18.08.2013, Abruf: 30.09.2013

[FGM+99] FIELDING, R. ; GETTYS, J. ; MOGUL, J. ; FRYSTYK, H. ; MASINTER, L. ; LEACH, P. ; BERNERS-LEE, T.: *Hypertext Transfer Protocol – HTTP/1.1*. http://www.w3.org/Protocols/rfc2616/rfc2616.html. Version: 1999, Abruf: 10.10.2013

[Gaw10] GAWANDE, Atul: *The Checklist Manifesto: How to Get Things Right*. 1. New York : Henry Holt and Company, LLC, 2010 http://books.google.de/books?id=x3IcNujwHxcC. – ISBN 9781429953382

[Gei13] GEIGER, Sabrina: *Konzeption und Entwicklung einer auf Smartphones optimierten mobilen Anwendung für kollaboratibes Checklisten-Management*. Ulm, Universität Ulm, Bachelorarbeit, 2013

[Goo13] GOOGLE: *Android Design*. http://developer.android.com/design/index.html. Version: 2013, Abruf: 09.10.2013

[Han13] HANDY APPS INC.: *Tasks N Todos - To Do List*. https://play.google.com/store/apps/details?id=com.handyapps.tasksntodos. Version: 27.08.2013, Abruf: 30.09.2013

[HS10] HAIGES, Sven ; SPIERING, Markus: *HTML5-Apps für iPhone und Android*. 1. Poing : Franzis Verlag GmbH, 2010 (Franzis Professional Series). http://books.google.de/books?id=pVr4MAEACAAJ. – ISBN 9783645602013

[IAB12] IABG: *V-Modell XT, Version 1.4*. http://v-modell.iabg.de/v-modell-xt-html/index.html. Version: 13.06.2012

[Ing13] INGENIOUS SOFTWARE SOLUTIONS: *Tasks+ To Do List Manager*. https://play.google.com/store/apps/details?id=com.iss.tasksplus. Version: 26.04.2013, Abruf: 30.09.2013

[jqu13a] *JQuery Mobile Website*. http://jquerymobile.com/. Version: 2013, Abruf: 10.10.2013

[jqu13b] *JQuery Website*. http://jquery.com/. Version: 2013, Abruf: 10.10.2013

[LFKJ+10] LESSING, Constanze ; FRANCOIS-KETTNER, Hedwig ; JONITZ, Günther ; BAUER, Hartwig ; SCHRAPPE, Matthias: Checklisten im OP – ein sinnvolles Instrument zur Verbesserung der Patientensicherheit? In: *Perioperative Medizin* 2 (2010), Nr. 4, S. 179–186. http://dx.doi.org/10.1016/j.periop.2010.04.003. – DOI 10.1016/j.periop.2010.04.003. – ISSN 18752772

[LR01] LEFF, A. ; RAYFIELD, J. T.: Web-application development using the Model/View-/Controller design pattern. In: *Fifth IEEE International Enterprise Distributed Object Computing Conference*, 4-7 Sept. 2001, S. 118–127

[mak13] MAKERAMEN: *noodles free - To Do List*. https://play.google.com/store/apps/details?id=com.makeramen.todolist. Version: 02.01.2013, Abruf: 30.09.2013

[MKR13] MUNDBROD, Nicolas ; KOLB, Jens ; REICHERT, Manfred: Towards a System Support for Collaborative Knowledge Work. In: *Business Process Management Workshops* Bd. 132. Heidelberg and Berlin : Springer, 2013, S. 31–42

[MRS12] MAINELLI, Tom ; REITH, Ryan ; SHIRER, Michael: *IDC Worldwide Quartely Tablet Tracker Analysis*. http://www.idc.com/getdoc.jsp?containerId=prUS23833612. Version: 05.12.2012, Abruf: 18.05.2013

[MRS13a] MAINELLI, Tom ; REITH, Ryan ; SHIRER, Michael: *Worldwide Tablet Market Surges Ahead on Strong First Quarter Sales*. http://www.idc.com/getdoc.jsp?containerId=prUS24093213. Version: 01.05.2013, Abruf: 30.05.2013

[MRS13b] MAINELLI, Tom ; REITH, Ryan ; SHIRER, Michael: *IDC Forecasts Worldwide Tablet Shipments to Surpass Portable PC Shipments in 2013*. http://www.idc.com/getdoc.jsp?containerId=prUS24129713. Version: 28.05.2013, Abruf: 30.05.2013

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

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

[Nie93] NIELSEN, Jakob: *Usability engineering*. [Updated ed.]. San Francisco and Calif : Morgan Kaufmann Publishers, 1993. – ISBN 0–12–518406–9

[Noz13] NOZBE.COM: *Nozbe HD for Tablets*. https://play.google.com/store/apps/details?id=com.nozbe.tablet. Version: 29.09.2013, Abruf: 30.09.2013

[Off13] OFFERGELD, Michael: *Vorlesungsskript zu Usability Engineering*. Ulm, Universität Ulm, Vorlesung, 2013

[Pro12] *ProDo im Google Playstore*. https://play.google.com/store/apps/details?id=com.othelle.todopro. Version: 04.11.2012, Abruf: 13.06.2013

[pro13] *proCollab - Process-aware Support for Collaborative Knowledge Workers*. http://www.uni-ulm.de/in/iui-dbis/forschung/projekte/procollab.html. Version: 29.04.2013, Abruf: 25.09.2013

[PS13] PUPPE, Martin ; SCHIDLACK, Michael: *Neuer Rekord bei Smartphones*. http://www.bitkom.org/de/presse/8477_77345.aspx. Version: 09.09.2013, Abruf: 01.10.2013

[Rei13] REICH, Daniel: *Konzeption und Entwicklung eines cloudbasierten Persistenz-Systems für kollaboratives Checklisten-Management*. Ulm, Universität Ulm, Bachelorarbeit, 2013

[Rem13] *Remember the Milk*. https://play.google.com/store/apps/details?id=com.rememberthemilk.MobileRTM. Version: 11.09.2013, Abruf: 16.09.2013

[RW12] REICHERT, Manfred ; WEBER, Barbara: *Enabling flexibility in process-aware information systems: Challenges, methods, technologies*. Berlin and Heidelberg : Springer, 2012. – ISBN 978–3–642–30408–8

[Sen13] SENCHA INC.: *Sencha Touch*. http://www.sencha.com/products/touch. Version: 2013, Abruf: 10.10.2013

[Thi13] THIEL, Norman: *Konzeption und Entwicklung einer Web-Applikation für kollaboratives Checklisten-Management*. Ulm, Universität Ulm, Bachelorarbeit, 2013

[Tod13a] *Todoist: To Do List, Task List*. https://play.google.com/store/apps/details?id=com.todoist. Version: 13.09.2013, Abruf: 16.09.2013

[Tod13b] TODOROO: *Astrid Tasks*. http://astrid.com. Version: 23.04.2013, Abruf: 14.05.2013

[TP13] THYLMANN, Marc ; POLS, Axel: *Tablet Computer drängen in die Berufswelt*. http://www.bitkom.org/de/presse/8477_75913.aspx. Version: 19.04.2013, Abruf: 01.10.2013

[Zie13] ZIEGLER, Jule: *Konzeption und Entwicklung eines cloudbasierten Servers für kollaboratives Checklisten-Management*. Ulm, Universität Ulm, Bachelorarbeit, 2013

Name: Andreas Köll Matrikelnummer: 723642

Erklärung

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

Ulm, den 13.11.2013 .....

Andreas Köll