ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Konzeption und Realisierung fortschrittlicher aufgabenbasierter Koordinationsverfahren in mobilen Anwendungen

Bachelorarbeitan der Universität Ulm

Vorgelegt von:
Matthias Gerber
matthias.gerber@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Nicolas Mundbrod

2015

Fassung 9. Juni 2015

© 2015 Matthias Gerber

This work is licensed under the Creative Commons. Attribution-NonCommercial-ShareAlike 3.0 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/de/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Satz: PDF-LaTeX 2ε

Kurzfassung

Die Globalisierung und der zunehmende Wandel von Produktions- zu Informationsgesellschaften führt dazu, dass Wissensarbeiten in Industriestaaten immer mehr an Bedeutung gewinnen. Dadurch rücken Bereiche die neues Wissen schaffen, wie Entwicklung und Forschung in den Mittelpunkt. Allerdings gibt es in diesem Bereich, im Gegensatz zu anderen, keine etablierte, prozessbasierte Unterstützung, da Wissensarbeiten dynamisch sind und sich damit von Fall zu Fall erheblich unterscheiden können. Diese Dynamik erfordert automatisch ein hohes Maß an Kommunikation und Zusammenarbeit zwischen den Wissensarbeitern. Zur Unterstützung von Wissensarbeitern wurde an der Universität Ulm das Projekt *proCollab* gestartet. Im Rahmen des Projektes wurde ein Prototyp entwickelt, der unter anderem eine mobile Applikation für Smartphones beinhaltet. Das Ziel ist es, Wissensarbeiter mithilfe eines Checklistenansatzes bei der kollaborativen Zusammenarbeit zu unterstützen. Da die Akzeptanz einer neuen Anwendung stark davon abhängt, dass sie dem Nutzer möglichst viel Arbeit abnimmt und keine unnötigen Arbeitsschritte verlangt, wurde die mobile Applikation von *proCollab* bezüglich diesen Aspekten untersucht, verbessert und mit neuen Funktionen versehen.

Inhaltsverzeichnis

1

Einleitung

Die weiter voranschreitende Globalisierung hat dazu geführt, dass Industriestaaten sich mehr von arbeitsintensiven Industrien ab-, und sich stattdessen den wissenschaftsintensiven Bereichen zuwenden [Kow11]. Diese Entwicklung hat dazu geführt, dass der Anteil an Arbeitsplätzen in wissensintensiven Industrien in hoch entwickelten Ländern stetig zugenommen hat [KS09]. Deshalb ist es wichtig, dass qualifizierte Arbeitskräfte - Wissensarbeiter genannt - einen leichten Zugang zu neuem Wissen und unterstützenden Technologien erhalten. Dazu gehört die Unterstützung für eine gute kooperative Zusammenarbeit, die gerade in den wissensintensiven Bereichen wie Forschung und Entwicklung von großer Bedeutung sind [KS09]. Dabei führen etablierte Vorgehensweisen und Unterstützungsformen für Routine-Prozesse zu Problemen, da sie nicht für den erheblichen Kommunikations- und Organisationsaufwand, der innerhalb von wissensintensiven Prozessen notwendig ist, entwickelt wurden. Diese Probleme werden im Folgenden genauer erläutert.

1.1 Problemstellung

Dynamische wissensintensive Prozesse sind schwer handhabbar, da sie von vielen Einflussfaktoren und dem Wissen der involvierten Wissensarbeiter abhängen [Tie10]. Zusätzlich sind typischerweise viele Wissensarbeiter aus unterschiedlichen Fachbereichen involviert und arbeiten kooperativ auf ein gemeinsames Ziel hin. Für eine effektive Zusammenarbeit müssen die Wissensarbeiter über etwaige Änderungen und abgeschlossene Ziele informiert werden. Diese Anforderungen führen zu einem sehr hohen Organisations- und Kommunikationsaufwand, um reibungslose und produktive Prozesse

zu ermöglichen. Für die Organisation und Kommunikation nutzen Wissensarbeiter bis heute Aufgabenlisten, wie z.B. To-do-Listen oder Checklisten. Die fehlende Übersicht bei der traditionell nicht synchronisierten, dezentralen Verwaltung von oft papierbasierten Aufgabenlisten macht die effektive Koordination und Kooperation von Wissensarbeitern noch einmal um einiges komplexer. Konkret führt dieser Umstand dazu, dass Fehler übersehen werden, Arbeit gar redundant verrichtet wird oder Prozesse nicht in dem Maße optimiert werden, wie es eigentlich über mehrere Ausführungen hinweg möglich wäre. Aufgrund der geschilderten Probleme ergibt sich ein großer Bedarf für ein Informationssystem, das komplexe, kollaborative Wissensarbeiten systematisch und nachhaltig unterstützt [MKR13]. Durch ein solches Informationssystem ist es zudem möglich, Arbeitsprozesse, die bisher auf Papier organisiert waren elektronisch zu verwalten. Dadurch können mögliche Medienbrüche vermieden werden.

1.2 Zielsetzung

Im Rahmen des Projekt *proCollab* an der Universität Ulm wurde der gleichnamige Ansatz konzipiert und als Prototyp realisiert. *ProCollab* beruht auf der Idee, die Koordination und Kooperation von Wissensarbeitern durch den Einsatz von kollaborativ nutzbaren Checklisten zu unterstützen [Gei13, Koe13, Rei13, Thi13, Zie13]. Zur Unterstützung der mobilen Zusammenarbeit wurde als Teil des Prototypen eine mobile Applikation (pC-App) für Smartphones entwickelt. Dies geschieht mit Hilfe einer zentralen Speicherung der Daten auf dem Server des Prototypen und der Verwaltung der Aufgaben über verschiedene Clients per Browser oder mobilem Endgerät. Die pC-App dient Wissensarbeitern dazu mobil Checklisten zu verwalten, d.h. insbesondere neue Aufgaben hinzuzufügen und bestehende zu aktualisieren oder zu löschen. Das Ziel dieser Arbeit ist es, die Benutzerfreundlichkeit der pC-App weiter zu verbessern und sie um neue Funktionen zu erweitern. Damit soll eine höhere Akzeptanz beim Endanwender erreicht werden, da unnötiger Arbeitsaufwand und fehlende Funktionen potentielle Nutzer abschrecken könnten. Hierzu werden detailliert Anforderungen für verbesserte Funktionalität erhoben und analysiert. Auf der Basis dieser Erkenntnisse wird dann erläutert wie diese konkret konzipiert und umgesetzt wurden. Zu den umgesetzten Funktionalitäten gehört

beispielsweise, dass die pC-App nun auch ohne Verbindung mit dem Server des *proCollab* Prototypen verwendet werden kann und die Verwaltung von Checklisten erleichtert wurde.

1.3 Struktur der Arbeit

Diese Arbeit besteht insgesamt aus sechs Kapiteln. **Kapitel 1** bietet eine Einführung in die behandelte Thematik. In **Kapitel 2** werden die grundlegenden Begriffe eingeführt, die für das Verständnis der folgenden Kapitel wichtig sind. **Kapitel 3** enthält eine Analyse von vergleichbaren mobilen Applikationen in Bezug auf *proCollab*. Zusätzlich werden die möglichen Verbesserungen von *proCollab* systematisch diskutiert. Diese Verbesserungen werden in **Kapitel 4** aufgegriffen und Lösungsansätze werden diskutiert. In **Kapitel 5** folgt schließlich eine detaillierte Beschreibung der tatsächliche Umsetzung der gewählten Lösungsansätze. **Kapitel 6** bietet abschließend eine Zusammenfassung der erstellten Arbeit, sowie einen Ausblick auf mögliche zukünftige Arbeiten am *proCollab* Prototyp.

2

Grundlagen

2.1 Kollaborative Wissensarbeit

Die Hauptaufgabe von Wissensarbeiten ist das Generieren, Verpacken oder Anwenden von Wissen [DJB96]. Dabei kommt es vor allem auf Vielfalt und Ausnahmen statt auf Routine an. Diese Wissensarbeiten werden von Arbeitern, mit einer hohen Qualifikation und einem hohen Niveau von Fachwissen, bearbeitet. Um die Produktivität von Wissensarbeitern zu erhöhen, ist es notwendig, dass die Kommunikation und Kooperation zwischen ihnen reibungslos funktioniert. Diese Zusammenarbeit wird als kollaborative Wissensarbeit bezeichnet und wird in Definition 2.1 beschrieben [MKR13].

Definition 2.1. ***Kollaborative Wissensarbeit***
Eine kollaborative Wissensarbeit bezeichnet einen Prozess, dessen Ausführung stark von Wissensarbeitern abhängt, die komplexe, von einander Abhängige Aufgaben erledigen. Diese Prozesse haben dabei das Ziel Wissen anzuwenden und zu generieren. Dies erfordert ein hohes Maß an Flexibilität während ihrer Planung und Ausführung.

Die Eigenschaften der kollaborativen Wissensarbeit sind wie folgt definiert [MR14]:

Nicht vorhersehbar: Kollaborative Wissensarbeiten beinhalten komplexe und dynamische wissensintensive Prozesse, die von verschiedenen, sich ständig verändernden Einflussfaktoren abhängig sind. Dadurch sind involvierte Wissensarbeiter gezwungen ständig mit unbekannten Einflüssen zu arbeiten, was es unmöglich macht, die wissensintensiven Prozesse mit allen Einzelheiten im Voraus abzusehen.

Zielorientiert: Da wissensintensive Prozesse nicht vorhersehbar sind, wird ein gemeinsames Ziel dazu genutzt, eine Grundlage für die Zusammenarbeit der Wissensarbeiter zu schaffen. Dadurch können die anstehenden Aufgaben und die zur Verfügung stehenden Ressourcen effektiv und zielorientiert angepasst werden. Um das Erreichen eines Ziels zu vereinfachen kann dieses in Teilziele aufgeteilt werden.

Entstehend: Da die wissensintensiven Prozesse nicht vorhergesehen werden können, müssen die Wissensarbeiter regelmäßig prüfen, welche Aufgaben bereits erledigt wurden. Mit diesem Wissen können sie dann das weitere Vorgehen bestimmen. Dadurch besteht kollaborative Wissensarbeit aus abwechselnden Phasen von Planung und Umsetzung, die von den Wissensarbeitern autonom bestimmt werden, um schrittweise das gemeinsame Ziel der Wissensarbeit zu erreichen.

Wissen schaffend: Wissensarbeiter, die an einer kollaborativen Wissensarbeit beteiligt sind, nutzen ihr Fachwissen und ihre Erfahrung um das gesetzte Ziel zu erreichen. Durch das Erreichen von Teilzielen und Zielen, entsteht weiteres Wissen. Durch die ständige Planung und Umsetzung erweitern Wissensarbeiter ebenfalls ihr impliziertes Wissen während der Laufzeit der Prozesse.

Der Lebenszyklus einer solchen Wissensarbeit ist in Abbildung 2.3 beispielhaft dargestellt [MKR13].

2.2 Fallbeispiel

Ein Anwendungsbeispiel für kollaborative Wissensarbeiten wird vorgestellt um das Verständnis für die besonderen Eigenschaften dieser Arbeit zu vergrößern. Als Beispiel dient die Behandlung eines Patienten mit einer Lungenembolie [DMR15]. Dieser Prozess ist stark wissensbasiert, da er vom medizinische Wissen und der Erfahrung des behandelnden Arztes, der Symptome des Patienten und weiteren fall- und patientenspezifischen Daten abhängt. Darüber hinaus kann die Behandlung es erfordern, dass Wissensarbeiter aus verschiedene Abteilungen dynamisch zusammenarbeiten müssen. Dies erfordert eine aktive Koordination und Kommunikation zwischen den verschiedenen Fachkräften, die verschiedene Fähigkeiten und Kompetenzen besitzen.

Ein erster Schritt im Behandlungsprozess ist die Aufnahme des Patienten und die Sichtung seiner Vorbefunde. Die Krankenakte ist zentral gelagert und repräsentiert das gesammelte Wissen, welches die Entscheidungsprozesse steuert. Sie wächst mit durchgeführten Prozessen und gesammelten Daten während der gesamten Behandlung weiter an. Die Sichtung der Vorbefunde und das klinische Wissen des aufnehmenden Arztes, können zu der Erkennung von Symptomen führen, die auf eine Lungenembolie hindeuten. An dieser Stelle wird gezielt versucht eine Lungenembolie zu diagnostizieren. Dies geschieht anhand einer medizinische Checkliste und zusätzlichen *„Wissensschichten“*, wie dem Wissen des aufnehmenden Arztes, Krankenhaus spezifischem Wissen und Wissen zum Patienten. Durch die Diagnose kann dann eine Checkliste zur Behandlung der Lungenembolie genutzt werden um Aktivitäten zu planen und die involvierten Wissensarbeiter zu bestimmen (Krankenschwestern, Radiologe, etc.). Anschließend werden therapeutische Maßnahmen, wie die Stabilisierung eines instabilen Patienten, Medikation und klinische Einschätzungen verbunden, an welchen verschiedene Fachkräfte des medizinischen Personals beteiligt sind. Das Vorankommen der Behandlung hängt dabei nicht strikt vom Abschließen bestimmter Aufgaben ab, sondern von einer Kombination von klinischen Daten und Einschätzungen und der Behandlung des Patienten (siehe Abbildung 2.1) [LR07]. Des weiteren kann sich der Behandlungsverlauf jederzeit ändern sollte der Patient beispielsweise instabil werden. In diesem Fall müssen alle geplanten Behandlungen neu evaluiert werden.

Das Fallbeispiel veranschaulicht die Kombination von vorhersehbaren Prozessen, wie der Aufnahme und Entlassung eines Patienten, mit unvorhersehbaren Prozessen wie der Behandlung des Patienten, die nur lose strukturiert sind und sich von Fall zu Fall unterscheiden können.

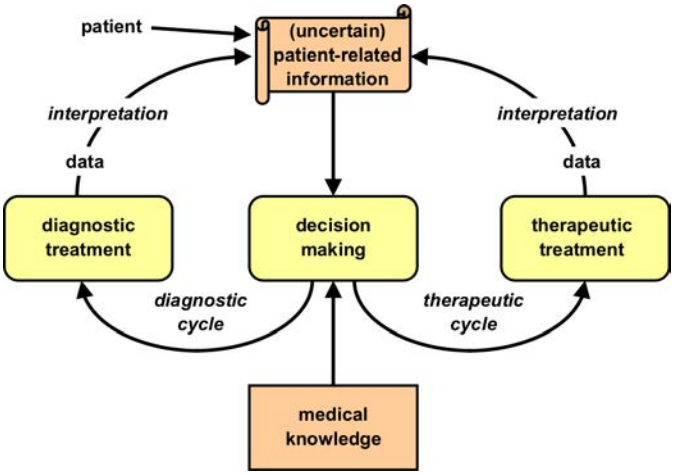


Abbildung 2.1: Einfluss von medizinischem Wissen auf einen Behandlungsprozess

2.3 proCollab

Durch hohe Komplexität und dynamische Einflussfaktoren können Wissensarbeiter nicht auf Werkzeuge zurückgreifen, die für Routinearbeiten ausgelegt wurden. Dies führt zu verminderter Produktivität und behindert die Wiederverwendung und Verbesserung von komplexen Lösungen. Um diese Probleme zu adressieren, wurde am Institut für Datenbanken und Informationssysteme der Universität Ulm *proCollab* (Process-aware Support for Collaborative Knowledge Workers) entwickelt [Pro15]. Das Ziel von *proCollab* ist die Unterstützung von kollaborativen Wissensarbeitern durch einen Ansatz, der zum einen auf einen Lebenszyklus (siehe Abbildung 2.3) beruht und sich zum Anderen die aufgabenbasierte Koordination über Checklisten zu eigen macht.

2.3.1 proCollab Checklisten-Management

Für kollaborative Wissensarbeiten, wie der Behandlung eines Patientens (siehe Abschnitt 2.2) sieht der *proCollab* Ansatz ein Meta-Modell mit den Entitäten organisatorischen Rahmen, Checklisten und Checklisteneinträgen vor. In den folgenden Abschnitten werden die Entitäten und ihre Zusammenhänge näher erläutert.

Organisatorischer Rahmen

Der Begriff des organisatorischen Rahmen ist in Definition 2.2 beschrieben.

Definition 2.2. ***Organisatorischer Rahmen (OR):***
Der Kontext für die Zusammenarbeit in einer Wissensarbeit wird in einem OR festgehalten. Zu diesem gehören Beispielsweise die verschiedenen Verantwortlichkeiten und das Fälligkeitsdatum einer Wissensarbeit. Ein solcher OR kann wiederum weitere, untergeordnete ORs sowie verbundene Checklisten enthalten.

In Bezug auf das Fallbeispiel (siehe Abschnitt 2.2) wäre die Behandlung des Patienten ein OR. Dieser kann in weitere, untergeordnete ORs unterteilt werden, beispielsweise in Aufnahme, Behandlung und Entlassung des Patienten.

Ein OR wird durch verschiedene Attribute beschrieben:

- Name: Der Name stellt die Bezeichnung eines ORs dar.
- Beschreibung: Eine Beschreibung, die einen OR ausführlich beschreibt.
- Ziel: Das bestimmte Ziel des ORs, auf das die beteiligten Wissensarbeiter hinarbeiten.
- Dauer: Die Dauer des ORs in Tagen, Monaten oder Jahren.
- Startdatum: Das Startdatum des ORs.
- Enddatum: Das angepeilte Enddatum des ORs.

Ein OR kann optional noch weitere Attribute enthalten:

- Positionsdaten: Positionsdaten, falls es sich um einen OR handelt, der an einen bestimmten Ort gebunden ist.
- Priorität: Eine Zahl zwischen eins (niedrig) und fünf (hoch), die die Dringlichkeit des ORs anzeigt.

Da sich Wissensarbeiten, wie etwa die Behandlung eines Patienten, oft ähneln können Vorlagen zu der Erstellung konkreter ORs genutzt werden. Diese Vorlagen werden als

organisatorische Rahmentypen (ORT) bezeichnet und werden in Definition 2.3 genauer beschrieben.

Definition 2.3. ***Organisatorischer Rahmentyp (ORT):***
Organisatorische Rahmentypen enthalten generische Attribute und Checklistentypen (siehe Definition 2.6), die für Wissensarbeiten eines Typs benötigt werden. Sie dienen als Vorlage für konkrete organisatorische Rahmeninstanzen, die mit spezifischen Daten erweitert werden können, um den Anforderungen der geplanten Wissensarbeit zu genügen.

Die ORTs erleichtern es, Prozesse vor zu strukturieren und gleichzeitig stetig zu verbessern. Zusätzlich helfen sie dabei, die Zeit zu minimieren, die zum Erstellen einer organisatorischen Rahmeninstanz (siehe Abschnitt 2.4) benötigt wird.

So ist der Ablauf für die Aufnahme eines Patienten meist ähnlich und kann anhand einer Vorlage durchgeführt werden. Allerdings müssen die Patientendaten instanzspezifisch aufgenommen werden, um die Unterstützung über eine organisatorische Rahmeninstanz zu garantieren (siehe Definition 2.4).

Definition 2.4. ***Organisatorische Rahmeninstanz (ORI):***
Eine ORI kann mithilfe eines ORT erstellt und dann weiter verfeinert werden. Alternativ kann eine ORI auch ohne ORT erstellt werden, falls kein ORT den Anforderungen genügt. Ein ORI enthält generell weitere, untergeordnete ORIs, sowie Cheklisteninstanzen (siehe Definition 2.7).

In unserem Beispiel für die Behandlung eines Patienten mit einer Lungenembolie, könnte seine Aufnahme im Krankenhaus in einer ORI festgehalten werden. Da es sich dabei um eine Routineaufgabe handelt kann diese von einem bestehenden ORT abgeleitet werden.

Checkliste

Für Checklisten gibt es viele Einsatzgebiete und somit auch eine Vielzahl von verschiedenen Definitionen. Im medizinischen Bereich werden Checklisten beispielsweise

oft angewandt, um Patientensicherheit zu erhöhen [WRW12]. Vor der Operation eines Patienten müssen dabei Checklisten abgearbeitet werden um Geräte zu prüfen, sicherzustellen dass bestimmte Aufgaben erledigt wurden, z.B. dass der Patient seine Identität bestätigt hat, um Verwechslungen zu vermeiden und um als Erinnerungshilfe in stressvollen Situationen zu fungieren. In der Industrie hingegen werden Checklisten meist dazu genutzt, Prozesse gleichbleibend zu strukturieren und Menschliche Fehler zu verhindern [WRW12]. Die Checkliste ist dabei immer einem OR untergeordnet, welcher ihren Kontext definiert. Sie liefert dabei keine vollständige Anleitung zum Erfüllen einer Aufgabe, sondern ist lediglich ein Hilfsmittel zur Unterstützung von Wissensarbeitern.

Analog zu den organisatorischen Rahmen können Checklisten in drei Arten unterteilt werden. Die allgemeine Checkliste, der Checklistentyp und die Checklisteninstanz. Diese werden in Definition 2.5, 2.6 und 2.7 genauer beschrieben.

Definition 2.5. ***Checkliste (CL):***
Eine CL ist eine hierarchisch geordnete Liste, die Checklisteneinträge beinhaltet, welche erfüllt werden müssen um eine Wissensarbeit voranzutreiben. So können anhand einer Checkliste Wissensarbeiten koordiniert werden. Eine CL hilft damit den Überblick über schon erledigte und noch ausstehende Aufgaben zu gewährleisten [DW93].

Durch den Einsatz von CLs innerhalb von ORs können Fehler in der Kommunikation vermieden und die Kommunikation zwischen Wissensarbeitern erleichtert werden.

Eine CL besteht mindestens aus:

- Name: Eine eindeutige Bezeichnung.

Sie kann jedoch optional noch weitere Daten enthalten:

- Beschreibung: Eine Beschreibung, die die CL näher beschreibt.
- Positionsdaten: Positionsdaten, falls es sich um eine CL handelt, die an einen bestimmten Ort gebunden ist, zum Beispiel wenn ein Krankenwagen das Krankenhaus erreicht und der Patient angemeldet werden muss.

- Priorität: Eine Zahl zwischen eins (niedrig) und fünf (hoch), die die Dringlichkeit der CL anzeigt
- Enddatum: Ein Fälligkeitsdatum falls ein solches für die CL existiert.

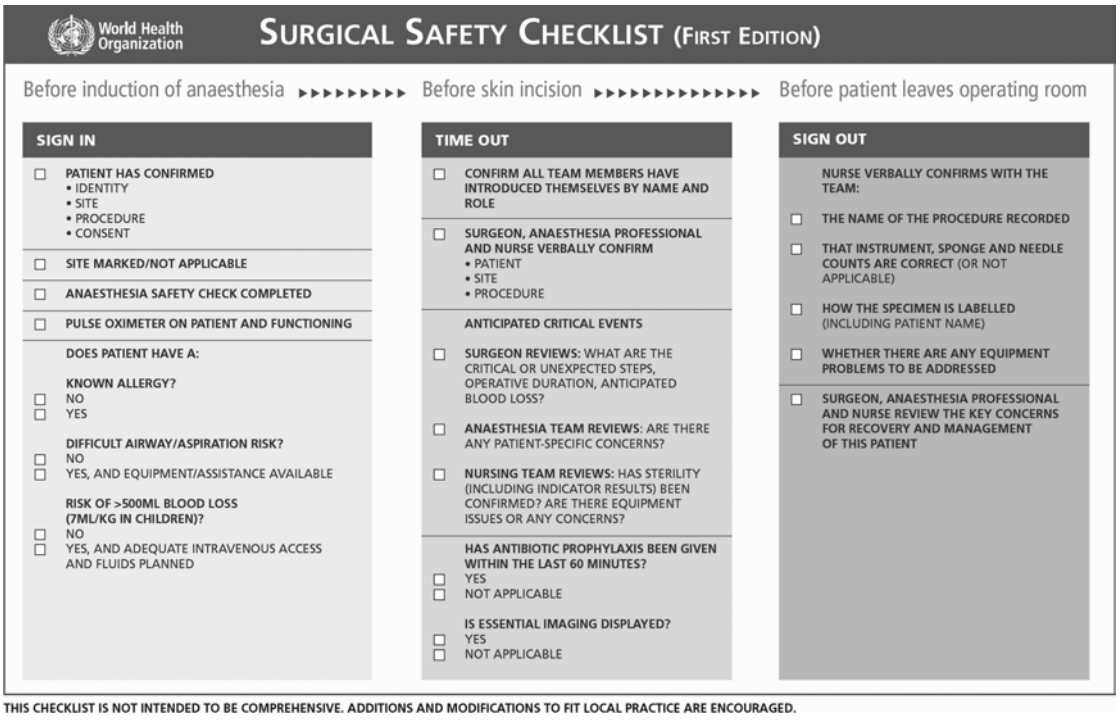
Eine CL ist immer einem OR zugeordnet und besitzt mindestens einen Checklisteneintrag (siehe Definition 2.8).

Definition 2.6. ***Checklistentyp (CLT):***
Ein CLT stellt eine Vorlage dar, in der vordefinierte Attribute und Checklisteneinträge für eine wiederkehrende Nutzung gespeichert werden können. Von einer CLT können konkrete Checklisteninstanzen abgeleitet werden. Nach dem Erstellen einer Checklisteninstanz anhand einer Vorlage, können deren Einträge individuell angepasst und erweitert werden.

Hieraus entstehen die selben Vorteile wie bei der Verwendung von ORTs (siehe Abschnitt 2.3). Diese sind namentlich eine gleichbleibende Struktur, die Möglichkeit Arbeitsabläufe einfach zu optimieren und die Zeitersparnis beim Erstellen von Checklisteninstanzen.

Definition 2.7. ***Checklisteninstanz (CLI):***
Bei einer CLI handelt es sich um eine konkrete CL, die Daten für das Erledigen eines Ziels oder Teilziels enthält. Die einzelnen Teile einer Wissensarbeit sind dabei in weiteren untergeordneten CLIs und Checklisteneintragsinstanzen organisiert. Eine CLI kann entweder von einem CLT abgeleitet werden oder ohne Vorlage erstellt werden. Beim Erstellen ohne Vorlage müssen die entsprechenden Daten, wie der Name der CLI und das Start- und Enddatum manuell eingetragen werden.

In Bezug auf das Fallbeispiel (siehe Abschnitt 2.2) kann eine Checkliste beispielsweise benutzt werden, nachdem bei dem Patienten eine Lungenembolie diagnostiziert wurde und er operativ behandelt werden soll. Da es dabei wichtig ist, dass keine Fehler auftreten kann sich der behandelnde Arzt an einer CL orientieren. Diese stellt sicher, dass keine wichtigen Schritte vergessen werden (siehe Abbildung 2.2) [WHD+10]. Die einzelnen Schritte werden dabei in einzelnen CEs festgehalten.



World Health Organization

SURGICAL SAFETY CHECKLIST (FIRST EDITION)

Before induction of anaesthesia ▸▸▸▸▸▸▸▸▸ Before skin incision ▸▸▸▸▸▸▸▸▸▸▸▸▸ Before patient leaves operating room

SIGN IN

- ☐ PATIENT HAS CONFIRMED
  - IDENTITY
  - SITE
  - PROCEDURE
  - CONSENT
- ☐ SITE MARKED/NOT APPLICABLE
- ☐ ANAESTHESIA SAFETY CHECK COMPLETED
- ☐ PULSE OXIMETER ON PATIENT AND FUNCTIONING
- DOES PATIENT HAVE A:

KNOWN ALLERGY?
- ☐ NO
- ☐ YES

DIFFICULT AIRWAY/ASPIRATION RISK?
- ☐ NO
- ☐ YES, AND EQUIPMENT/ASSISTANCE AVAILABLE

RISK OF >500ML BLOOD LOSS (7ML/KG IN CHILDREN)?
- ☐ NO
- ☐ YES, AND ADEQUATE INTRAVENOUS ACCESS AND FLUIDS PLANNED

TIME OUT

- ☐ CONFIRM ALL TEAM MEMBERS HAVE INTRODUCED THEMSELVES BY NAME AND ROLE
- ☐ SURGEON, ANAESTHESIA PROFESSIONAL AND NURSE VERBALLY CONFIRM
  - PATIENT
  - SITE
  - PROCEDURE

ANTICIPATED CRITICAL EVENTS

- ☐ SURGEON REVIEWS: WHAT ARE THE CRITICAL OR UNEXPECTED STEPS, OPERATIVE DURATION, ANTICIPATED BLOOD LOSS?
- ☐ ANAESTHESIA TEAM REVIEWS: ARE THERE ANY PATIENT-SPECIFIC CONCERNS?
- ☐ NURSING TEAM REVIEWS: HAS STERILITY (INCLUDING INDICATOR RESULTS) BEEN CONFIRMED? ARE THERE EQUIPMENT ISSUES OR ANY CONCERNS?

HAS ANTIBIOTIC PROPHYLAXIS BEEN GIVEN WITHIN THE LAST 60 MINUTES?
- ☐ YES
- ☐ NOT APPLICABLE

IS ESSENTIAL IMAGING DISPLAYED?
- ☐ YES
- ☐ NOT APPLICABLE

SIGN OUT

NURSE VERBALLY CONFIRMS WITH THE TEAM:

- ☐ THE NAME OF THE PROCEDURE RECORDED
- ☐ THAT INSTRUMENT, SPONGE AND NEEDLE COUNTS ARE CORRECT (OR NOT APPLICABLE)
- ☐ HOW THE SPECIMEN IS LABELLED (INCLUDING PATIENT NAME)
- ☐ WHETHER THERE ARE ANY EQUIPMENT PROBLEMS TO BE ADDRESSED
- ☐ SURGEON, ANAESTHESIA PROFESSIONAL AND NURSE REVIEW THE KEY CONCERNS FOR RECOVERY AND MANAGEMENT OF THIS PATIENT

THIS CHECKLIST IS NOT INTENDED TO BE COMPREHENSIVE. ADDITIONS AND MODIFICATIONS TO FIT LOCAL PRACTICE ARE ENCOURAGED.

Abbildung 2.2: Beispiel einer WHO OP-Checkliste

Checklisteneintrag

Auch Checklisteneinträge sind in drei Typen unterteilt, der allgemeine Eintrag, der Eintragstyp und die Eintragsinstanz. Diese sind in den Definitionen 2.8, 2.9 und 2.10 beschrieben.

Definition 2.8. ***Checklisteneintrag (CE):***
Ein CE beschreibt eine Bestimmte Aufgabe, die Teil einer Checkliste ist. Er besteht mindestens aus einem Namen sowie einem Zustand, welcher anzeigt wie weit die Bearbeitung vorangeschritten ist. (Er kann jedoch auch weitere Daten wie etwa ein Fälligkeitsdatum oder Positionsbezogene Daten enthalten)

Ein CE besteht mindestens aus:

- Namen: Eine eindeutige Bezeichnung.
- Zustand: Ein Wert der angibt wie weit die Bearbeitung des Eintrages vorangeschritten ist.

Ein CE kann jedoch optional noch weitere Daten enthalten:

- Beschreibung: Eine Beschreibung, die den CE näher beschreibt, beispielsweise eine Frage, die beantwortet werden muss, oder eine Bedingung die erfüllt sein muss.
- Positionsdaten: Positionsdaten, falls es sich um eine Aufgabe handelt, die an einen bestimmten Ort gebunden ist.
- Priorität: Eine Zahl zwischen eins (niedrig) und fünf (hoch), die die Dringlichkeit des CEs anzeigt
- Enddatum: Ein Fälligkeitsdatum, falls ein solches für den Eintrag existiert.

Auch hier können die CEs anhand von Vorlagen erstellt werden. Eine solche Vorlage wird Checklisteneintragstyp genannt und wird in Definition 2.9 genauer beschrieben.

Definition 2.9. ***Checklisteneintragstyp (CET):***
Der CET ist eine Vorlage anhand derer man neue CEIs initialisieren kann. Die Vorlage beinhaltet dabei den Namen und eine Beschreibung des CEs.

Dies hat den Vorteil, dass Einträge leichter optimiert und gleichbleibend strukturiert werden können.

Definition 2.10. ***Checklisteneintragsinstanz (CEI):***
Eine Checklisteneintragsinstanz enthält Daten (siehe Definition 2.8), die für die Erfüllung der von ihr beschriebenen Aufgabe nötig sind. Sie kann entweder von einem CET abgeleitet werden oder ohne einen Typ speziell für eine CLI erstellt werden.

2.3.2 proCollab Lebenszyklus

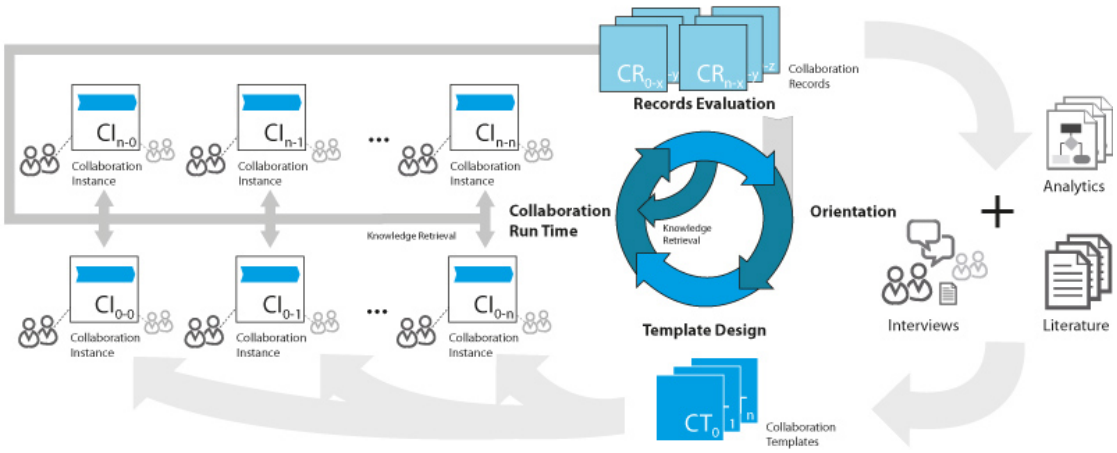


Abbildung 2.3: *proCollab* Lebenszyklus zur Unterstützung kollaborativer Wissensarbeit

Der *proCollab* Lebenszyklus, dargestellt in Abbildung 2.3, besteht aus verschiedenen Phasen, die hier näher beschrieben werden [MKR13].

Orientation Phase: Es werden Informationen gesammelt, die ein Bild davon geben, wie die Wissensarbeiter zusammenarbeiten. Dafür werden die Wissensarbeiter ausführlich

befragt und es wird verwandte Literatur und Expertenwissen evaluiert. Neben dem für die Arbeit nötigen Wissen werden die verschiedenen Prozesse, die für die Erfüllung der Wissensarbeit nötig sind, dokumentiert und integriert. Des weiteren muss ermittelt werden, wie die verschiedenen Wissensarbeiter untereinander kommunizieren.

Template Design Phase: Anhand des so ermittelten Informationsfluss und der angestrebten Zusammenarbeit wird ein ORT für die Wissensarbeit erstellt (siehe Abschnitt 2.3.1). Anhand dieses ORT können nun konkrete Wissensarbeiten in einer ORI (siehe Definition 2.4) strukturiert werden. Die ORI bietet den Wissensarbeitern dabei zugriff auf Informationen, Möglichkeiten der Kommunikation und Koordination, sowie ein gemeinsames Ziel auf das hingearbeitet wird.

Collaboration Run Time Phase: Wissensarbeiter können neue ORIs erstellen um ihre kollaborative Wissensarbeit voranzutreiben. Die ORIs können dabei von ORTs abgeleitet sein oder individuell erstellt werden. Durch die Bereitstellung der abgeschlossenen ORIs haben die Wissensarbeiter zugriff auf Informationen, die dabei helfen, das gemeinsame Ziel schneller zu erreichen.

Records Evaluation Phase: Abgeschlossene ORIs können als wichtige Wissensquelle angesehen werden. Durch Änderungen, die and ORIs vorgenommen wurden und Befragungen der Wissensarbeiter können die ORTs optimiert werden um den Ablauf zukünftiger Wissensarbeiten zu verbessern.

3

Anforderungen

3.1 Analyse Ist-Stand

Die Analyse des Ist-Standes beschäftigt sich im ersten Teil mit der Analyse von vergleichbaren Systemen und vergleicht diese im zweiten Teil mit dem derzeitigen Funktionsumfang von *proCollab*. Dieser Vergleich dient dazu, dass festgestellt werden kann welche nützlichen Funktionen *proCollab* bisher noch nicht unterstützt, und wie diese in anderen Applikationen umgesetzt wurden.

3.1.1 Any.DO

Ein vergleichbares System, mit welchem sich Aufgaben und Notizen verwalten lassen ist Any.DO [Any15]. Die Anmeldung für Any.Do kann übersprungen werden, allerdings stehen dann Funktionen wie das Teilen von Listen mit anderen Benutzern nicht zur Verfügung. Die Anmeldung kann entweder über einen bestehenden Google oder Facebook Account als auch mit einer Registrierung per E-Mail erfolgen. Nach der Anmeldung können Aufgaben in ein Textfeld eingetragen werden (siehe Markierung 1, Abbildung 3.1). Wählt man einen so erstellten Eintrag aus (siehe Markierung 2, Abbildung 3.1), kann dieser durch weitere Daten ergänzt werden. Dazu gehören die auf der rechten Seite von Abbildung 3.1 abgebildeten Einstellungen: wann an die Aufgabe erinnert werden soll, wann und ob sie wiederholt werden soll, sowie die Möglichkeit die Aufgabe mit Positionsdaten zu verknüpfen.

Eine Aufgabe kann auch mit Dateien, die mit ihr in Verbindung stehen, verknüpft werden. Zusätzlich lassen sich Unteraufgaben erstellen und es ist möglich, andere Nutzer zu

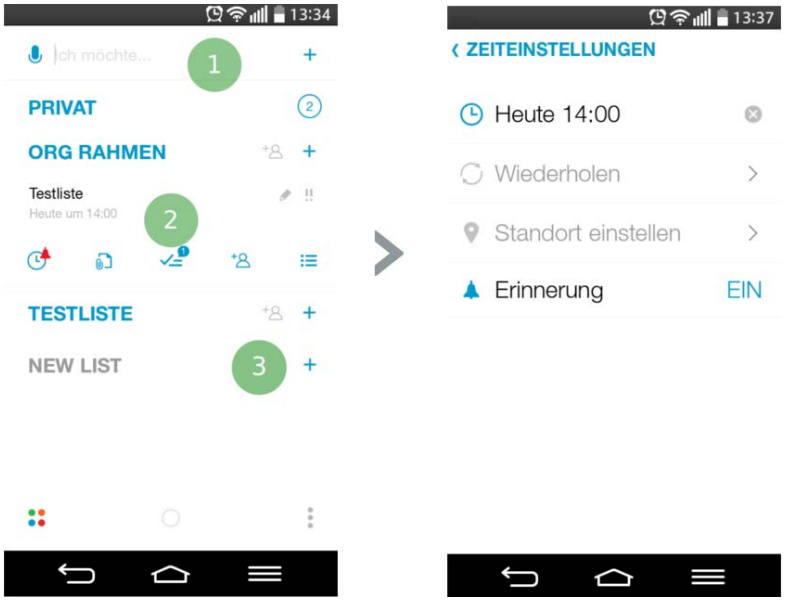


Abbildung 3.1: Beispieldialog der Anwendung *Any.DO*

einer Zusammenarbeit einzuladen. Durch das auswählen des Eintrages *„NEW LIST“* (Markierung 3, Abbildung 3.1) lassen sich neue Listen anlegen. Die Listen selbst enthalten hierbei keine Daten außer ihrem Namen.

Aufgaben können entweder in Zusammenhang mit ihren Listen oder in einer Ansicht, die die Aufgaben nach ihrem Fälligkeitsdatum sortiert, betrachtet werden.

3.1.2 Wunderlist

Wunderlist ist eine weitere mobile Applikation, die es erlaubt Aufgaben zu verwalten [Wun15]. Die Anmeldung erfolgt hierbei entweder durch das Erstellen eines Accounts per Angabe von Name, E-Mail Adresse und Passwort, oder alternativ per Google- oder Facebook-Account. Der Nutzer gelangt zu einer Ansicht, auf der die Bestehenden Checklisten angezeigt und neue erstellt werden können. Zur Erstellung einer Liste muss nur deren Name angegeben werden. Es besteht zusätzlich die Möglichkeit, weitere Personen zu einer Zusammenarbeit einzuladen. Wählt man eine Checkliste aus, gelangt man zu einer Detailansicht der Liste, in der die bisher erstellten Aufgaben angezeigt werden. Durch das Tippen auf das Textfeld (siehe Abbildung 3.2, Markierung 1) können

neue Einträge erstellt werden. Ein Drücken auf den Stern, markiert den gewählten Eintrag als wichtig. Die Checkbox eines Eintrages erlaubt es diesen als Abgeschlossen zu markieren. Hält man einen Eintrag gedrückt, kann dieser in eine andere Liste verschoben werden (siehe Abbildung 3.2, Markierung 2). Einfaches Drücken öffnet eine Detailansicht des gewählten Eintrages, in der ein Enddatum eingetragen werden kann. Des Weiteren kann man sich an die Aufgabe erinnern lassen und Notizen für sie anlegen. In der Leiste am unteren Bildschirmrand befinden sich Symbole um eine Checkliste zu verwalten. Mit dem Symbol bei Markierung 3 lassen sich weitere Personen zu einer Zusammenarbeit einladen. Das Symbol bei Markierung 4 erlaubt es, Einstellungen für die gesamte Checkliste zu ändern. Hier kann etwa eine Beschreibung für die Checkliste eingegeben werden und eingestellt werden, ob es sich um eine nummerierte Liste handelt oder nicht. Durch klicken auf den Stift bei Markierung 5 lassen sich Listen verwalten. Hier kann der Name geändert werden, Personen zur Zusammenarbeit eingeladen oder von dieser entfernt werden, sowie die gesamte Checkliste gelöscht werden.

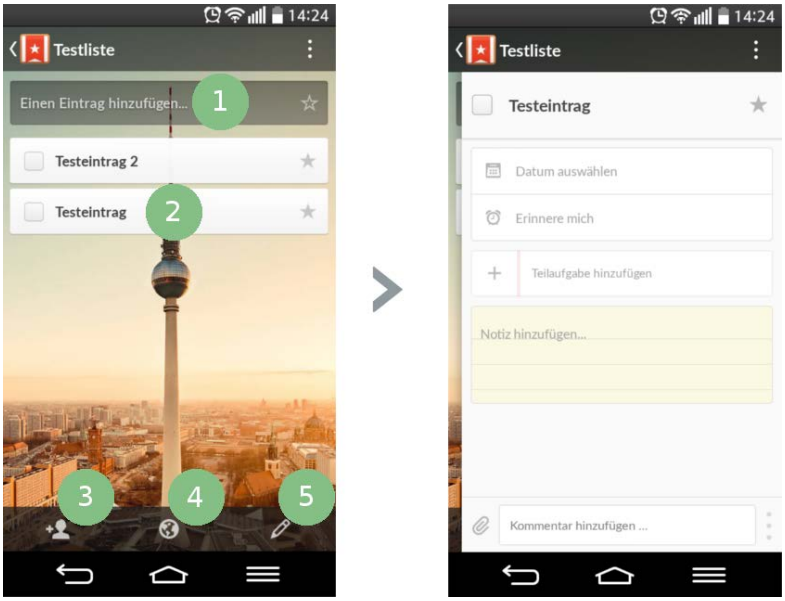


Abbildung 3.2: Beispieldialog der Anwendung *Wunderlist*

3.1.3 Todoist

Todoist ist eine weitere mobile Applikation für das Organisieren von Aufgaben [Tod15]. Die Anmeldung erfolgt hierbei über einen Google-Account, oder alternativ per E-Mail Adresse und Passwort. Durch das auswählen der Schaltfläche *Plus* (Abbildung 3.3, Markierung 1) lassen sich neue Projekte und Unterprojekte erstellen. Projekten können, für eine bessere Übersicht, Farben zugeordnet werden. Durch das Auswählen eines Projektes gelangt man in die mit Markierung 2 gekennzeichnete Ansicht. Hier können einem gewählten Projekt Aufgaben zugewiesen werden. Eine Aufgabe besteht hierbei aus einem Namen, einem Fälligkeitsdatum und einer Priorität. Durch das Drücken der Menütaste können weiter Mitarbeiter zu einer Zusammenarbeit in einem bestimmten Projekt eingeladen werden. Durch Auswählen des Todoist-Symbols wird die Übersicht über alle Projekte eingeblendet (siehe Abbildung 3.4). Diese können über Filter nach ihrer Priorität, ihrer Fälligkeit und welchem Nutzer sie zugewiesen sind sortiert werden.

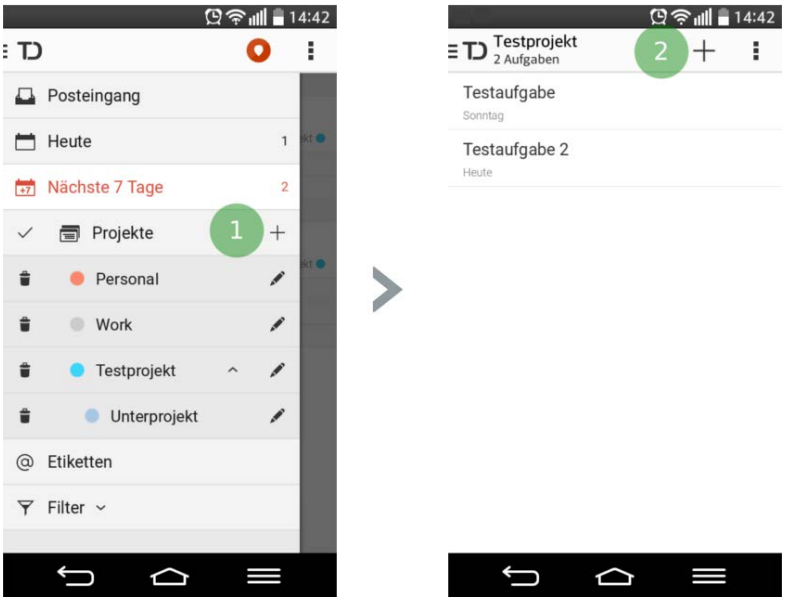


Abbildung 3.3: Beispieldialog der Anwendung *Todoist*

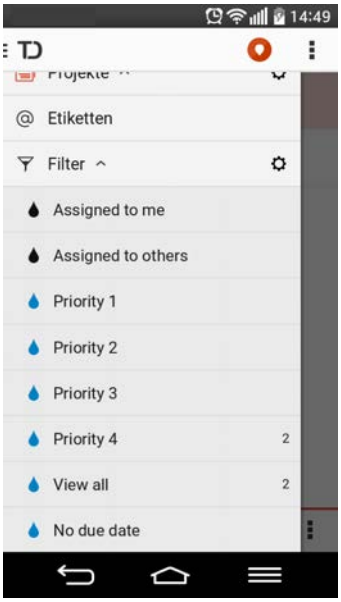


Abbildung 3.4: Anzeige der Filter, die das Sortieren von Listen in *Todoist* erlauben

3.1.4 proCollab

Die mobile Applikation für Smartphones *proCollab* (pC-App) verfügt bereits über viele der grundlegenden Funktionen. Nachdem der Benutzer sich für die pC-App durch das Ausfüllen eines Formulars registriert hat, stehen ihm Funktionen zum Verwalten seiner organisatorischen Rahmen zur Verfügung. Wählt er die Option *„Organisatorischer Rahmen“* (Abbildung 3.5, Markierung 1) aus, so gelangt er auf eine neue Seite, die ihm verschiedene Optionen zum Verwalten seiner organisatorischen Rahmen anbietet. Unter „Organisatorische Rahmentypen“ können alle zur Verfügung stehenden Vorlagen betrachtet werden. Anhand dieser kann dann ein neuer organisatorischer Rahmen erstellt werden. Sollte keine passende Vorlage vorhanden sein, kann unter dem Punkt *„Organisatorische Rahmeninstanz erzeugen“* eine individuelle Rahmeninstanz erzeugt werden. Durch die Auswahl von *„Organisatorische Rahmeninstanzen“* (Abbildung 3.5, Markierung 2) kann der Wissensarbeiter schließlich alle organisatorischen Rahmeninstanzen, in denen er involviert ist, betrachten. Um einen schnelleren Überblick zu erhalten, kann er diese zusätzlich nach bestimmten Stichworten und Daten filtern (Abbildung 3.5, Markierung 3). Jedem dieser organisatorischen Rahmen können wiederum

weitere organisatorische Rahmen, sowie Checklisten mit diversen Checklisteneinträgen hinzugefügt werden. Diese Checklisteneinträge können, je nach dem wie weit ihre Bearbeitung vorangeschritten ist, mit einem Status wie *„Offen“* oder *„Abgeschlossen“* versehen werden. Da *proCollab* großen Wert auf die Zusammenarbeit zwischen Wissensarbeitern legt, ist es möglich, einem organisatorischen Rahmen Wissensarbeiter mit verschiedenen Rollen zuzuweisen. Diese können dann gemeinsam den gewählten Rahmen bearbeiten.

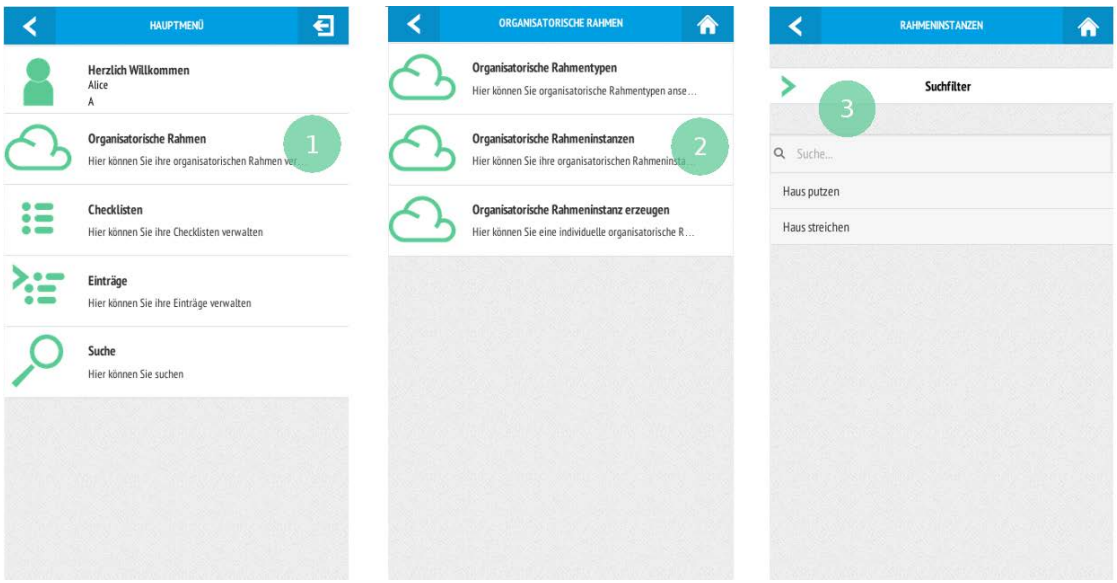


Abbildung 3.5: Anzeigen von organisatorischen Rahmeninstanzen

In der Benutzerverwaltung können persönliche Daten geändert, oder das Konto gar gelöscht werden (Abbildung 3.6, Markierung 1). Zudem existiert eine Suchfunktion, die es dem Nutzer erlaubt nach einer Vielzahl von Daten zu suchen. Dazu gehört unter anderem die Suche nach anderen Wissensarbeitern sowie nach Checklisten und deren Checklisteneinträge (Abbildung 3.6, Markierung 2).

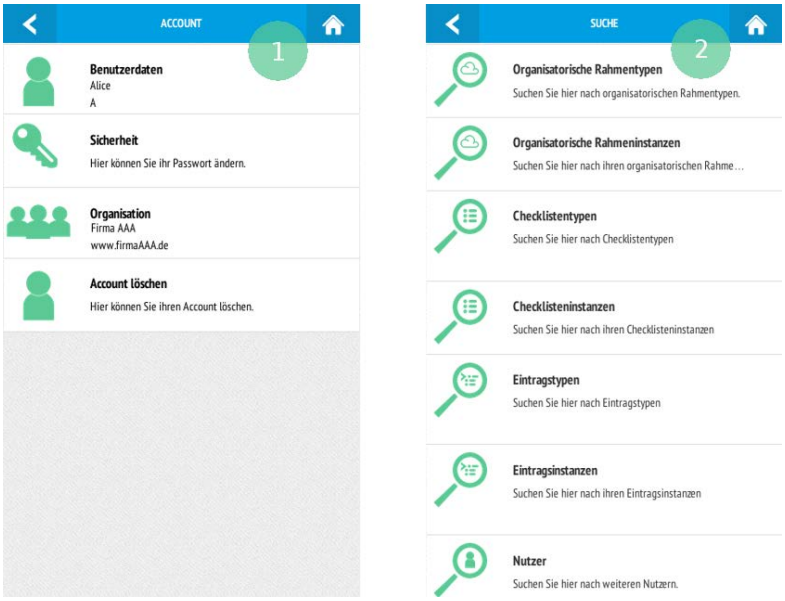


Abbildung 3.6: Oberfläche der Benutzerverwaltung und Suche

3.2 Analyse Soll-Stand

Wie Abschnitt 3.1 gezeigt hat, unterstützt die pC-App bereits eine Vielzahl der für eine Unterstützung von kollaborativer Wissensarbeit, wünschenswerten Funktionen. Allerdings hat der Vergleich mit den etablierten Anwendungen, sowie die Analyse des derzeitigen *proCollab* Prototypen gezeigt, dass noch einige wünschenswerte Funktionen fehlen. Diese sollen die Wissensarbeiter weiter entlasten und für eine bessere Koordination und Kommunikation sorgen.

Verbesserung der Benutzeroberfläche

In Zukunft soll es einfacher möglich sein, einen schnellen Überblick über Aufgaben zu erhalten, die eine hohe Priorität haben, sich in räumlicher Nähe befinden oder bald fällig sind. Dadurch soll verhindert werden, dass wichtige Aufgaben keine Beachtung finden oder Fristen unbemerkt verstreichen. Durch diese Funktionen wird dem Wissensarbeiter zusätzlich ein Teil des Organisationsaufwand abgenommen.

Es soll zukünftig auch möglich sein die verschiedenen Typen zu unterscheiden. Dies ist bisher nur schwer möglich, da ORIs, CLIs und CEIs kein eindeutiges Symbol besitzen, anhand dessen sie zu unterscheiden sind. Dies verringert die Übersichtlichkeit der App stark und kann leicht zu Fehlern führen.

Zusätzlich soll das Anzeigen von Details erleichtert werden. Bisher muss ein Wissensarbeiter beispielsweise eine eigene Seite aufrufen, wenn er erfahren will, wann ein Eintrag fällig ist und welche weiteren Details er besitzt. Dies ist ein unnötiger Mehraufwand, da diese Details auch leicht auf der selben Seite wie die Einträge dargestellt werden können. Dies führt zu einem schnelleren und einfacheren Arbeitsfluss.

Ein weiterer Schwachpunkt der Benutzeroberfläche ist es, dass derzeit keine Aussagekräftige Fehlermeldungen zurückgegeben werden, sollte eine Aktion auf Probleme stoßen. Dies kann leicht zu Verwirrungen führen, da oft nicht klar ist ob eine Aktion ausgeführt wurde oder nicht. Dieser Mangel soll mithilfe von aussagekräftigen Fehlermeldungen, die über die ganze pC-App hinweg ein uniformes Aussehen haben, gelöst werden.

proCollab speichert derzeit noch nicht an welcher Position man sich in einem organisatorischen Rahmen oder einer Checkliste befindet. Dies hat allerdings den Nachteil, dass man immer wieder an die gleiche Stelle navigieren muss, sollte man gleichzeitig mit mehreren Rahmen arbeiten. Dies führt zu einem erheblich größeren Arbeitsaufwand, der in Zukunft vermieden werden soll.

Des weiteren soll die Benutzeroberfläche dahingehend optimiert werden, dass oft benötigte Funktionen, so schnell wie möglich erreichbar sind. Dies erlaubt eine schnellere Bearbeitung von Wissensarbeiten mit Hilfe der pC-App.

Verwalten von Listen

Bisher ist es nicht möglich, ORIs in der pC-App zu verwalten. Dazu gehören Funktionen wie das Verschieben von Einträgen innerhalb einer ORI, sowie zwischen verschiedenen ORIs und das Zusammenfügen von verschiedenen ORIs. Ohne diese Funktionen müssten ORIs und CLIs bei Änderungen komplett neu erstellt werden, was einen großen

Arbeitsaufwand mit sich bringt. Deshalb soll eine einfache Benutzeroberfläche erstellt werden, die es den Wissensarbeitern erlaubt, die gewünschten Aktionen intuitiv zu erledigen.

Offlinebetrieb

Es derzeit noch nicht möglich die pC-App ohne Verbindung zum Internet zu benutzen, da die dazu notwendigen Daten noch nicht zwischengespeichert werden. Dies schadet der Benutzerfreundlichkeit sehr, da nicht immer davon ausgegangen werden kann, dass eine Verbindung mit dem Internet besteht. Dies würde dazu führen, dass Wissensarbeiter wieder auf andere Werkzeuge zurückgreifen müssten. Allerdings kann es leicht zu Problemen kommen, sollte man Änderungen im Offlinebetrieb zulassen. Dazu gehören zum Beispiel:

- Es werden Änderungen an einem Element ausgeführt das auf dem Server nicht mehr existiert.
- Der Nutzer hat die nötigen Berechtigungen für eine Aktion auf Serverseite nicht mehr.
- Ein anderer Nutzer hat die gleichen, oder sehr ähnliche, Daten bereits eingefügt.
- Wie werden Konflikte, die entdeckt werden nachdem der Benutzer wieder online ist, gelöst?

Mehrsprachige Benutzeroberfläche

Da wie in Kapitel 1 erwähnt wurde, die Globalisierung immer weiter voranschreitet, ist es wichtig die pC-App möglicht vielen Menschen zugänglich zu machen. Dies erlaubt es internationale Zusammenarbeit leichter zu koordinieren. Deshalb soll die pC-App zukünftig in mehreren Sprachen angeboten werden.

4

Konzept

In Kapitel 3.2 wurden die Anforderungen für Funktionen und Verbesserungen der Benutzeroberfläche beschrieben, die in *proCollab* noch nicht integriert worden sind. Dieses Kapitel diskutiert verschiedene Lösungsansätze für diese Funktionalität. In Kapitel 5 folgt schließlich die Beschreibung der gewählten Implementierung und deren konkreten Umsetzung.

4.1 Erweiterung der Funktionalität

Dieser Abschnitt diskutiert verschiedene Lösungsansätze für die Integrierung der in Kapitel 3 beschriebenen Funktionalität.

4.1.1 Verwaltung von ORIs

Im aktuellen Zustand der pC-App gibt es keine einfache Möglichkeit Teile einer ORI, wie CLIs und untergeordnete ORIs, in eine andere ORI zu verschieben. Des weiteren ist es bisher nicht möglich zwei ORIs zu einer zusammenzufassen. Dies bedeutet aber, dass es nur schwer möglich ist ORIs umzustrukturieren, nachdem sie erstellt wurden (siehe Abschnitt 3.2).

Eine Möglichkeit diese Funktionalität zu realisieren, ist in einem ersten Schritt alle ORIs anzuzeigen, die keinem anderem ORI untergeordnet sind. In diesen können dann, eine oder zwei ORIs ausgewählt werden, die verwaltet werden sollen. Wählt der Wissensarbeiter zwei ORIs aus können weitere Optionen eingeblendet werden, die es zum Beispiel erlauben die beiden ORIs zu einer zusammenzufassen (siehe Abbildung

(a) (b)

Abbildung 4.1: Verschieben einer Checklisteninstanz

4.1a). In einem zweiten Schritt können dann die beiden ORIs unter- oder nebeneinander dargestellt werden. Nun können die verschiedenen ORIs und CLIs der beiden ORIs, beliebig per Drag and Drop verschoben werden (siehe Abbildung 4.1b).

Ein anderer Ansatz ist, dass in einem ersten Schritt in einer ORI alle CLIs und ORIs markiert werden müssen, die verschoben werden sollen (siehe Abbildung 4.2a). In einem zweiten Schritt wird dann die ORI oder die CLI gewählt in die, die gewählten ORIs und CLIs verschoben werden sollen. Diese Vorgehensweise hat allerdings den Nachteil, dass alle ausgewählten ORIs und CLIs der selben ORI oder CLI zugeordnet werden (siehe Abbildung 4.2b).

(a) (b)

Abbildung 4.2: Eine Möglichkeit für das Verschieben von Organisatorischen Rahmen

4.1.2 Push Notifications

Da mehrere Wissensarbeiter an einer organisatorischen Rahmeninstanz beteiligt sein können, ist es wichtig, dass diese über etwaige Änderungen an dieser Rahmeninstanz informiert werden. Etwaige Änderungen könnten durch farbliche Hervorhebung oder ein spezielles Symbol angezeigt werden (siehe Abbildung 4.3b). Um diese Informationen zu erhalten müssen die Wissensarbeiter allerdings die pC-App öffnen. Um Wissensarbeiter über Änderungen zu informieren, ohne dass die pC-App geöffnet ist könnten Push Notifications verwendet werden. Diese können an die betroffenen Wissensarbeiter gesendet werden, sobald eine Änderung in der organisatorischen Rahmeninstanz auftritt. Dies hat den Vorteil, dass sich Wissensarbeiter jederzeit über Änderungen informieren lassen können, obwohl die pC-App nicht geöffnet ist (siehe Abbildung 4.3a). Dies wird dadurch erreicht, dass der Server des *proCollab* Prototypen die Entsprechende Nachricht an einen Server von Google weiterleitet, welcher dann die Nachricht an das Gerät des betreffenden Wissensarbeiter sendet [Goo15a].

4.1.3 Offlinebetrieb

Anzeigen von Daten im Offlinebetrieb

Da nicht davon ausgegangen werden kann, dass eine Verbindung zum Internet besteht, ist es wichtig, dass auch in einem *Offlinebetrieb* die Grundfunktionen der pC-App verfügbar sind (siehe Abschnitt 3.2). Bisher bietet die pC-App noch keinen lokalen Speicher an, der es ermöglicht zwischengespeicherte Daten auch offline zu betrachten (siehe Abbildung 4.4a). Eine Möglichkeit diesen Offlinebetrieb zu realisieren, wäre es in bestimmten Abständen beim Server anzufragen ob es Änderungen gab und diese dann in einen lokalen Speicher zu übernehmen. Kann die Verbindung mit dem Server nicht aufgebaut werden, können die Daten aus dem lokalen Speicher angezeigt werden. Dies hat allerdings den Nachteil, dass man über Änderungen nicht sofort informiert wird, sondern darauf warten muss, dass die pC-App erneut mit dem Server synchronisiert.

Eine andere Herangehensweise der Aktualisierung der lokalen Daten wäre es, alle Daten von Anfragen an den Server in einen lokalen Speicher zu schreiben. Das heißt es

(a) (b)

Abbildung 4.3: Benachrichtigung über Änderungen in einem organisatorischen Rahmen

werden unter anderem alle ORIs und deren CLIs in einen lokalen Speicher geschrieben. Während nun geprüft wird ob auf dem Server neue Daten vorhanden sind können die zwischengespeicherten Daten angezeigt werden. Liefert der Server nun andere Daten, als die aus dem lokalen Speicher angezeigten, wird die Benutzeroberfläche und der lokale Speicher mit diesen aktualisiert (siehe Abbildung 4.4b).

Änderungen im Offlinebetrieb

Änderungen an ORIs, CLIs oder CEIs, im *Offlinebetrieb* bringen viele Schwierigkeiten mit sich, da es leicht zu Konflikten zwischen verschiedenen Änderungen kommen kann. Eine Möglichkeit mit diesen Hindernissen umzugehen ist es, alle Änderungen zwischenzuspeichern und sie dann, später, wenn der Server wieder erreichbar ist, in der Reihenfolge in der sie getätigt wurden an den Server zu senden. Sollte es nun zu Konflikten kommen, die nicht automatisch behoben werden können, wird dem Benutzer eine spezielle Seite angezeigt, auf der die Konflikte betrachtet und gelöst werden können [Wei15]. Manche auftretenden Konflikte können hierbei aber nicht gelöst werden, beispielsweise etwa dann, wenn der Benutzer die nötigen Berechtigungen nicht mehr besitzt. Durch die manuelle Konfliktlösung kann ein erheblicher Arbeitsaufwand entstehen, da der Nutzer viele bis alle Änderungen noch einmal begutachten und entscheiden muss, welche Änderungen übernommen und welche verworfen werden sollen (siehe Abbildung 4.5b).

Eine alternative Möglichkeit diese Schwierigkeiten zu vermeiden ist es, das Ändern von Daten, die kollaborativ geändert werden könne, im Offlinebetrieb nicht zu gestatten. Dies schränkt natürlich die Nutzbarkeit der pC-App im Offlinebetrieb stark ein, da nur bereits angelegte Daten, die sich im lokalen Speicher befinde, betrachtet werden können. Dadurch wird die Konfliktlösung allerdings stark vereinfacht, da solange die pC-App eine Verbindung mit dem Server hat, immer nur ein Konflikt auftreten kann, der gelöst werden muss [Wei15]. Dadurch kann auf eine eigene Oberfläche für die Konfliktlösung bei Änderungen in einem dedizierten Offlinebetrieb verzichtet werden (siehe Abbildung 4.5a).

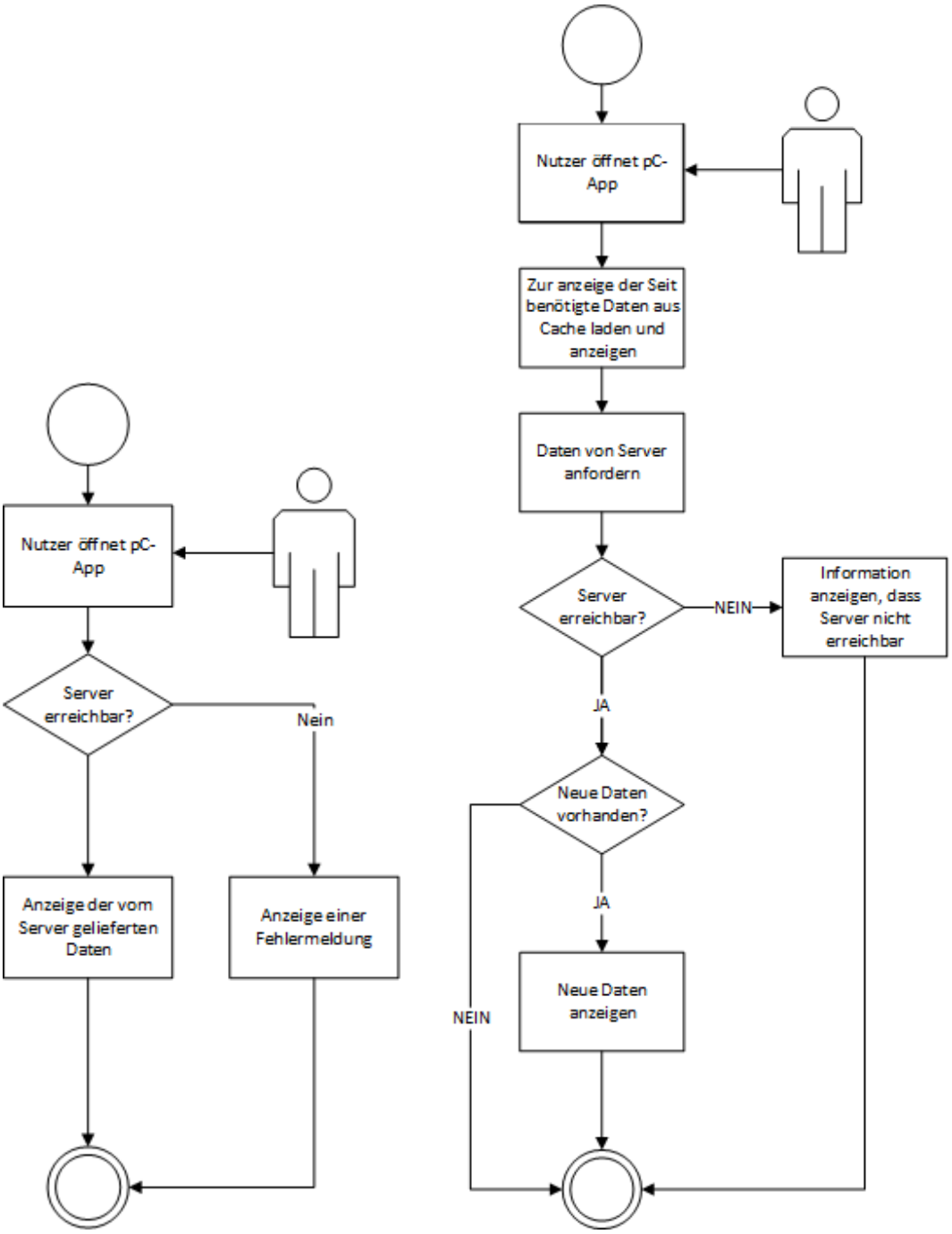


Abbildung 4.4: Offlinebetrieb der pC-App mit (b) und ohne (a) lokalem Speicher

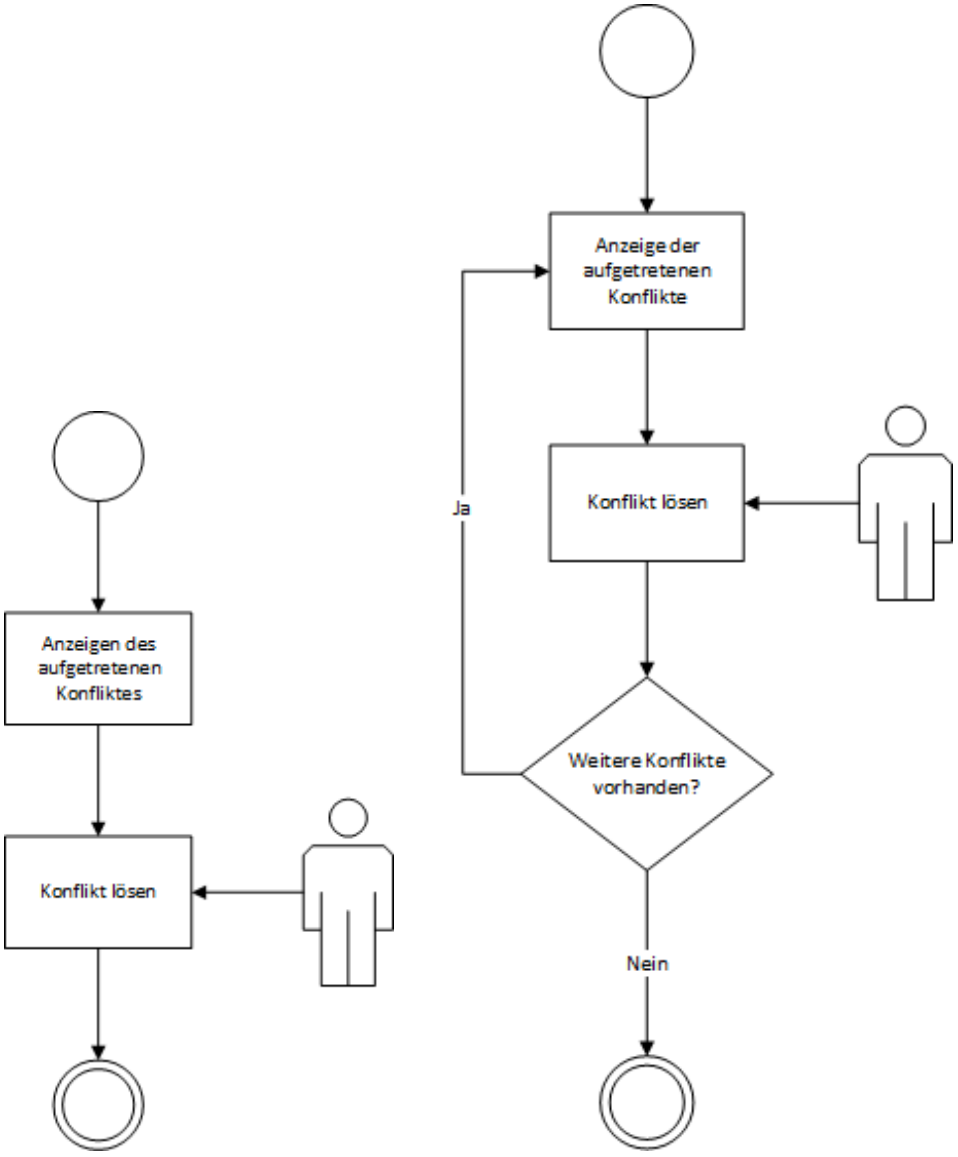


Abbildung 4.5: Verschiedene Konfliktlösungsstrategien

4.1.4 Personalisierung der pC-App

Verschiedene ORIs können unterschiedliche Anforderungen besitzen. Daher ist es wichtig, dass Einstellungen wie die räumliche Distanz bis zu welcher eine Markierung angezeigt wird, oder ab wann ein Alarm für ein ablaufendes Element angezeigt wird, konfiguriert werden können (siehe Abschnitt 3.2).

Ein Ansatz ist es, dass diese Konfigurationsdaten in jedem erstellten OR hinterlegt werden, so dass alle Benutzer der pC-App eine gleiche Umgebung vorfinden. Dies hat allerdings den Nachteil, dass die Symbole zwischen verschiedenen ORIs inkonsistent verwendet werden können, was leicht zu Fehlern führen kann. Beispielsweise könnten Wissensarbeiter die Zeit, die für die Bearbeitung einer ORI verbleibt falsch einschätzen, wenn das Symbol, das anzeigt, dass eine ORI bald fällig ist, für eine ORI bereits sieben Tage vor dem Enddatum erscheint und in einer andern ORI erst einen Tag vor dem Enddatum erscheint.

Ein weiterer Ansatz ist es, den Benutzer selbst diese Einstellungen vornehmen zu lassen. Diese Einstellungen gelten dann für die gesamte pC-App so dass sie konsistent verwendet werden. Allerdings können dadurch verschiedene ORIs keine unterschiedlichen Einstellungen erhalten.

4.1.5 Mehrsprachige Benutzeroberfläche

Durch die zunehmende Globalisierung ist es wichtig, dass die pC-App mehr als nur eine Sprache unterstützt, damit ein möglichst großes Publikum erreicht werden kann und Internationale Kollaborationen leichter möglich werden (siehe Abschnitt 3.2). Dies könnte durch eine eigene pC-App, pro Sprache gelöst werden, allerdings müssten bei diesem Ansatz alle Änderungen in den verschiedenen pC-Apps jeweils einzeln angewandt werden. Sprachdateien, die erst zur Laufzeit entsprechend der gewünschten Sprache geladen werden, beheben dieses Problem, da alle Nutzer die gleiche pC-App verwenden. Es wäre zudem möglich die Sprachdateien von einem Server zu laden, um immer eine Aktuelle Sprachdatei auf dem Gerät zu haben. Dieser Ansatz hat allerdings

den Nachteil, dass er nicht so performant ist wie eine statische Benutzeroberfläche, die nur eine Sprache unterstützt.

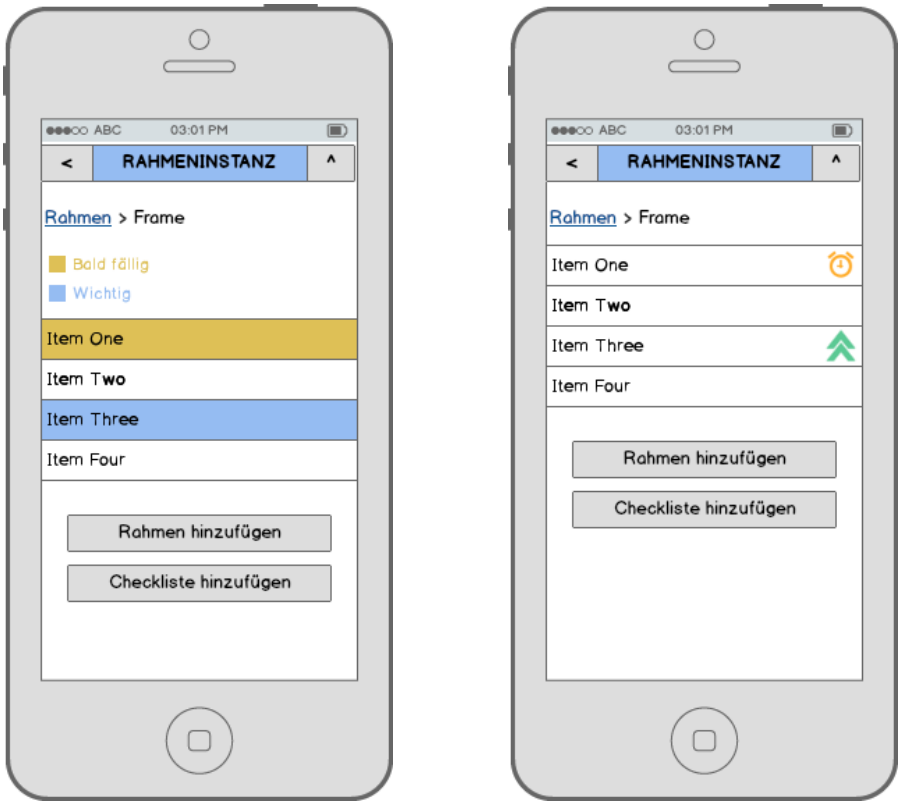
Weiterhin wäre es möglich, beide genannten Ansätze zu kombinieren und anhand von Sprachdateien für jede Sprache eine eigene pC-App zu erzeugen. Allerdings würde dieser Ansatz beim Aktualisieren der pC-App zu viel Mehraufwand führen.

4.2 Verbesserung der Benutzeroberfläche

Um eine hohe Akzeptanz, für die pC-App, bei den Wissensarbeitern zu erreichen ist es wichtig, dass die Anwendung Informationen einfach darstellt und vermieden wird, dass gleiche Schritte wiederholt getätigt werden müssen.

4.2.1 Position in ORIs und CLIs

Eine ORI oder CLI kann als ein einfacher Baum dargestellt werden. Dabei ist die ORI bzw. die CLI selbst, die Wurzel des Baumes. Die untergeordneten ORIs und CLIs sind seine Knoten und CEIs seine Blätter. Aufgrund dieser hierarchischen Darstellung können ORIs und CLIs eine beliebige Tiefe besitzen. Daher ist es von Vorteil, wenn die Position in diesem Baum, an der sich ein Nutzer der pC-App befindet, gespeichert wird. Dadurch muss nach einem erneuten Aufrufen der ORI oder CLI nicht wieder bis zu der zuletzt besuchten Position navigiert werden. Im folgenden wird diese Position als Zielknoten bezeichnet. Für die Speicherung der Position, an der sich ein Nutzer in einer ORI oder CLI befindet gibt es zwei grundsätzliche Herangehensweisen. Zum einen kann der gesamte Pfad zu dem Zielknoten gespeichert werden, das heißt es werden alle Knoten zwischen der Wurzel des Baumes und der gewünschten Position gespeichert. Zum anderen kann aber auch nur der Wurzelknoten und die gewünschte Position gespeichert werden. Das Speichern des gesamten Pfades hat den Vorteil, dass der Pfad nicht bestimmt werden muss. Allerdings ist es bei dieser Herangehensweise schwerer, Änderungen in der ORI oder CLI zu verarbeiten, weil bei jeder Änderung überprüft werden muss ob sich der Pfad von Wurzel- zu Zielknoten verändert hat. Dies muss bei der Speicherung von Wurzel- und Zielknoten nicht berücksichtigt werden, da



(a) (b)

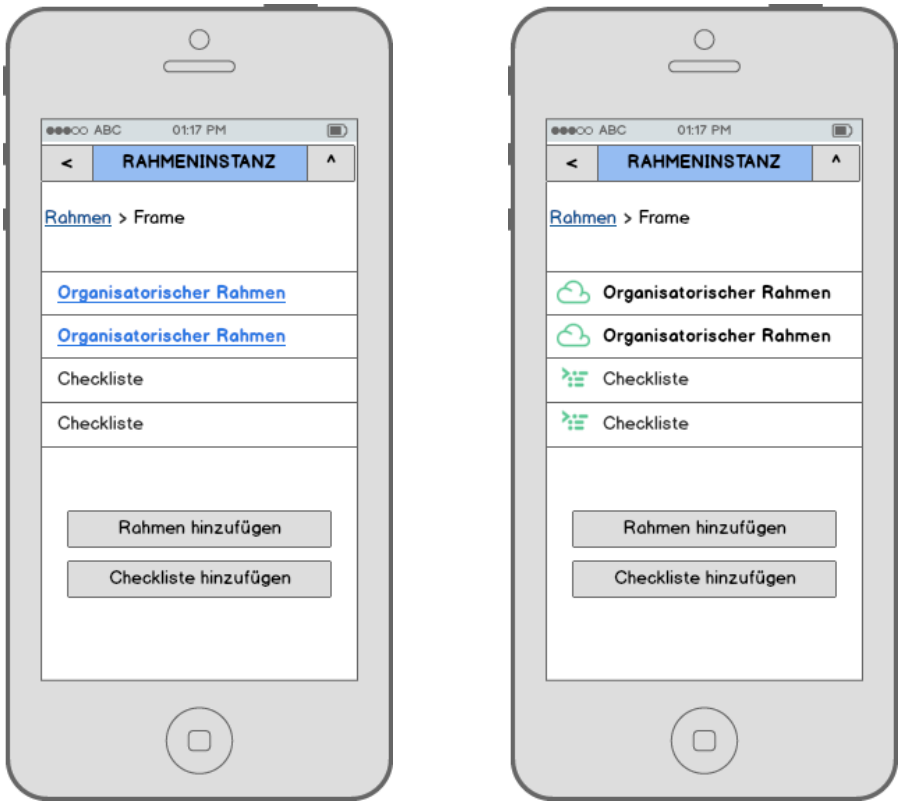
Abbildung 4.6: Farbliche Hervorhebung (a) im Vergleich mit Symbolen (b)

der Pfad immer bestimmt werden kann, solange Start- und Zielknoten sich noch in der ORI oder CLI befinden. Allerdings wird dazu eine etwas längere Laufzeit benötigt, da der Pfad bei jedem Aufruf erneut bestimmt werden muss.

4.2.2 Hervorheben von Priorität und Fälligkeitsdatum

Die Priorität und die Fälligkeit lassen sich beispielsweise per farblicher Markierung oder mit Symbolen darstellen (siehe Abbildung 4.6). Symbole haben hierbei den Nachteil, dass sie Platz verbrauchen, welcher auf mobilen Applikationen typischerweise schnell knapp wird [Bre02].

Eine farbliche Markierung würde dagegen keinen Platz einnehmen. Jedoch kann die pC-App sehr schnell unübersichtlich werden, da eine farbliche Hervorhebung bei weitem



(a) (b)

Abbildung 4.7: Farbliche Hervorhebung (a) im Vergleich mit Symbolen (b)

nicht so intuitiv wie ein Symbole ist, da zusätzlich eine Legende für die verschiedenen Farben benötigt wird.

4.2.3 Unterscheidung von ORI, CLI und CEI

Wie in Abschnitt 3.2 beschrieben, besitzen ORI, CLI und CEI bisher kein eindeutiges Unterscheidungsmerkmal in der Benutzeroberfläche. Für eine verbesserte Unterscheidung, gibt es die Möglichkeit ORI, CLI und CEI verschieden, farblich hervorzuheben (siehe Abbildung 4.7a) oder mit Symbolen zu markieren (siehe Abbildung 4.7b). Darüber hinaus kann die Schriftgröße oder die Schriftstärke angepasst werden, um anzuzeigen, dass es sich um einen bestimmten Eintragstypen handelt (siehe Abbildung 4.7).

4.2.4 Konsistenzprüfung

Konsistenzprüfung auf Nutzerseite, z.B. dass eine untergeordnete ORI nicht später fällig sein darf, als die übergeordnete, ist ein weiterer Aspekt der die Benutzerfreundlichkeit der pC-App erhöht. So müssen Formulare zum erstellen oder ändern einer ORI oder CLI nicht erst abgeschickt werden, um darüber informiert zu werden, dass Daten falsch eingetragen wurden oder gar fehlen. Stattdessen kann durch eine entsprechende Konsistenzprüfung direkt per Markierung des entsprechenden Feldes einen Hinweis darauf geben werden (siehe Abbildung 4.8).

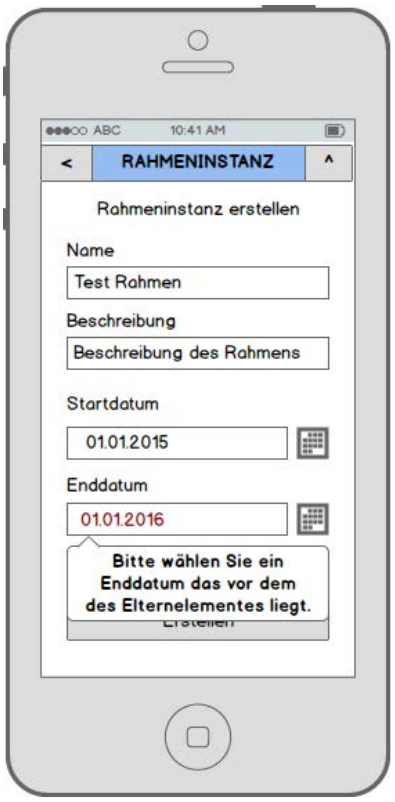


Abbildung 4.8: Markierung von falsch eingetragenen Daten beim Erstellen eines organisatorischen Rahmen

4.2.5 Integration von Positionsdaten

Um Wissensarbeitern anzuzeigen, dass sie sich in der Nähe einer zu erledigenden Aufgabe befinden, soll die pC-App um Funktionen zur Verwendung von Positionsdaten erweitert werden (siehe Abschnitt 3.2). Dies erlaubt es bestimmte Aufgaben mit der derzeitigen Postion zu verknüpfen. Dies soll auch anhand von gegebenen Adressen funktionieren, da man sich nicht immer am gewünschten Standort befindet. Durch diese Funktion sollen Wissensarbeiter bei der Optimierung ihrer Arbeitsabläufe unterstützt werden. So können sie beispielsweise, anhand ihrer räumlichen Distanz, besser einschätzen welche Aufgaben zusammen bearbeitet werden können.

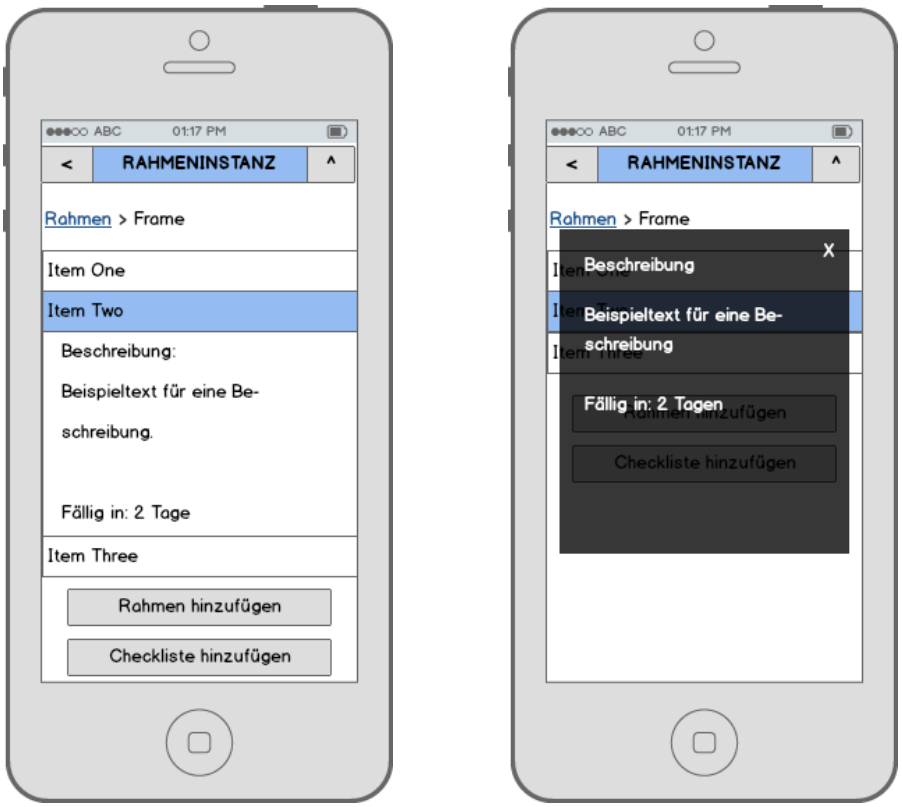
4.2.6 Anzeige der Details von ORI, CLI und CEI

Um die Benutzeroberfläche weiter zu verbessern sollen Details, wie beispielsweise das genaue Fälligkeitsdatum eines ORI, einfacher betrachtet werden können (siehe Abschnitt 3.2). Das Anzeigen von Details eines bestimmten ORI, CLI oder CEI kann über verschiedene Wege erreicht werden. Die Details können entweder auf einer neuen Seite angezeigt werden, die Anzeige der ORI kann vergrößert werden, um alle Informationen anzeigen zu können (siehe Abbildung 4.9a), oder es kann ein Pop-up erstellt werden das die Informationen enthält (Siehe Abbildung 4.9b).

Um diese Elemente anzuzeigen gibt es wiederum verschiedene Optionen. Beispielsweise das Auswählen eines Symbols oder das lange Drücken der gewünschten ORI.

4.2.7 Verbesserte Menüführung

Die Menüführung der pC-App wurde bisher noch nicht optimiert (siehe Abschnitt 3.2). So müssen zum Aufrufen von häufig verwendete Funktionen derzeit oft unnötig viele Aktionen ausgeführt werden (siehe Abbildung 4.10). Ein Menü, das von überall in der pC-App erreicht werden kann, ermöglicht es die Anzahl der benötigten Schritte stark zu verringern, da bis jetzt die meisten Funktionen nur über das Hauptmenü erreichbar sind (siehe Abbildung 4.11). Dadurch wird den Wissensarbeitern unnötiger Arbeitsaufwand erspart.



(a) (b)

Abbildung 4.9: Zwei verschiedene Arten Details einer organisatorischen Rahmeninstanz anzuzeigen

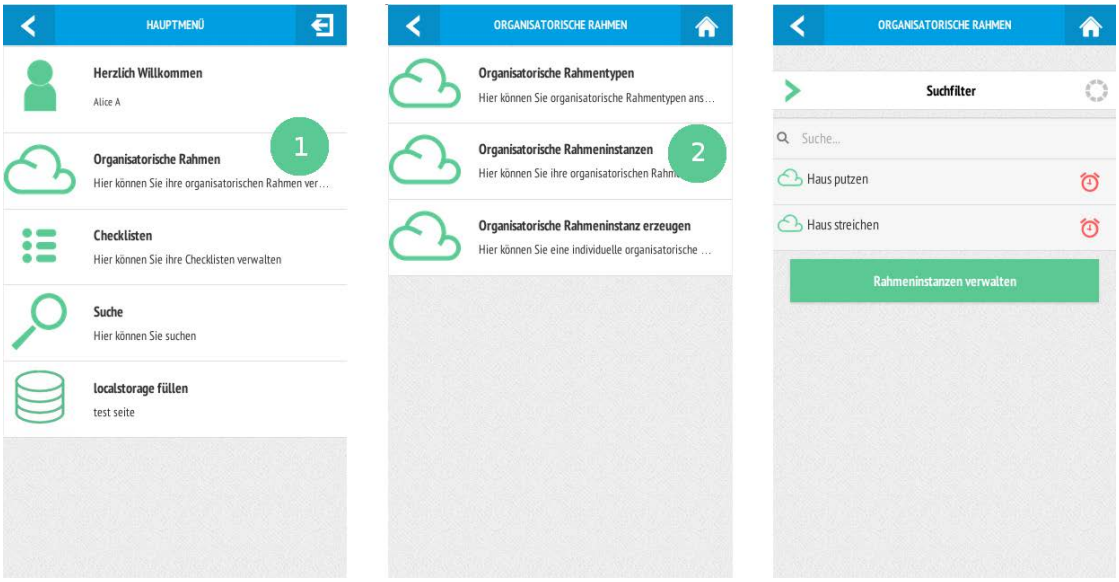


Abbildung 4.10: Schritte zum Anzeigen eines organisatorischen Rahmens

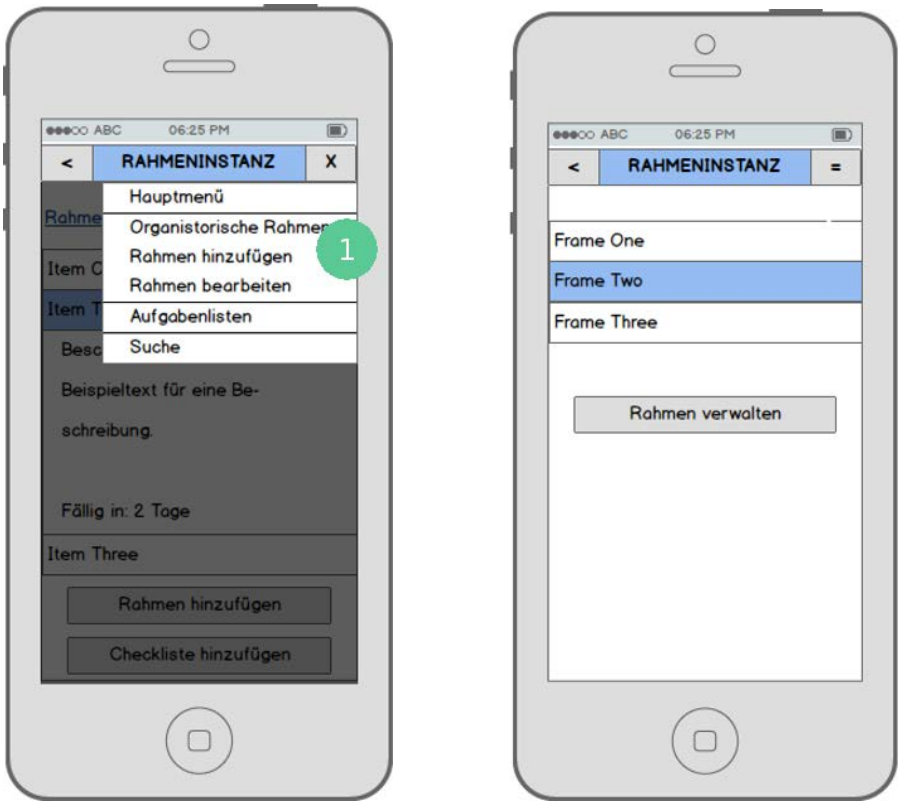
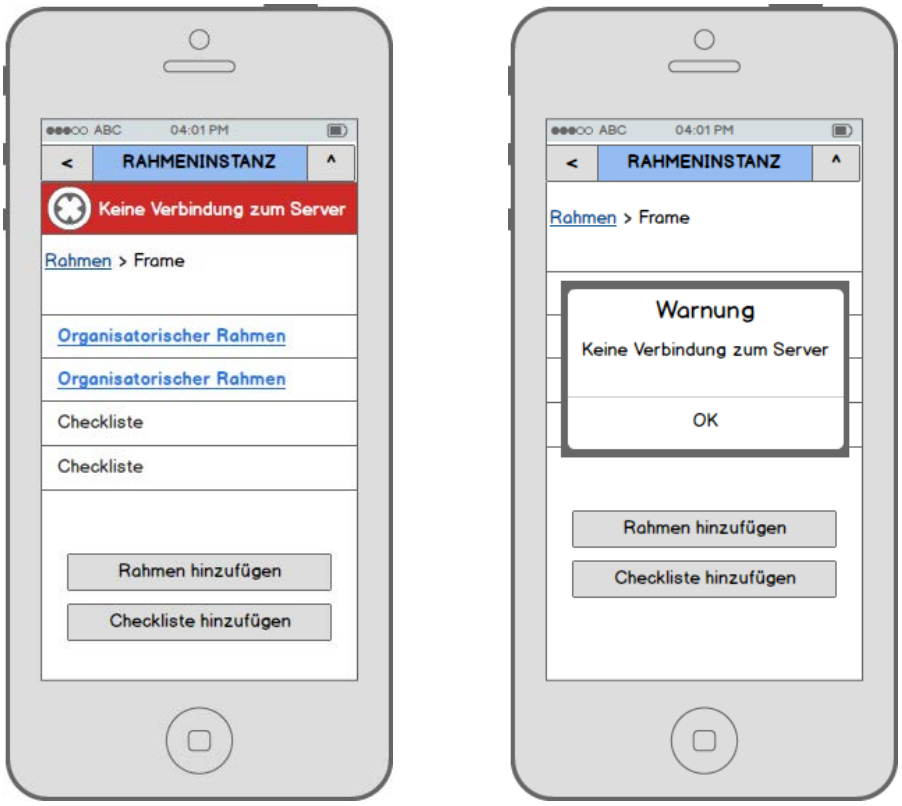


Abbildung 4.11: Anzeigen eines organisatorischen Rahmens mittels Menü

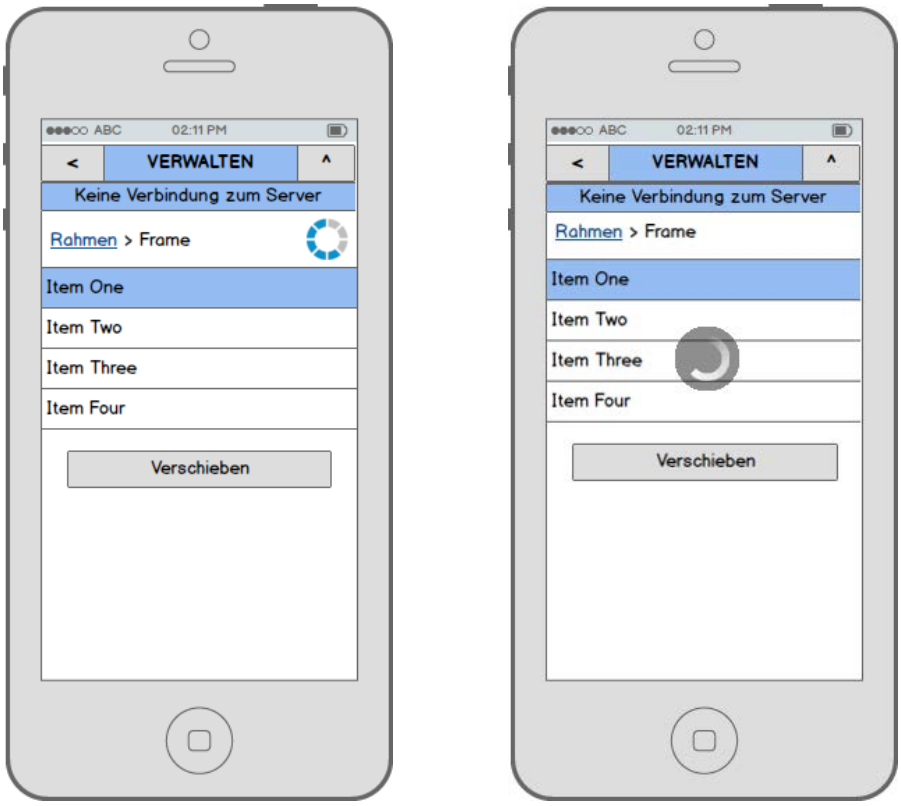


(a) (b)

Abbildung 4.12: Anzeigen von Fehlermeldungen

4.2.8 Fehlermeldungen

Der Wissensarbeiter muss über verschiedene Fehler informiert werden können. Dazu gehört zum Beispiel ein Hinweis, wenn keine Verbindung zum Server aufgebaut werden konnte. Dies könnte beispielsweise durch einfache Pop-ups realisiert werden (siehe Abbildung 4.12b). Allerdings müsste der Wissensarbeiter diese nach jeder Fehlermeldung bestätigen, was bei einer Nutzung im Offlinebetrieb oft vorkommen würde. Ein anderer Ansatz ist das Einblenden einer Informationsleiste. Diese kann zwar leichter übersehen werden, erfordert jedoch keine Interaktion mit dem Nutzer (siehe Abbildung 4.12b).



(a) (b)

Abbildung 4.13: Verschiedene Arten den Nutzer über das laden von Daten zu informieren

4.2.9 Ladeanimationen

Um dem Wissensarbeiter anzuzeigen, dass noch Daten vom Server geladen werden kann eine *Ladeanimation* verwendet werden. Eine mittig platzierte *Ladeanimation* (siehe Abbildung 4.13b), die nach dem Ladevorgang ausgeblendet wird, bietet dabei den Vorteil, dass sie immer gut sichtbar angezeigt werden kann. Eine Ladeanimation in Form eines Symbols, das immer angezeigt wird, bietet dagegen den Vorteil, dass das gleiche Symbol auch für das manuelle Aktualisieren der Daten genutzt werden kann (siehe Abbildung 4.13a).

5

Umsetzung

In diesem Kapitel wird vorgestellt, wie die vorgestellten Lösungen aus Kapitel 4 umgesetzt wurden. Hierfür werden die Frameworks *phoneGap* und *jQuery Mobile* genutzt.

5.1 Verwendete Technologien

Die pC-App wurde mithilfe von zwei wesentlichen Technologien entwickelt. Dabei handelt es sich um *PhoneGap* und *jQuery Mobile*. Diese werden im folgenden Abschnitt näher beschrieben.

jQuery Mobile

jQuery Mobile ist ein auf HTML5 basierendes Framework mit einem Fokus auf *responsive Webdesign*. Es unterstützt alle gängigen Webbrowser, sowie die gängigen mobilen Endgeräte. Es basiert auf den zwei Bibliotheken jQuery und jQuery-ui und unterstützt Entwickler beim Erstellen von Bedienoberflächen für Smartphones und Tablets [JQu15].

PhoneGap

PhoneGap ist ein Framework für die plattformübergreifende Entwicklung von Applikationen für mobile Endgeräte. Es ermöglicht die Anwendungen in JavaScript, HTML5 und CSS3 zu erstellen, ohne auf gerätespezifische Programmiersprachen zurückgreifen zu müssen. Des Weiteren ermöglicht es *PhoneGap*, besser auf Hardwarekomponenten von Smartphones zuzugreifen [Pho15].

5.2 Funktionalität

In diesem Abschnitt wird die konkrete Implementierung der in Kapitel 4 diskutierten Lösungen, für eine bessere Funktionalität des *proCollab* Prototypen, beschrieben.

5.2.1 Verwalten von ORIs

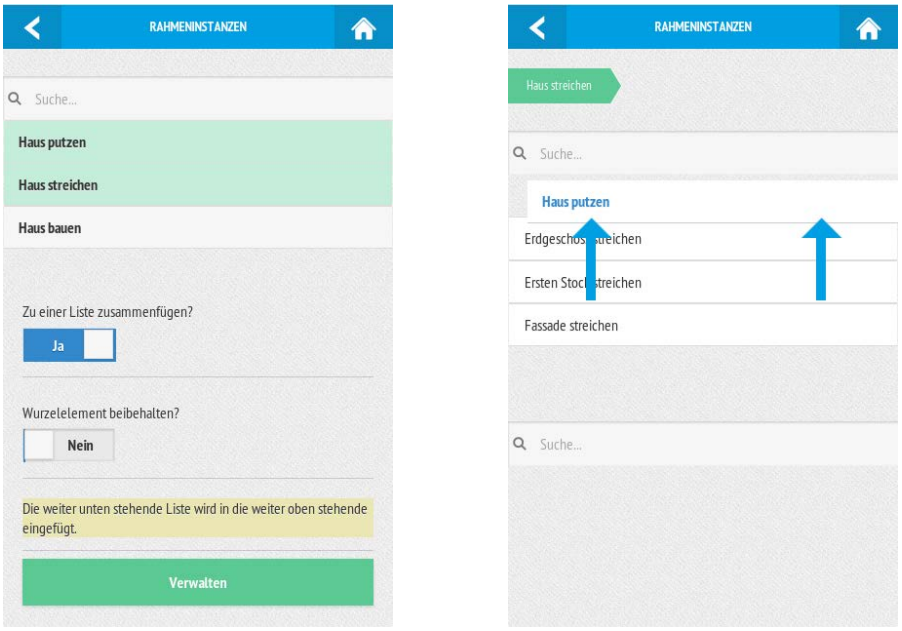
Zur Verwaltung von ORIs wurde der Ansatz von zwei untereinander stehenden ORIs gewählt, in welchen Einträge per Drag and Drop verschoben werden können (siehe Abschnitt 4.1.1). Dazu muss auf einer ersten Seite ausgewählt werden, ob ein oder zwei ORIs verwaltet werden sollen (siehe Abbildung 5.1a). Soll nur ein OR verwaltet werden, wird dieser und eine leere ORI namens Zwischenablage angezeigt. Dies ermöglicht es die Tiefe der gewählten ORI oder CLI, d.h. welcher ORI oder CLI sie untergeordnet ist, zu verändern. Sind zwei ORIs ausgewählt worden erscheinen weitere Optionen (siehe Abbildung 5.1a). Diese erlauben es dem Wissensarbeiter auszuwählen, ob die ORIs zu einer ORI zusammengeführt werden sollen und wenn ja, ob das Wurzelelement dabei erhalten bleiben soll. Nachdem die gewünschten Optionen ausgewählt wurden, werden die beiden ORIs untereinander angezeigt und die Einträge können verschoben werden (siehe Abbildung 5.1b).

Ist das Verschieben eines Eintrages nicht möglich, erscheint ein Dialog, der den Wissensarbeiter auf den aufgetretenen Konflikt aufmerksam macht.

Da ORs und CLs eine ähnliche Datenstruktur besitzen kann der gewählte Ansatz in Zukunft auch noch für das Verwalten von CLs implementiert werden.

5.2.2 Push Notifications

Push Notifications erlauben es einer Applikation Daten zu empfangen, ohne dass eine ständige Verbindung mit dem Server dazu nötig ist. Dadurch können Wissensarbeiter beispielsweise über Änderungen in einem organisatorischen Rahmen informiert werden. Da es sich bei Wissensarbeiten oft um stark dynamische Abläufe handelt (siehe Abschnitt 2.1), ist dies wichtig, um die Übersicht zu behalten. Google bietet für die Realisierung von



(a) (b)

Abbildung 5.1: Zusammenfügen von zwei organisatorischen Rahmen

Push Notifications auf der Androidplattform das sogenannte *Google Cloud Messaging* (GCM) an [Goo15a]. Eine Implementierung von GCM besteht aus dem Server der Applikation, dem GMC Connection Server und der Applikation selbst (siehe Abbildung 5.2). Die einzelnen Komponenten interagieren dabei wie folgt:

- Google bietet den GCM Connection Server an, der Nachrichten vom Server der Applikation, in unserem Fall dem Server des *proCollab* Prototypen, empfängt und an die gewünschten Benutzer der Applikation weiterleitet.
- Jede Installation der Client Applikation erhält hierzu vom Google Connection Server eine eindeutige Registration-ID, anhand derer sie angesprochen werden kann.
- Die Client Applikation übergibt diese Registration-ID, dem Server der Applikation, der diese dann dem entsprechenden Nutzer zuordnen kann.

Zur Implementierung auf Clientseite wird hierfür das *PushPlugin* verwendet, welches Phonegap Anwendungen, direkt erlaubt, Push Notifications zu empfangen [Eas15].

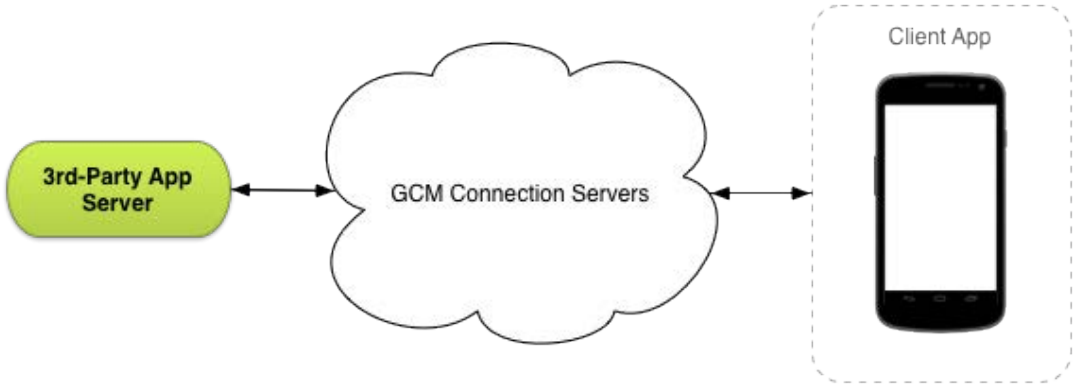


Abbildung 5.2: GMC-Architektur

5.2.3 Offlinebetrieb

Der Offlinebetrieb wurde jeweils mit dem zweiten Lösungsansatz aus Abschnitt 4.1.3 realisiert. Dies bedeutet, dass es nicht möglich ist, Änderungen vorzunehmen, solange keine Verbindung mit dem Server besteht. Dies hat den Vorteil, dass die Konfliktlösung um ein vielfaches einfacher zu handhaben ist. Das Anzeigen von Daten wurde so gelöst, dass zuerst die Daten aus dem Zwischenspeicher angezeigt werden. Gleichzeitig wird überprüft, ob sich Daten auf dem Server inzwischen geändert haben. Ist dies der Fall werden die aktualisierten Daten angezeigt und lokal gespeichert (siehe Abbildung 5.3).

5.2.4 Personalisierung

Die Personalisierung wurde mithilfe einer Einstellungen-Seite in der Benutzerverwaltung gelöst. Hier kann ein Benutzer einstellen, ab welchem räumlichen Abstand zu einer Aufgabe eine Markierung, in Form eines Symbols, angezeigt wird, wie lange es dauert bis der Zwischenspeicher als veraltet angesehen wird und in welchen Zeitabständen die verschiedenen Alarm-Symbole angezeigt werden. Diese Einstellungen werden lokal gespeichert und können so, nach den Wünschen des jeweiligen Benutzers, eingestellt werden (siehe Abbildung 5.4).

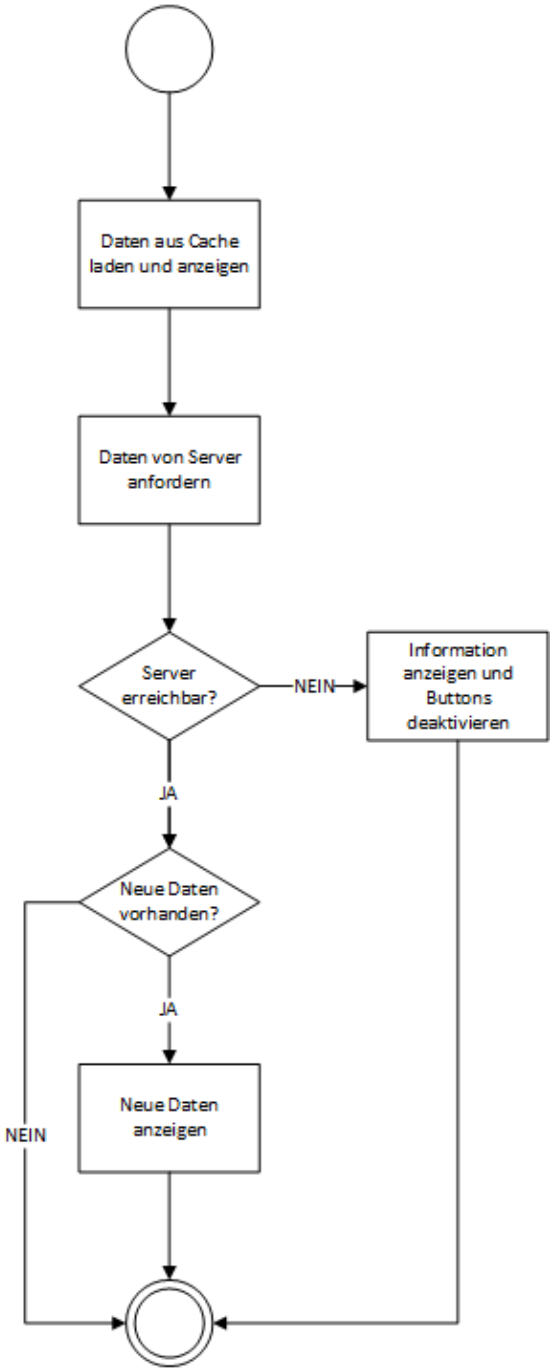


Abbildung 5.3: Ablaufsdiagramm zum Laden von Daten von Zwischenspeicher und Server

5.2.5 Mehrsprachige Benutzeroberfläche

Die mehrsprachige Benutzeroberfläche wurde mit Hilfe von Sprachdateien realisiert. Dabei wird beim ersten Start der pC-App überprüft in welcher Sprache das Betriebssystem des Smartphones eingestellt ist. In dieser Sprache wird die pC-App dann entsprechend angezeigt. Derzeit werden nur die Sprachen Deutsch und Englisch angeboten. Sollte das Betriebssystem eine andere Sprache haben, wird standardmäßig die Sprache auf Englisch gestellt, da Englisch weiter verbreitet ist als die deutsche Sprache. In der Benutzerverwaltung hat der Nutzer die Möglichkeit, die Sprache jederzeit umzustellen (siehe Abbildung 5.4).

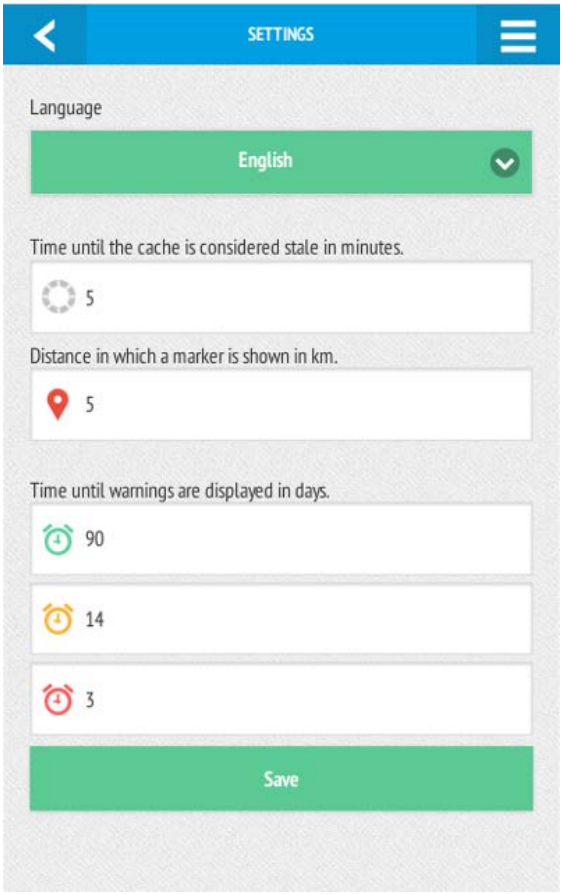


Abbildung 5.4: Einstellungsmöglichkeiten der pC-App

Technisch wurde die mehrsprachige Benutzeroberfläche so realisiert, dass bevor eine Seite angezeigt wird, der Text von HTML-Elementen, die eine Klasse namens `ln_*` besitzen, durch den entsprechenden Text aus einer Sprachdatei ersetzt werden (siehe Abbildung 5.5).

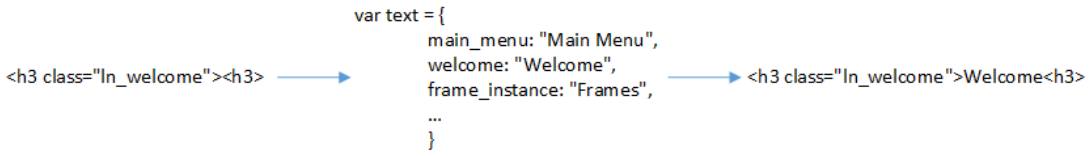


Abbildung 5.5: Laden von Text aus einer Sprachdatei

5.3 Benutzerfreundlichkeit

In diesem Abschnitt wird näher darauf eingegangen, welche Schritte unternommen wurden, um die Benutzerfreundlichkeit der pC-App weiter zu verbessern.

5.3.1 Position in ORIs und CLIs

Um zu verhindern, dass Wissensarbeiter immer wieder an die gleiche Position in einer ORI oder CLI navigieren müssen wird die zuletzt besuchte Position gespeichert. Im folgenden wird diese als Zielknoten bezeichnet. Eine ORI oder CLI kann als ein einfacher Baum dargestellt werden (siehe Abschnitt 4.2.1). Da die Laufzeit eines Algorithmus, der den Pfad vom Wurzel- zum Zielknoten berechnet, bei der Größe der ORIs vernachlässigt werden kann, wurde dieser Ansatz gewählt. Dies hat den in Abschnitt 4.2.1 beschriebenen Vorteil, dass Änderungen im Baum nicht auch im Pfad geändert werden müssen. Da es sich bei den Bäumen um unidirektionale, ungewichtete Graphen handelt, kommen grundsätzlich zwei Algorithmen in Betracht. Die Breiten- und die Tiefensuche. Beide Algorithmen haben eine ähnliche Laufzeit, die mit $\mathcal{O}(|E|)$ abgeschätzt werden kann, wobei $|E|$ für die Anzahl der Kanten im Graph steht. Da bei der Breitensuche jede Ebene nacheinander durchsucht wird, wurde diese gewählt, da nicht davon auszugehen

ist, dass der Zielknoten sich stets in den unteren Regionen des Baums befindet. Die Umsetzung dieses Ansatzes ist in Listing 5.1 gezeigt.

```
1  var queue = [];
2  queue.push(frames);
3  while(queue.length > 0){
4    var current = queue.shift();
5    /* Wenn Zielknoten erreicht: baue Pfad zum Wurzelknoten */
6    if(current["id"] == target){
7      var j = current["id"];
8      var path = [];
9      while(j != root){
10       path.push(j);
11       j = parents[j];
12     }
13     path.push(parseInt(root));
14     var res = [];
15     res.push(path);
16     res.push(current);
17     return res;
18   }
19   /* Derzeitiger Knoten nicht Zielknoten, fuege Kinder des Knotens zur
         Queue hinzu */
20   for(var i = 0; i < current["children"].length; i++){
21     parents[current["children"][i]["id"]] = current["id"];
22     queue.push(current["children"][i]);
23   }
24 }
```

Listing 5.1: JavaScript Code zur Bestimmung des Pfades

Die Variable `frames` beinhaltet dabei den gesamten organisatorischen Rahmen. Dieser wird in einer Warteschlange gespeichert. Nun wird festgestellt, ob sich die `ID` des Zielknoten unter den Kindern des ersten Knoten der Warteschlange befindet. Ist dies nicht der Fall wird die Warteschlange um alle Kinder des derzeitigen Knoten erweitert. Dies wird wiederholt bis es keine Kinder mehr gibt oder der gewünschte Knoten gefunden wurde. Wurde der gewünschte Knoten gefunden, werden die `IDs`, die von diesem bis zum Wurzelknoten benötigt werden in einem Tupel zurückgegeben. Sollte der Knoten

nicht gefunden werden wird nur die `ID` des Wurzelknoten zurückgegeben, sodass dieser angezeigt wird.

5.3.2 Berechnung des Abstandes zu einem Zielort

Um einen Wissensarbeiter zu informieren, dass er sich in der Nähe einer Aufgabe befindet (siehe Abschnitt 4.2.5), muss der Abstand zwischen den Positionsdaten des Wissensarbeiter und den Positionsdaten die in der jeweiligen ORI, CLI oder CEI gespeichert sind, berechnet werden. Die Positionsdaten, die mit einer Aufgabe verknüpft werden sollen, werden dabei anhand einer gegebenen Adresse bestimmt. Um die Positionsdaten anhand dieser Adresse bestimmen zu können, muss auf einen Service von Drittanbietern zurückgegriffen werden. Allerdings haben große Anbieter wie Google Maps oder Bing Maps Beschränkungen in ihren AGBs, was das Zwischenspeichern und Darstellen der Daten betrifft [Goo15b, Mic15]. Kleinere Anbieter wie OpenStreetMap's Nominatim oder OpenCage haben großzügigere AGBs, was das Zwischenspeichern und Darstellen von Daten betrifft [Lok15]. Aus diesen gründen wird für die pC-App der Service von OpenCage verwendet. Die Positionsdaten sind dabei als geographischen Koordinaten (geographische Breite und geographische Länge) gespeichert.

Der Abstand zwischen zwei geographischen Koordinaten kann mit der Haversine Formel berechnet werden [WNT05]. Dabei sei $R = 6371$ der Radius der Erde und zwei Koordinaten in Längen und Breitengrad haben die Namen `[lat1, lon1]` und `[lat2, lon2]`. Dann kann die Distanz d zwischen den zwei Punkten mit dem Code in Listing 5.2 berechnet werden.

```
1 dlon = lon2 - lon1;
2 dlat = lat2 - lat1;
3 a = (sin(dlat/2)) ^2 + cos(lat1) * cos(lat2) * (sin(dlon/2))^2;
4 c = 2 * atan2(sqrt(a), sqrt(1-a));
5 d = R * c;
```

Listing 5.2: Pseudocode zur Berechnung der Distanz zwischen zwei Koordinaten

Durch diese Formel und der HTML5-Funktion zur Bestimmung des derzeitigen Standortes [W3C15], kann nun berechnet werden, ob man sich in einem bestimmten Abstand

zu einer zu erledigenden Aufgabe befindet. Ist dies der Fall, kann ein entsprechendes Symbol gesetzt werden (siehe Abbildung 5.6).

5.3.3 Priorität und Fälligkeit

Symbole sind intuitiver, da farbliche Markierungen zusätzlich eine Legende benötigen. Da zudem auch mehrere Symbole gleichzeitig ohne Konflikte angezeigt werden können, wurde diese Lösung realisiert. Abbildung 5.6 zeigt die Umsetzung des Mockups 4.6. Dabei zeigt ein Wecker in den Farben grün, orange oder rot an, welche Aufgaben in der nächsten Zeit fällig sind. Die Anzahl der Pfeile, ihre Farbe und in welche Richtung sie zeigen signalisiert die Priorität des Eintrages. Grau und nach unten bedeutet dabei geringe Priorität, grün und nach oben hohe Priorität. Die Symbole werden in `SPAN` Elementen angezeigt, die je nach Inhalt, eine passende CSS-Klasse erhalten. Entsprechend dieser werden die passenden Symbole angezeigt. Wählt man ein Symbol aus, erscheint ein Pop-up mit weiteren Details für den Eintrag. Dazu gehören seine Priorität, das Fälligkeitsdatum und eine Beschreibung (siehe Abbildung 5.9).

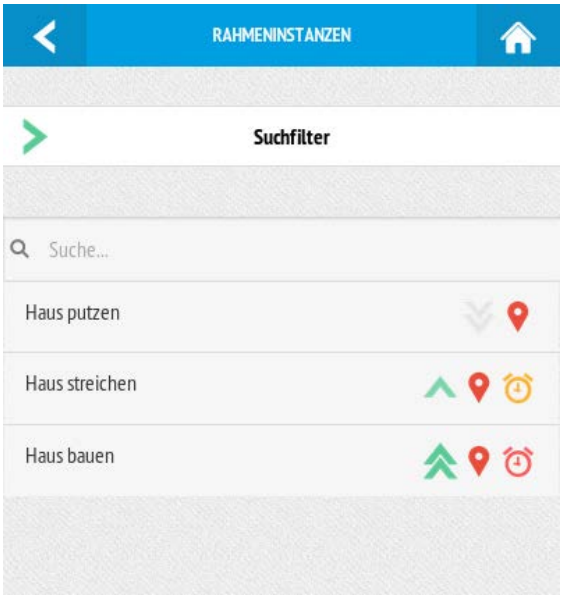


Abbildung 5.6: Anzeige von Priorität, Fälligkeit und örtlicher Nähe mittels Symbolen

5.3.4 Unterscheidung verschiedener Eintragstypen

Die Unterscheidung verschiedener Eintargstypen wurde mithilfe von Symbolen und unterschiedlicher Schriftfarbe und Schriftstärke realisiert. Um eine konsistente Benutzeroberfläche zu erhalten, wurden hierzu die selben Grafiken verwendet, wie im Hauptmenü der pC-App (siehe Abbildung 5.7).

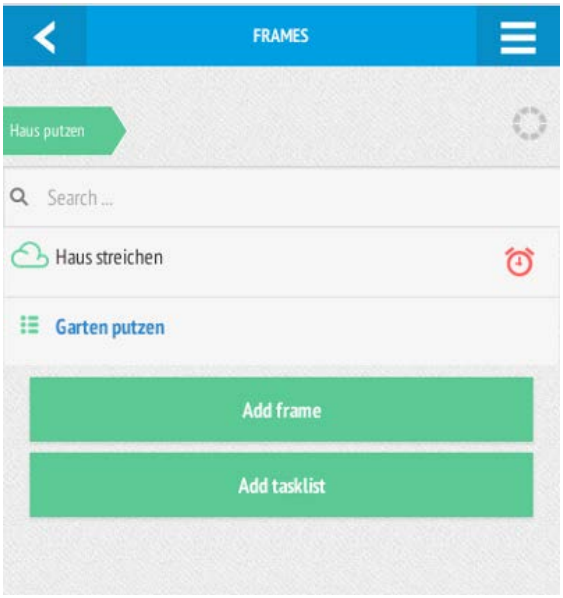
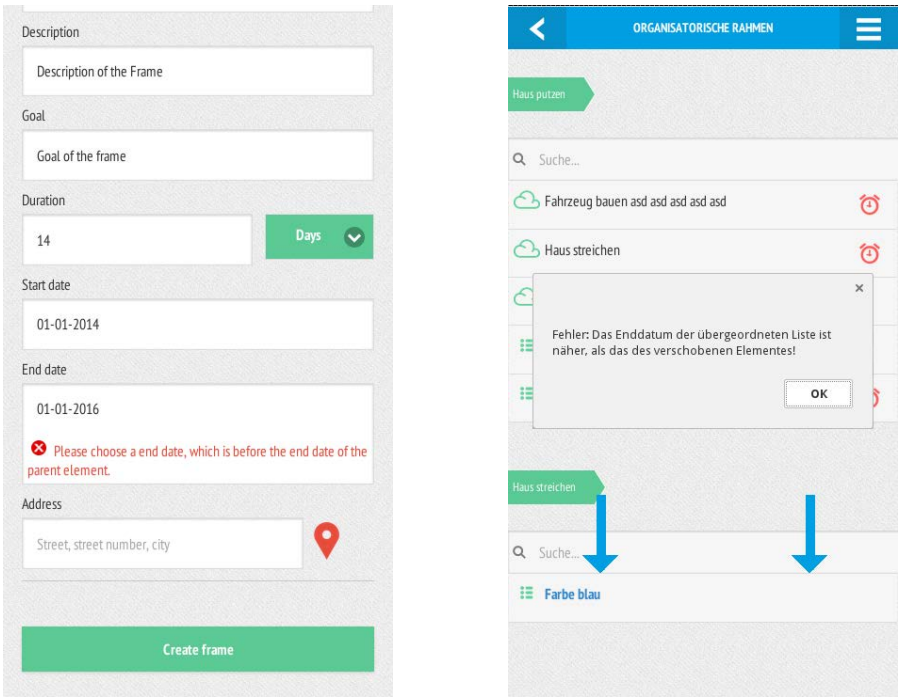


Abbildung 5.7: Anzeige einer organisatorischen Rahmeninstanz und einer Checklisteninstanz

5.3.5 Konsistenzprüfung

Um zu verhindern, dass der Nutzer inkonsistente Daten an den Server sendet, wurde die Konsistenz- und Fehlerprüfung auf Nutzerseite weiter verbessert. Dies wurde mithilfe des *jQuery Validation Plugin* und selbst geschriebenen Methoden realisiert. So kann ein Nutzer nun keine ORIs und CLIs in einen bestehenden OR einfügen, sollte dieser ein früheres Enddatum haben als das einzufügende Element (siehe Abbildung 5.8a). Diese Konsistenzprüfung wird auch ausgeführt, falls schon bestehende ORIs verwaltet werden (siehe Abbildung 5.8b).



(a) (b)

Abbildung 5.8: Anzeige verschiedener Fehlermeldungen

5.3.6 Anzeige der Details von ORI, CLI und CEI

Das Anzeigen von Detail eines ORI, CLI oder CEI wurde mit den beiden in Abschnitt 4.2.6 vorgestellten Ansätzen realisiert. Durch langes Drücken eines ORI, CLI oder CEI kann eine Beschreibung für die gewählte Instanz angezeigt werden. Wählt man ein Symbol aus, erhält man zusätzlich zu der Beschreibung weitere Informationen in einem Pop-up. Dazu gehört das Fälligkeitsdatum, die Priorität und die Adresse des Eintrages (siehe Abbildung 5.9).

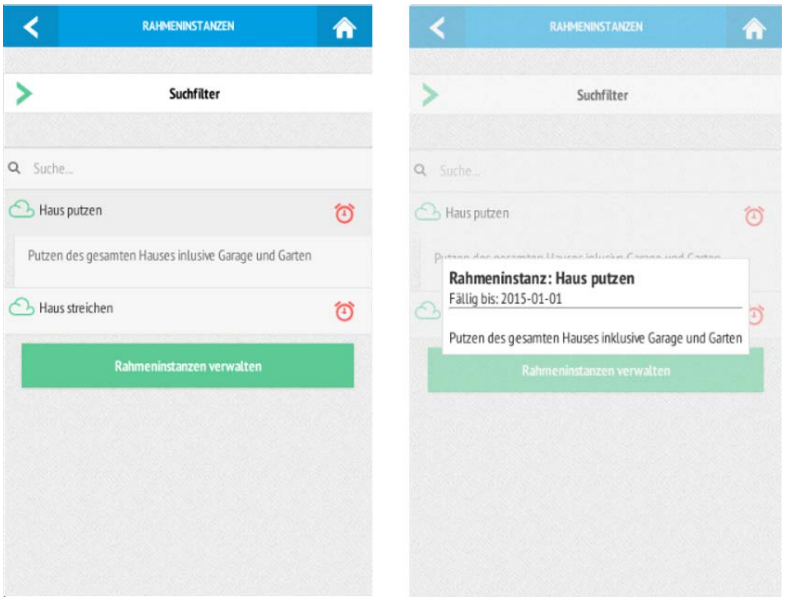


Abbildung 5.9: Anzeigen von Details einer Rahmeninstanz

5.4 Menüführung

Die Menüführung wurde überarbeitet. Bisher mussten fast alle Funktionen aus dem Hauptmenü der pC-App aufgerufen werden. Die meist genutzten Funktionen sind nun auch über eine Navigation erreichbar, was ein schnelleres und angenehmeres Navigieren ermöglicht (siehe Abbildung 5.10).

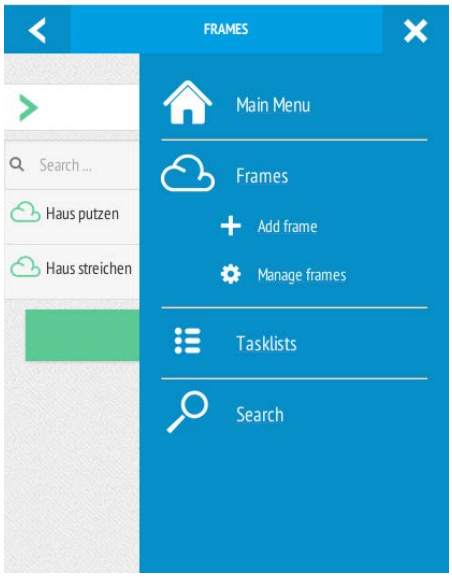


Abbildung 5.10: Neue Navigation des *proCollab* Prototypen

5.5 Fehlermeldungen

Die auftretenden Fehler werden in einer Informationsleiste angezeigt, die eine dem Fehler entsprechende Nachricht anzeigt. Dabei können die für den Nutzer nichtssagenden Fehlercodes in für den Nutzer verständliche Wörter gebracht werden.

6

Fazit

Dieses Kapitel bietet in Abschnitt 6.1 einen Überblick über die vorgestellten Inhalte der Arbeit und in Abschnitt 6.2 einen Ausblick über mögliche Erweiterungen des *proCollab* Prototyp.

6.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde der *proCollab* Prototyp untersucht und verbessert. Dazu wurden in **Kapitel 2** die notwendigen Begriffe definiert und durch ein Fallbeispiel eine Einführung in die Thematik gegeben. Dieses verdeutlicht die Notwendigkeit eines kollaborativen Checklisten-Managements bei wissensintensiven Arbeiten.

In **Kapitel 3** werden anschließend verschiedene Applikationen, die das Ziel haben Wissensarbeiter zu unterstützen untersucht und anschließend mit dem *proCollab* Prototypen verglichen. Anhand der dabei gemachten Beobachtungen und einer Ist-Analyse wurde dann der gewünschte Soll-Zustand beschrieben.

Daraufhin werden in **Kapitel 4** verschiedene Lösungsansätze diskutiert, die zum gewünschten Soll-Zustand führen können. Die verschiedenen Lösungsansätze wurden mit Mockups erweitert, die einen Eindruck für die möglichen Implementierungen bieten. Dabei wird detailliert auf die Vor- und Nachteile der möglichen Implementierungen eingegangen.

In **Kapitel 5** folgt schließlich die Auswahl der am besten geeigneten Lösungen aus Kapitel 4 und eine Beschreibung der tatsächlichen Implementierung anhand von Bildschirmfotos und Codebeispielen.

6.2 Ausblick

Der Prototyp wurde in diese Arbeit um diverse wichtige Funktionen erweitert, die das Checklistenmanagement auf mobilen Geräten weiter verbessern. Es gibt allerdings noch einige Aspekte, die noch nicht realisiert wurden oder noch verbessert werden können. So wurde zum Beispiel bereits der Grundstein für eine vollkommene Offlinenutzung der Applikation gelegt. Es ist jedoch während der Offlinenutzung noch nicht möglich, Änderungen zu tätigen, die zwischengespeichert werden und dann sobald eine Verbindung zum Server besteht angewandt werden. Um dies zu ermöglichen, könnte die Applikation auf Nutzer- als auch auf Serverseite um, wichtige Methoden zur Konfliktlösung erweitert werden.

Eine weitere mögliche Erweiterung wäre das Hinzufügen einer Kommentar- oder Chat-Funktion mit deren Hilfe Wissensarbeiter zusätzlich kommunizieren können.

Möglicherweise könnte auch eine Funktion angeboten werden, die es erlaubt, Dateien wie PDF-Formulare oder Textdateien, mit bestimmten Aufgaben zu verknüpfen.

Das Rollenmanagement kann in Zukunft noch weiter verfeinert werden, um den speziellen Anforderungen von verschiedenen Wissensarbeiten besser gerecht zu werden.

Der so erweiterte Prototyp von *proCollab* stellt einen weiteren Schritt in der Unterstützung von Wissensarbeitern dar. Wie gut sich dieser Checklistenansatz, auf mobilen Endgeräten, für das verwalten von Wissensarbeiten eignet, wird die weitere Entwicklung des Projekts *proCollab* zeigen.

Abbildungsverzeichnis

Literaturverzeichnis

[Any15] ANY.DO: *ANY.DO*. http://www.any.do/. Version: 2015, Abruf: 02.06.2015

[Bre02] BREWSTER, Stephen: Overcoming the lack of screen space on mobile computers. In: *Personal and Ubiquitous Computing* 6 (2002), Nr. 3, S. 188–205

[DJB96] DAVENPORT, Thomas ; JARVENPAA, Sirkka ; BEERS, Michael: Improving knowledge work processes. In: *Sloan management review* 37 (1996), S. 53–65

[DMR15] DI CICCIO, Claudio ; MARRELLA, Andrea ; RUSSO, Alessandro: Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches. In: *Journal on Data Semantics* 4 (2015), Nr. 1, S. 29–57

[DW93] DEGANI, Asaf ; WIENER, Earl: Cockpit Checklists : Concepts, Design, and Use. In: *Human Factors* 35 (1993), Nr. 2, S. 345–359

[Eas15] EASTERDAY, Bob: *Cordova Push Notifications Plugin*. https://github.com/phonegap-build/PushPlugin. Version: 2015, Abruf: 05.03.2015

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

[Goo15a] GOOGLE: *Google Cloud Messaging*. https://developer.android.com/google/gcm/gcm.html. Version: 2015, Abruf: 04.03.2015

[Goo15b] GOOGLE: *Google Maps/Google Earth APIs Terms of Service*. https://developers.google.com/maps/terms. Version: 2015, Abruf: 01.03.2015

[JQu15] JQUERY: *About | jQuery Mobile*. https://jquerymobile.com/about/. Version: 2015, Abruf: 01.06.2015

[Koe13] KOELL, Andreas: *Konzeption und Enwicklung einer auf Tablets optimierten mobilen Anwendung für kollaboratives Checklisten-Management*, Bachelorarbeit, 2013

[Kow11] KOWALEWSKI, Julia: Specialization and employment development in Germany: An analysis at the regional level. In: *Papers in Regional Science* 90 (2011), Nr. 4, S. 789–811

[KS09] KOWALEWSKI, Julia ; STILLER, Silvia: Strukturwandel im deutschen Verarbeitenden Gewerbe. In: *Wirtschaftsdienst* 89 (2009), Nr. 8, S. 548–555

[Lok15] LOKKU: *OpenCage Geocoder Frequently Asked Questions*. `http://geocoder.opencagedata.com/faq.html`. Version: 2015, Abruf: 01.03.2015

[LR07] LENZ, Richard ; REICHERT, Manfred: IT support for healthcare processes - premises, challenges, perspectives. In: *Data and Knowledge Engineering* 61 (2007), Nr. 1, S. 39–58

[Mic15] MICROSOFT: *Microsoft® Bing™ Maps Platform APIs' Terms Of Use*. `http://www.microsoft.com/maps/product/terms.html`. Version: 2015, Abruf: 01.03.2015

[MKR13] MUNDBROD, Nicolas ; KOLB, Jens ; REICHERT, Manfred: Towards a system support of collaborative knowledge work. In: *Business Process Management Workshops*, 2013, S. 31–42

[MR14] MUNDBROD, Nicolas ; REICHERT, Manfred: Process-Aware Task Management Support for Knowledge-Intensive Business Processes : Findings , Challenges , Requirements. In: *IEEE 18th Int'l Distributed Object Computing Conference - Workshops and Demonstrations (EDOCW 2014)*, IEEE Computer Society Press, 2014

[Pho15] PHONEGAP: *PhoneGap | FAQs*. `http://phonegap.com/about/faq/`. Version: 2015, Abruf: 02.06.2015

[Pro15] PROCOLLAB: *Process-aware Support for Collaborative Knowledge Workers*. http://www.uni-ulm.de/in/iui-dbis/forschung/projekte/procollab.html. Version: 2015, Abruf: 12.02.2015

[Rei13] REICH, Daniel: *Konzeption und Entwicklung eines Cloud-basierten Persistenz-Systems für kollaboratives Checklisten-Management*, Bachelorarbeit, 2013

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

[Tie10] TIEMANN, Michael: Wissensintensive Berufe. In: *Bundesinstitut für Berufsbildung, Bonn* (2010)

[Tod15] TODOIST: *Todoist*. https://de.todoist.com/. Version: 2015, Abruf: 02.06.2015

[W3C15] W3C: *Geolocation API Specification*. http://www.w3.org/TR/geolocation-API/. Version: 2015, Abruf: 02.06.2015

[Wei15] WEIGELT, Enrico: *Synchronisation und Fehlertoleranz mobiler Clients in einem bestehenden System für kollaboratives Task-Management*, Bachelorarbeit, 2015

[WHD+10] WEISER, Thomas ; HAYNES, Alex ; DZIEKAN, Gerald ; BERRY, William ; LIPSITZ, Stuart ; GAWANDE, Atul u. a.: Effect of a 19-item surgical safety checklist during urgent operations in a global patient population. In: *Annals of surgery* 251 (2010), Nr. 5, S. 976–980

[WNT05] WARURU, Anthony ; NDUATI, Ruth ; TYLLESKÄR, Thorkild: A software tool for creating simulated outbreaks to benchmark surveillance systems. In: *BMC Medical Informatics and Decision Making* 7 (2005), S. 1–7

[WRW12] WALKER, Isabeau ; RESHAMWALLA, Sophie ; WILSON, Iain: Surgical safety checklists: do they improve outcomes? In: *British journal of anaesthesia* (2012), S. aes175

[Wun15] WUNDERLIST: *Wunderlist*. `https://www.wunderlist.com`. Version: 2015, Abruf: 02.06.2015

[Zie13] ZIEGLER, Jule: *Konzeption und Entwicklung eines Cloud-basierten Servers für kollaboratives Checklisten-Management*, Bachelorarbeit, 2013

Name: Matthias Gerber Matrikelnummer: 726161

Erklärung

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

Ulm, den .....

Matthias Gerber