Masterarbeit

Konzeption und Entwicklung einer gelegentlich verbundenen mobilen Applikation für die Datenerfassung in Gesundheitsorganisationen

Michaela Ferk, BSc

Institut für Softwaretechnologie,
Technische Universität Graz

Betreuer: Univ.-Prof.Dipl-Ing.Dr.techn. Franz Wotawa

Graz, 24. Jänner 2013

Diese Seite wurde absichtlich frei gelassen.

Master's Thesis

(This thesis has been written in German language.)

Design and Development of an Occasionally Connected Mobile Application for Data Acquisition in Health Organisations

Michaela Ferk, BSc

Institute for Software Technology,
Graz University of Technology

Supervisor: Univ.-Prof.Dipl-Ing.Dr.techn. Franz Wotawa

Graz, 24 January 2013

Diese Seite wurde absichtlich frei gelassen.

Kurzfassung

Vor dem Hintergrund des aktuell stark expandierenden mHealth-Marktes zeigt diese Arbeit die Entwicklung einer mobilen Anwendung für die Gesundheitsdokumentation von Patienten. Konkret wird die Internationale Klassifikation der Funktionsfähigkeit, Behinderung und Gesundheit (ICF) der Weltgesundheitsorganisation (WHO) für die Android-Plattform implementiert. Es wird gezeigt, dass die Entwicklung mobiler Gesundheitsapplikationen für den professionellen Einsatz von zahlreichen Unsicherheitsfaktoren flankiert wird. Bereits die Entscheidung, welche mobilen Plattformen, bei der heutzutage heterogenen Landschaft an mobilen Betriebssystemen, unterstützt werden, hat große Auswirkungen auf den Erfolg einer Anwendung. Neben technologischen Aspekten sind auch aktuelle Markttrends der Branche zu beachten, um eine hohe Akzeptanz bei den Anwendern zu erreichen. Ebenso muss der Softwareentwicklungsprozess bereits in frühen Phasen an den Anwendern ausgerichtet werden, um ein Produkt zu entwickeln, das den Nutzeranforderungen entspricht. Diese Arbeit zeigt, wie mit Usability-Tests basierend auf Papier-Prototypen eine entsprechende Nutzerorientierung bereits früh im Softwareentwicklungsprozesses realisiert werden kann. Ein weiterer Schlüsselfaktor dieser Arbeit liegt in der Anbindung der mobilen ICF-Anwendung an das Backend. Obwohl im Umfeld professioneller Geschäftsanwendungen SOAP-basierte Webservices weit verbreitet sind, ergreifen wir den Trend der letzten Jahre und zeigen die Integration basierend auf REST-basierten Webservices. Diese Architektur hat uns eine schnelle Entwicklung eines Software-Prototypen unter Vermeidung unnötiger Komplexität ermöglicht, ohne auf grundlegende Sicherheitsmechanismen verzichten zu müssen.

Schlüsselwörter

mobile Gesundheitsapplikation, mHealth, ICF, Webservice, REST, Benutzerfreundlichkeit, Android

ÖSTAT Klassifikation

1140, 1138, 3927, 1157

ACM Klassifikation

D.2, J.3, C.2.4, H.5

Diese Seite wurde absichtlich frei gelassen.

Abstract

Against the background of the rapidly expanding market of mobile health (mHealth) applications this work shows the development of a mobile application regarding the documentation of patients' health states. Specifically, the International Classification of Functioning, Disability and Health (ICF) — introduced by the World Health Organization (WHO) — is implemented on the Android platform. We show that the development of mobile health applications for professional use is accompanied by numerous uncertainties. Even deciding which of the numerous mobile operating systems to support has a huge impact on the success of an application. Besides technological aspects market trends also have to be considered in order to achieve a high rate of user acceptance. Likewise, the software-development process has to be user-centered from early stages on in order to develop a product that properly meets the user requirements. This work shows how usability tests based on paper mockups achieve the necessary user orientation early in the software-development process. Another key factor in this work is the integration of the mobile application with the backend. Although SOAP webservices are widely used for mobile business applications, we follow the recent trend and show the integration based on REST-ful webservices. This architecture allows us to develop a software prototype avoiding unnecessary complexity and without compromising basic security mechanisms.

Keywords

mobile health, mHealth, application, ICF, web service, REST, Usability, Android

ÖSTAT classification

1140, 1138, 3927, 1157

ACM classification

D.2, J.3, C.2.4, H.5

Diese Seite wurde absichtlich frei gelassen.

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, 24. Jänner 2013

Michaela Ferk, BSc

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, 24 January 2013

Michaela Ferk, BSc

Diese Seite wurde absichtlich frei gelassen.

Danksagung

Am Ende eines Studiums erkennt man einmal mehr: Der Weg ist das Ziel. In diesem Sinne möchte ich allen Personen danken, die mich auf dem Weg zu dieser Arbeit begleitet haben. Mein herzlicher Dank gilt meinem Betreuer, Herrn Dr. Peischl, der stets interessante Aspekte und Sichtweisen in die Arbeit einbrachte und sie so in die richtige Richtung lenkte. Ein großes Dankeschön an die Mitarbeiter von FERK Systems, die mein Interesse für das spannende Feld der "mobile Health" geweckt haben. Ein ganz besonderer Dank gilt meinen Eltern, die dieses Studium stets voll unterstützt haben. Weiters möchte ich meinem Freund, Raphael, danken, der in den letzten Monaten große Geduld und Nervenstärke bewiesen hat, sowie meinen Freunden für die entspannte Zeit abseits der Universität.

Michaela Ferk, BSc

Graz, 24. Jänner 2013

Diese Seite wurde absichtlich frei gelassen.

Inhaltsverzeichnis

1 Einleitung und Motivation **15**

1.1 Ausgangssituation 17

1.2 Internationale Klassifikation der Funktionsfähigkeit, Behinderung und Gesundheit 18

1.3 Verwandte Arbeiten 21

1.4 Ziele 23

2 Theoretische Grundlagen **27**

2.1 Erfolgsfaktor Usability 27

2.1.1 Charakeristik mobiler Geräte 27

2.1.2 Methoden des Usability Engineerings 28

2.2 Webservice-Architekturen 33

2.2.1 Einleitung 33

2.2.2 Besondere Anforderungen von Geschäftsanwendungen 34

2.2.3 SOAP-basierte Webservices 35

2.2.4 REST-basierte Webservices 41

3 Analyse und Vergleich mobiler Technologien **51**

3.1 Markttrends bei mobilen Gesundheitsanwendungen 51

3.2 Technologische Charakteristiken 53

3.3 Entscheidungsfindung 58

4 Nutzerzentriertes Design im Softwareentwicklungsprozess **59**

4.1 Vorgehensweise und Methoden 59
4.2 Usability-Tests mit Papier-Prototypen 61
4.2.1 Forschungsfragen 61
4.2.2 Nutzergruppen 62
4.2.3 Testablauf und -aufbau 64
4.2.4 Ergebnisse und Empfehlungen 67

5 Experimentelle Implementierung der ICF-Anwendung 83
5.1 Funktionsumfang 83
5.1.1 Authentifizierung und Synchronisation 83
5.1.2 ICD- und ICF-Datenmanipulation 84
5.1.3 ICF-Auswertung 85
5.2 Software-Architektur 86
5.2.1 REST-basierte Backend-Anbindung 86
5.2.2 Software-Komponenten im Schichtenmodell 88
5.3 Synchronisationsproblematik 91
5.3.1 Typische Anwendungsfälle 100
5.4 Sicherheitsmechanismen 101

6 Zusammenfassung 103

Abbildungsverzeichnis 105

Tabellenverzeichnis 107

Abkürzungsverzeichnis 109

Wissenschaftliche Referenzen 111

Web-Referenzen 117

1. Einleitung und Motivation

Mobile Technologien hielten in den letzten Jahren einen rasanten Einmarsch in unser tägliches Leben. Smartphones und ihre vielfältigen Applikationen sind kaum noch wegzudenken. Während sich die ersten mobilen Anwendungen vor allem auf private Kunden konzentrierten, drängen mittlerweile auch mehr und mehr mobile Geschäftsanwendungen auf den Markt, die für den professionellen Einsatz gedacht sind. Besonders für den Gesundheitsbereich, der durch einen hohen Informationsbedarf in einem dynamischen Umfeld geprägt ist, haben mobile Technologien großes Potential. Die intensive Forschungsarbeit und enorme marktwirtschaftliche Entwicklung dieses Bereichs haben in den letzten Jahren den Begriff der "mobile Health" oder "mHealth" geprägt.

Die vorliegende Arbeit konzentriert sich auf die mobile Unterstützung der Gesundheitsdokumentation von Patienten. Konkret wird hierzu die Internationale Klassifikation der Funktionsfähigkeit, Behinderung und Gesundheit (ICF, engl.: International Classification of Functioning, Disability and Health) als mobile Anwendung umgesetzt. Diese, von der World Health Organization (WHO) publizierte, Klassifikation stellt ein umfassendes System zur Dokumentation des Gesundheitszustandes von Patienten dar.

Die Entwicklung derartiger Anwendungen für den professionellen Einsatz stellt die Softwareentwicklung vor zahlreiche Herausforderungen. Zu Beginn des Entwicklungsprozesses steht die Frage, welche mobile Plattformen unterstützt werden sollen. Bei der momentan heterogenen Landschaft an mobilen Betriebssystemen ist eine Abwägung der Vor- und Nachteile einzelner Technologien unter Berücksichtigung der Markttrends innerhalb der spezifischen Branche von großer Bedeutung. Das zweite Kriterium, speziell bei hochgradig interaktiven Systemen, stellt die Akzeptanz der Mitarbeiter dar. Dieser Faktor wird umso gewichtiger, je belastender

die berufliche Tätigkeit der Mitarbeiter ist. Gerade im Bereich von Gesundheitsorganisationen ist die Belastung oft enorm. Eine hohe Akzeptanz kann nur durch einen Softwareentwicklungsprozess erreicht werden, der sich an den Bedürfnissen der Anwender orientiert, um ein nützliches und benutzerfreundliches System hervorzubringen. Besonders maßgeblich ist die Nutzerorientierung der Softwareentwicklung in Bereichen, die durch unklare und dynamische Anforderungen geprägt sind. Dies ist bei der mobilen Umsetzung der ICF der Fall, da diese Klassifikation in der Praxis bislang keinen breiten Einzug gehalten hat. Somit sind zu Entwicklungsbeginn keine konkreten Daten über den Nutzungskontext oder die Anforderungen in der Praxis bekannt.

Ein weiterer Schlüsselfaktor jeder mobilen Geschäftsanwendung ist das Interaktionskonzept zwischen mobiler Applikation und Backend. Mobile Geschäftsanwendung werden von zahlreichen Qualitäts- und Sicherheitsanforderungen flankiert, die vor allem die Kommunikation bzw. den Datenabgleich der mobilen Endgeräte mit dem zentralen Server betreffen. Webservices stellen in diesem Zusammenhang eine populäre Architekturmöglichkeit dar. Sie erlauben die plattformunabhängige Zusammenarbeit von verteilten Anwendungen über internetbasierte Protokolle. Während über die Jahre hinweg viele Spezifikationen für SOAP-basierte Webservices entstanden, wurde besonders in letzter Zeit die REST-basierte Architektur von Webservices populär. Die Anwendbarkeit beider Architekturen auf unsere mobile Gesundheitsapplikation muss detailliert geprüft werden bevor die Umsetzung einer prototypischen Anwendung initiiert wird.

Die erfolgreiche Implementierung der mobilen Gesundheitsanwendung wird somit von drei wesentlichen Faktoren tangiert. Zu Beginn des Entwicklungsprozesses müssen marktstrategische Überlegungen zur Technologieauswahl angestellt werden. Weiters muss der Softwareentwicklungsprozess die flexible Reaktion auf Rückmeldungen von Nutzern erlauben, um eine benutzerfreundliche Anwendung zu realisieren. Die softwaretechnische Realisierung der Nutzeranforderungen sowie der dahinterstehenden Geschäftslogik und Backend-Interaktion bilden die Basis der Anwendung.

Um den Kontext der vorliegenden Arbeit zu klären, werden in Abschnitt 1.1 und Abschnitt 1.2 die Ausgangssituation der Arbeit sowie Grundlagen zu ICF behandelt. In Abschnitt 1.3 zeigen wir verwandte Arbeiten zu diesem Themengebiet, bevor in Abschnitt 1.4 die Ziele der Arbeit detailliert erläutert werden.

1.1 Ausgangssituation

Die Prozesse des Gesundheitswesens sind vielfältig und können abhängig von der Art der Gesundheitsorganisation stark variieren. Durch den Einsatz mobiler Technologien können diese Prozesse optimal unterstützt bzw. verbessert werden, was sowohl zu einer erhöhten Patientenzufriedenheit als auch zu einer Arbeitszeitersparnis für das Gesundheitspersonal und somit zu einer Kostenersparnis für das Management führt. Eine Arbeit, in der die Einführung mobiler Technologien im Gesundheitsbereich genau diese gewünschten Effekte erzielen konnte, zeigen beispielsweise Holzinger u.a. [20].

Das Grazer IT-Unternehmen FERK Systems will ebenfalls auf mHealth-Technologien setzen. Derzeit wird eine Software entwickelt, die die Prozesse von Gesundheitsorganisationen allumfassend unterstützt - dies reicht von der Pflege- und Therapieplanung bis hin zur Arbeitszeiterfassung der Mitarbeiter. Teilweise soll diese Prozessunterstützung durch mobile Anwendungen erreicht werden. Die Basis für eine mobile Unterstützung wurde im Rahmen dieser Diplomarbeit geschaffen, indem ein Teilprozess als mobile Anwendung umgesetzt wurde. Bei diesem Teilaspekt konzentrierten wir uns auf die Dokumentation des Gesundheitszustandes von Patienten. FERK Systems setzt hier auf die von der World Health Organization (WHO) [91] entwickelte Classification of Functioning, Disability and Health, kurz ICF, die in Abschnitt 1.2 detailliert beschrieben wird.

In der Pflege kann der Einsatz von ICF am sogenannten Pflegeprozess erläutert werden. Abbildung 1.1 zeigt die Phasen des Pflegeprozesses der WHO [3]. Die Dokumentation des Gesundheitszustandes des Patienten spielt in der ersten Phase, der Pflegeanamnese, eine Rolle. Auf der Datenerhebung dieses Schrittes basieren die Pflegeplanung sowie -durchführung. Danach wird abermals der Gesundheitszustand dokumentiert und die Pflege evaluiert, bevor der Kreislauf von Neuem beginnt.

Typischerweise finden alle Phasen des Pflegeprozesses am Bett beziehungsweise im Umfeld des Patienten statt. Mobile Technologien bringen hier eine große Zeit- und Kostenersparnis durch eine optimale Unterstützung der Arbeitsvorgänge. Das Pflegepersonal kann die Patientendaten vor Ort erfassen und die Synchronisation mit dem zentralen Datenbestand erfolgt automatisiert. Weiters fördert die mobile

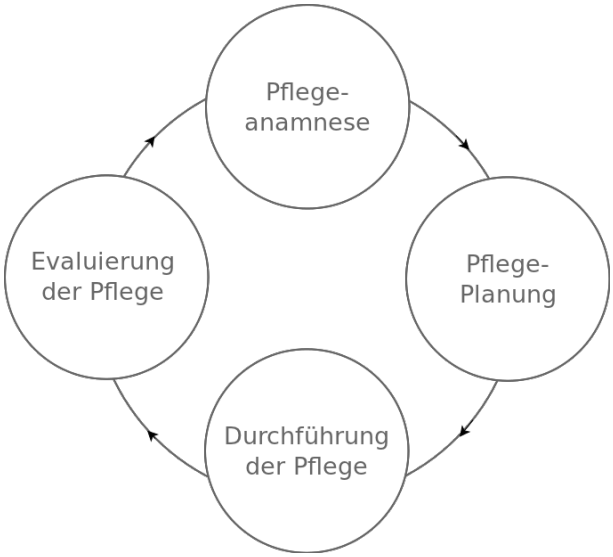

Abbildung 1.1: Pflegeprozess der WHO (vgl. [3]).

IT-Unterstützung die Information und Kommunikation aller beteiligter Berufsgruppen. Die Daten eines Patienten sind im Regelfall schnell allen zuständigen Personen, seien es weitere Pfleger/-innen, Mediziner/-innen oder Physiotherapeuten/-innen, zugänglich. Abbildung 1.2 zeigt die Prozessunterstützung beispielhaft anhand eines abstrahierten UML-Sequenzdiagrammes [46]. Der Krankenpfleger greift direkt am Bett des Patienten auf die zentral gespeicherte Patienteninformation zu. Die Dokumentation des Gesundheitszustandes erfolgt ebenfalls über die mobile Anwendung, wobei die neuen Daten sofort an das Backend-System übermittelt werden. Ab diesem Zeitpunkt erhalten auch andere Akteure die aktuellen Gesundheitsdaten des Patienten, wie hier beispielsweise der zuständige Physiotherapeut.

1.2 Internationale Klassifikation der Funktionsfähigkeit, Behinderung und Gesundheit

Die Dokumentation des Gesundheitszustandes mit der mobilen Anwendung basiert auf der, von der WHO entwickelten, Internationalen Klassifikation der Funktionsfähigkeit, Behinderung und Gesundheit (ICF, engl.: International Classification of Functioning, Disability and Health). Die ICF soll mit der, ebenfalls von der WHO verfassten, internationalen statistischen Klassifikation der Krankheiten (ICD, engl: International Classification of Diseases) kombiniert werden. Als Grundlage für diese

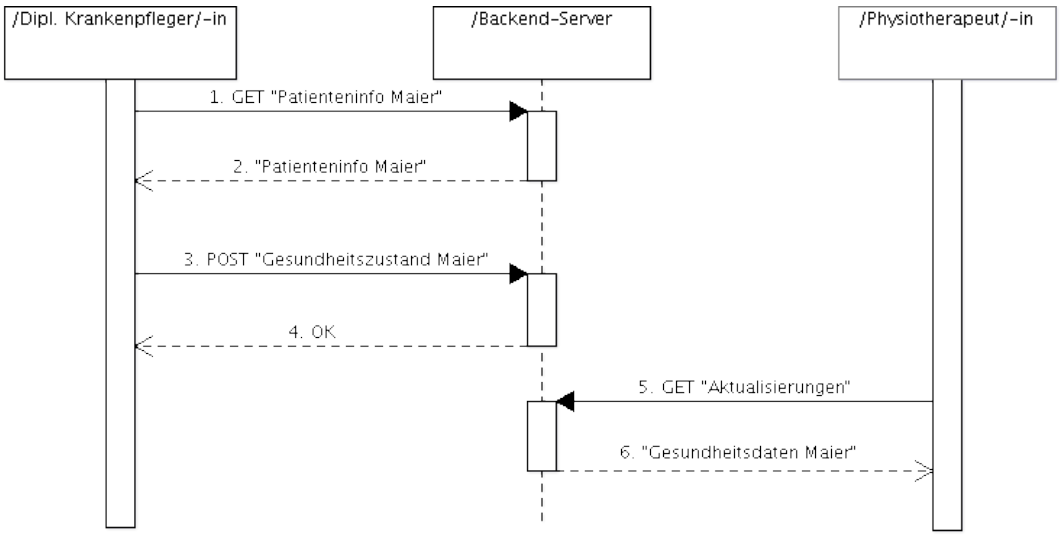

Abbildung 1.2: Beispiel einer mobilen IT-Unterstützung im Pflegeprozess.

Arbeit beschreiben wir in den folgenden Abschnitten das Wesentliche zu den Standards ICD und ICF. Für detaillierte Informationen verweisen wir den interessierten Leser auf [87, 88].

ICF: ICF stellt eine Klassifikation zur umfassenden Beschreibung des Gesundheitszustands einer Person dar. "Umfassend" bedeutet in diesem Zusammenhang, dass der Gesundheitszustand einer Person nicht ausschließlich ausgehend von körperlichen Funktionen oder Störungen beurteilt wird, sondern auch die Möglichkeit einer aktiven Lebensgestaltung sowie Faktoren aus dem Umfeld des Patienten einfließen. Abbildung 1.3 zeigt die einzelnen Komponenten der ICF sowie ihre Wechselwirkungen, wobei die personenbezogenen Faktoren derzeit noch nicht realisiert sind. Jeder Komponente sind sogenannte ICF-Codes zugewiesen, die einzelne Aspekte der Gesundheit, der Fähigkeiten oder Behinderungen eines Menschen betrachten. Die folgende Aufzählung nennt beispielhaft ICF-Codes der einzelnen Komponenten:

- **Körperfunktionen und -strukturen**
  - b210 Sehfunktion
  - b1340 Schlafqualität
- **Aktivitäten und Partizipation**
  - d450 Gehen

  - d490 Körperposition ändern

- **Umweltfaktoren**
  - e310 Familie
  - e460 Soziale Einstellungen

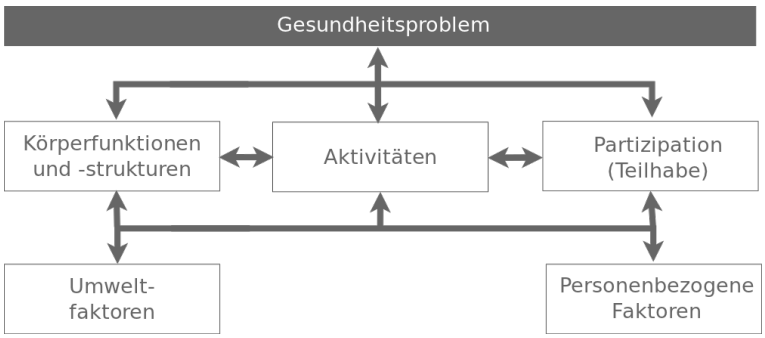

Abbildung 1.3: Komponenten der ICF (vgl. [39]).

Der Aufbau der ICF ähnelt jenem von ICD. Die Beurteilung einzelner ICF-Codes erfolgt in der Regel anhand einer fünfstufigen Werteskala, wobei der Wert mit dem Grad der Beeinträchtigung des Patienten steigt. Für detaillierte Informationen zur Beurteilung und den einzelnen Beurteilungsmerkmalen verweisen wir auf [89].

ICD: Die internationale statistische Klassifikation der Krankheiten (ICD, engl.: International Classification of Diseases) stellt ein Klassifikationssystem für medizinische Diagnosen dar. Jeder Diagnose wird dabei innerhalb eines hierarchischen Systems ein drei- oder vierstelliger Code zugeordnet. Tabelle 1.1 zeigt ein Beispiel für die ICD-Codierung ausgehend vom ICD-Kapitel bis hin zu einer expliziten ICD-Kategorie.

Zusammenhang ICD - ICF: Um aus den tausenden Codes der ICF jene zu dokumentieren, die für einen Patienten relevant sind, muss es standardisierte Sets an ICF-Codes geben. Im Allgemeinen hängt die Relevanz von ICF-Codes von den ICD-Diagnosen eines Patienten ab. Daher wurden sogenannte *ICF-CoreSets* für ICD-Diagnosen bzw. Diagnosegruppen entwickelt. Diese *ICF-CoreSets* enthalten ICF-Codes, die speziell für die ICD-Diagnose(gruppe) des Patienten relevant sind und in regelmäßigen Abständen dokumentiert werden sollen. Aus der Literatur ist uns

Stufe	Code	Bezeichnung
Kapitel	E00–E90	Endokrine, Ernährungs- und Stoffwechselkrankheiten
Gruppe	E10–E14	Diabetes mellitus
Kategorie (3-stellig)	E10.-	Primär insulinabhängiger Diabetes mellitus [Typ-1-Diabetes]
Kategorie (4-stellig)	E10.0-	Primär insulinabhängiger Diabetes mellitus [Typ-1-Diabetes]: Mit Koma

Tabelle 1.1: Beispiel einer ICD-Codierung.

bekannt, dass beispielsweise *ICF-CoreSets* entwickelt wurden, die auf Patienten mit Rückenschmerzen, Osteoporose, Diabetes mellitus, Schmerzstörungen, depressiven Störungen, etc. anwendbar sind [36].

1.3 Verwandte Arbeiten

Die Zahl an Gesundheitsapplikationen und Forschungsarbeiten im mHealth-Bereich wächst stetig. Zahlreiche spezielle Anwendungsgebiete haben sich sowohl für den privaten als auch den professionellen Einsatz herauskristallisiert. So reichen mobile Anwendungen für private Nutzer von Fitness- und Wellness-Applikationen über Anwendungen zur Medikamentenverwaltung bis hin zur Unterstützung bei der Selbstdiagnose. Auch die Vielfalt an mobilen Gesundheitsapplikationen für den professionellen Einsatz ist groß. Das Angebot reicht von Applikationen zur Anzeige medizinischer Bildaufnahmen über Anwendungen zur ICD-Codierung bis hin zu elektronsichen Patientendatensystemen.

Software-Lösungen für den Einsatz von ICF scheinen am Markt allerdings noch rar zu sein. Arbeiten im Zusammenhang mit ICF konzentrieren sich vorrangig auf rein medizinische Aspekte, wobei vor allem der Erstellung von *ICF-CoreSets* Bedeutung zukommt, wie beispielsweise unter [51] ersichtlich ist. Laut Bender [5] stoßen viele Einrichtungen bei der praktischen Anwendung der ICF auf Probleme. Dies begründet sich seiner Meinung nach darauf, dass die Fachliteratur bislang kaum Leitfäden zum praktischen Einsatz der komplexen Klassifikation entwickelt hat.

Die *Rehab-CYCLE* Software der *RehabNet AG* [51] war die erste umfassende, ICF-basierte Desktopanwendung für das Rehabilitations-Management. Die Software

erlaubte die Beschreibung des Gesundheitszustandes von Patienten anhand von ICF sowie das Formulieren von Rehabilitationszielen und die Planung entsprechender Maßnahmen. Unterstützt wurde diese Funktionalität durch einen integrierten ICF-Browser und *ICF-CoreSets* für zahlreiche Krankheitsbilder. Die Software ist in enger Kooperation mit Kliniken und universitären Einrichtungen entstanden und wurde von 2004 bis 2011 erfolgreich in Kliniken und Gesundheitszentren in der Schweiz und in Deutschland eingesetzt. Mittlerweile wurde *Rehab-CYCLE* durch die Web-Applikation *RehabNET-IPS* abgelöst. Nachdem *RehabNET-IPS* über den Browser bedient wird, ist ein Zugriff auch über mobile Endgeräte möglich. Informationen zu einem speziell auf mobile Endgeräte abgestimmten Design sind uns allerdings nicht bekannt.

Als einziges ICF-basiertes Dokumentationssystem im deutschsprachigen Raum, das explizit auch eine Lösung für mobile Endgeräte anbietet, konnten wir das System *ICOsys* von *Management Partners* identifizieren [58, 59]. Die Möglichkeit der ICF-Dokumentation ist dabei in ein Modulsystem eingebettet, das Funktionalitäten von Patienten- und Ressourcenverwaltung bis hin zur Mitarbeiterzeiterfassung anbietet. Als Referenzen werden die *Lebenshilfe Kärnten* [69] und die *assista Soziale Dienste GmbH* [49] angegeben. Die letzte offizielle Information zu *ICOsys.mobil* stammt aus dem Jahr 2008. Zu diesem Zeitpunkt setzte *Management Partners* auf das heute nicht mehr unterstützte Betriebssystem Windows Mobile. Es ist nicht ersichtlich, ob das Unternehmen die Portierung ihres Systems auf eine aktuell unterstützte Plattform plant. Nähere Informationen zum Design der ICF-Dokumentation bzw. zur grundsätzlichen Software-Architektur sind uns nicht bekannt.

Ein weiteres Unternehmen, das die Patientenverwaltung inklusive Dokumentation des Gesundheitszustandes in eine umfassende Administrationssoftware für Pflegedienste integriert, ist der deutsche Marktführer *MediFox* [43]. *MediFox* bietet mobile Softwarelösungen basierend auf der Plattform Windows Phone und dem .Net-Framework an. Das Unternehmen schlägt bei der Dokumentation des Gesundheitszustandes allerdings einen anderen Weg ein. Der Gesundheitszustand von Patienten wird nicht basierend auf ICF, sondern auf diversen Skalen dokumentiert. Als Beispiel kann hier die Braden-Skala genannt werden. Hier besteht ein großer Unterschied zum ICF-basierten Konzept. Jede dieser Skalen bzw. Assessments betrachtet isoliert einen Teil des Gesundheitszustandes anhand eines spezialisierten

Beurteilungssystems. ICF betrachtet den Gesundheitszustand eines Patienten gesamtheitlich und nutzt dazu ein einheitliches, quantifizierbares Beurteilungsschema.

1.4 Ziele

Grundlegendes Ziel der vorliegenden Arbeit ist die Umsetzung der ICF als mobile Anwendung. Dem Gesundheitspersonal soll somit die Möglichkeit der Dokumentation und Auswertung des Gesundheitszustandes direkt am Krankenbett des Patienten gegeben werden. Durch eine Backend-Anbindung zwischen mobilen Endgeräten und zentralem Server soll der Datenabgleich der erfassten Gesundheitsdaten stattfinden. Die Problematik der Aufgabenstellung und die entsprechenden Zielsetzungen verteilen sich auf mehrere Ebenen, die wir im Laufe des Softwareentwicklungsprozesses bearbeiten.

Die erste Ebene zu Beginn des Entwicklungsprozesses betrifft die Einbeziehung technischer und marktpolitischer Überlegungen in die Entscheidung über die zu unterstützenden mobilen Plattformen. Die Entscheidung für eine bestimmte Technologie sollte bei der Einführung eines kommerziellen Produktes auch wesentlich durch Trends innerhalb einer bestimmten Branche geprägt sein, um eine möglichst hohe Akzeptanz unter den Benutzern zu erzielen. Gerade der Gesundheitsbereich ist geprägt durch Personal, das bzgl. Ausbildungsstand und Wissen stark differiert und unterschiedliche Anforderungen an die geplante Applikation stellt. Um die Widerstände gegen die neue Technologie möglichst gering zu halten, müssen Plattform und Endgerät so gewählt werden, dass sich die Benutzergruppen bestmöglich damit identifizieren können. Darüber hinaus muss die mobile Technologie bestimmte technische Charakteristiken aufweisen, die für die Entwicklung der Anwendung essentiell sind. Dies betrifft beispielsweise Sicherheitsaspekte oder die Möglichkeit von Multithreading, um ein Blockieren der Benutzeroberfläche zu verhindern. Basierend auf einer Analyse marktpolitischer und technologischer Faktoren mobiler Technologien soll eine Entscheidung über die Plattform-Unterstützung getroffen werden.

Im zweiten Schritt soll der Fokus des Entwicklungsprozesses auf das Design der Benutzeroberfläche gelegt werden. Wie bereits erwähnt wurde, ist der IT-unterstützte ICF-Einsatz in der Praxis noch nicht gängig. Dadurch ergeben sich neu zu entwickelnde Arbeitsvorgänge und nur vage definierbare Anforderungen an die mobile

Anwendung. Bei einer Anforderungsanalyse zu Beginn eines sequentiellen Softwareentwicklungsprozesses, auf der die Entwicklung bis hin zum Softwareeinsatz aufbaut, ist die Gefahr groß, eine Applikation zu entwickeln, die weder nützlich noch benutzerfreundlich ist. Gerade im anspruchsvollen Berufsalltag des Gesundheitsbereiches wird eine Anwendung, die die Bedürfnisse des Nutzers nicht berücksichtigt, zu einer Belastung und folglich nicht eingesetzt werden. Beispielsweise schreiben Holzinger u. a. [19], dass die großen Vorteile mobiler Anwendungen im Bereich der Medizin und der Pflege aktuell durch eine mangelhafte Benutzerfreundlichkeit vieler Anwendungen abgeschwächt werden. Um dieses Risiko zu schmälern, ist es Ziel dieser Arbeit, im Rahmen eines flexiblen Softwareentwicklungsprozesses, die Anwendung von potentiellen Nutzern evaluieren zu lassen. Die Evaluierung soll Rückmeldungen von Krankenpfleger/-innen und Physiotherapeut/-innen zu ersten Oberflächendesigns auf Papier liefern. Erste Designvorschläge sollen potentielle Nutzer dazu anregen, Vergleiche mit ihrer derzeitigen beruflichen Praxis anzustellen und Verbesserungsvorschläge zu liefern. Zahlreiche wissenschaftliche Arbeiten zu mobilen Gesundheitsanwendungen verweisen auf den Einsatz des sogenannten Prototypings, häufig beginnend bei Papier-Prototypen. Der interessierte Leser sei hier beispielsweise auf [10, 17, 19] verwiesen. Prototypen bilden ein gemeinsames Verständnis von Entwicklern und Nutzern. Ihre Erstellung benötigt meist keinen großen Aufwand, wohingegen ihr Effekt groß ist. Durch die frühe Erstellung und Evaluierung von Prototypen kann der Softwareentwicklungsprozess auf einfache Weise am Anwender orientiert werden. Somit wird das Risiko vermindert eine Applikation zu entwickeln, die am Markt nicht akzeptiert wird.

Basierend auf den ersten Design-Evaluierungen zu Beginn des Softwareentwicklungsprozesses soll die Benutzeroberfläche der mobilen Anwendung angepasst sowie die dahinterstehende Geschäftslogik samt Server-Integration implementiert werden. Die Anforderungen an die Backend-Anbindung sind bei Geschäftsapplikationen besonders hoch. Zahlreiche Qualitätsanforderungen in der Kommunikation und dem Datenabgleich zwischen mobilem Endgerät und zentralem Server müssen beachtet werden, wie beispielsweise die zuverlässige Übermittlung von Nachrichten oder die Unterstützung von Transaktionen, um die Datenkonsistenz zu gewährleisten. Beim Umgang mit sensiblen Gesundheitsdaten sind vor allem auch Sicherheitsaspekte bei der Wahl der Architektur zu berücksichtigen. Daten müssen vor unbefugtem Zugriff

geschützt sowie ihre Integrität und Authentizität sichergestellt werden. Neben diesen Qualitäts- und Sicherheitsanforderungen spielt auch die schnelle und einfache Erweiterbarkeit der Anbindung für ein kleines Softwareunternehmen mit begrenzten Ressourcen eine große Rolle. Unter diesen Gesichtspunkten sollen zwei mögliche Webservice-Architekturen, nämliche SOAP- und REST-basierte Architekturen, analysiert werden. Aufbauend auf dieser Analyse soll die Backend-Anbindung prototypisch realisiert werden.

Nach einer ersten Implementierung der Anwendung ist eine neue Iteration an Usability-Evaluierungen durch Gesundheitspersonal sinnvoll. Im Sinne eines iterativen Prototypings können die Iterationen so lange fortgesetzt werden bis die Anforderungen der Nutzer ausreichend erfüllt werden. Diese Vorgehensweise ist empfehlenswert, fällt aber nicht mehr in den Umfang der vorliegenden Arbeit, die mit einer Iteration an Evaluierungen und der darauf aufbauenden Implementierung der prototypischen Anwendung abschließt.

2. Theoretische Grundlagen

Dieses Kapitel gibt eine Einführung in die theoretischen Grundlagen der vorliegenden Arbeit. In Abschnit 2.1 werden die wichtigsten Usability Engineering Methoden, mit besonderem Fokus auf mobile Endgeräte, vorgestellt. Danach behandelt Abschnitt 2.2 die möglichen Architekturen der Backend-Anbindung. Dazu werden SOAP- und REST-basierte Webservices analysiert.

2.1 Erfolgsfaktor Usability

Dieser Abschnitt stellt mögliche Methoden des Usability Engineerings zur Integration in unseren Softwareentwicklungsprozess vor. Wir wollen trotz begrenzter Ressourcen eine optimierte Benutzerfreundlichkeit unserer mobilen Anwendung erreichen. Allerdings bringen mobile Endgeräte und Anwendungen neue Herausforderungen für ein benutzerfreundliches Design mit sich. Nutzer tendieren besonders zu mobilen Anwendungen, deren Interface sich nicht am traditionellen Desktop-Design orientiert [31]. Denn gerade diese mobilen Anwendungen werden oft als besonders benutzerfreundlich empfunden. Dieses Kapitel behandelt daher zu Beginn in Abschnitt 2.1.1 typische Limitierungen und Besonderheiten mobiler Endgeräte. Danach werden in Abschnitt 2.1.2 die wichtigsten Methoden des Usability Engineerings vorgestellt und mögliche Adaptionen bzw. Kombinationen für den mobilen Kontext erläutert.

2.1.1 Charakeristik mobiler Geräte

Designprinzipien, die für Desktop-Anwendungen Gültigkeit besitzen, können nicht Eins-zu-Eins auf mobile Endgeräte übertragen werden. Aktuelle Smartphones bzw. Tablet-PCs weisen Eigenschaften auf, die neue Anforderungen an ein gelungenes

Oberflächendesign stellen. Im Folgenden werden diese Eigenschaften basierend auf Inostroza u. a. [24] und Nayebi u. a. [31], erläutert.

- **Mobiler Nutzungskontext:** Der Nutzer ist bei der Verwendung eines Smartphones in den meisten Fällen stärker vom Umfeld abgelenkt als am Schreibtisch vor seinem Desktop-PC.
- **Unterschiedliche Bildschirmgrößen und Auflösungen:** Kleine Bildschirme erschweren das Arbeiten mit dem mobilen Gerät. Zusätzlich können niedrige Auflösungen für eine schlechte Qualität sorgen.
- **Begrenzte Ressourcen:** Obwohl Rechenleistung und Speichergrößen auf modernen Smartphones ständig optimiert werden, sind diese Faktoren, ebenso wie die begrenzte Akkulaufzeit, zu beachten.
- **Dateneingabe:** Smartphones werden üblicherweise mit dem Finger oder einem Stylus bedient. Für eine hohe Benutzerfreundlichkeit ist auf einfache und schnelle Eingabemöglichkeiten zu achten. Designs, die auf Tastatur und Maus abzielen sind daher fehl am Platze.

2.1.2 Methoden des Usability Engineerings

Trotz der besonderen Anforderungen, die mobile Endgeräte an ein benutzerfreundliches Design stellen, unterscheiden sich Usability-Engineering-Methoden für mobile Systeme bisher meist nicht von den Methoden des Desktop-Bereichs [25]. Laut Lee und Grice [27] hat das grundlegende Ziel der Methoden, nämlich die Optimierung der Usability, auch für mobile Anwendungen Gültigkeit. Dennoch ist ihrer Meinung nach der Einsatz einer traditionellen Methode für mobile Anwendungen problematisch. Ihrer Meinung nach müssen traditionelle Usability-Methoden adaptiert bzw. unterschiedliche Methoden kombiniert werden, um dem mobilen Kontext gerecht zu werden.

Wir stellen in den folgenden Abschnitten, basierend auf Holzinger [18] und Rubin und Chisnell [38], traditionelle Methoden des Usability Engineerings vor, die laut Holzinger [18] in Softwareentwicklungsprozesse jeglicher Art integriert werden sollten. Relevante Adaptionen oder Kombinationen dieser traditionellen Methoden für den mobilen Bereich heben wir besonders hervor.

Prinzipiell ist bei den Methoden des Usability Engineerings zwischen den sogenannten Untersuchungsmethoden und Testmethoden zu unterscheiden. Bei den Untersuchungsmethoden wird ein Design typischerweise von Usability-Experten auf die Einhaltung etablierter Usability-Standards untersucht. Bei den Testmethoden werden hingegen Rückmeldungen von den tatsächlichen Nutzern des Systems eingeholt.

Untersuchungsmethoden

In diesem Abschnitt beschreiben wir die wichtigsten Untersuchungsmethoden zur Evaluierung der Usability eines Designs.

Heuristische Evaluierung. Bei dieser Methode werden die Designs einer Bewertung durch Usability-Experten unterzogen. Diese greifen bei ihrer Beurteilung typischerweise auf etablierte Heuristiken zurück. Auf Domänenwissen muss bei dieser Methode verzichtet werden, vorausgesetzt sie wird von reinen Usability-Experten ohne Domänenhintergrund durchgeführt. Inostroza u. a. [24] sahen die Notwendigkeit traditionelle Usability-Standards der heuristischen Evaluierung für mobile Oberflächen zu erweitern. Sie präsentieren die folgenden Punkte für heuristische Evaluierungen mobiler Anwendungen:

- **Systemfeedback:** Der Nutzer sollte jederzeit über den Zustand des Systems informiert sein. Als Beispiele können hier Informationen zum Prozessfortschritt langwieriger Operationen oder Benachrichtigungen über fertiggestellte Hintergrundprozesse genannt werden.
- **Orientierung an der realen Welt:** Die Informationsdarstellung sollte sich am Nutzer und seiner Umwelt orientieren. Das verwendete Vokabular sollte an den Domänenhintergrund des Nutzers angepasst werden. Technische Fachbegriffe aus der Programmierung sind, vor allem auch bei Fehlermeldungen, zu vermeiden.
- **Einfache Benutzerführung:** Der Nutzer sollte die Möglichkeit haben, seine Aktionen auf einfache Weise rückgängig zu machen bzw. wiederherzustellen. Mögliche Optionen und Informationen zur Nutzung der Anwendung sollten für den Nutzer stets deutlich sichtbar sein.

- **Konsistenz und Einhaltung von Standards:** Um dem Nutzer die Möglichkeit zu geben bereits erlernte Konzepte weiterhin anzuwenden, sollte auf die Einhaltung etablierter Standards sowie eine konsistente Informationsaufbereitung geachtet werden.

- **Fehlervermeidung und -behebung:** Ein gutes Oberflächendesign soll vor potentiellen Fehlern schützen. Aus diesem Grund sollen beispielsweise nur die zu einem Zeitpunkt verfügbaren Optionen angezeigt werden. Auf mögliche Fehler soll der Nutzer rechtzeitig hingewiesen werden. Bei auftretenden Fehlern sollten die Fehlermeldungen so eindeutig und einfach wie möglich formuliert sein.

- **Individualisierung:** Der Nutzer soll in gewissem Umfang über Konfigurationsmöglichkeiten verfügen, um seine persönlichen Vorlieben einfließen zu lassen.

- **Ästhetisches und minimalistisches Design:** Das Design soll sich auf die wesentlichen Informationen beschränken.

- **Hilfe und Dokumentation:** Eine Schritt-für-Schritt-Anleitung für die aktuelle Aufgabe des Benutzers soll zur Verfügung stehen.

- **Interaktion:** Das Endgerät bzw. die Anwendungen sollen für die wichtigsten Optionen ständig Buttons bereitstellen. Prinzipiell sollte die Positionierung der Oberflächenelemente an die Ergonomie der menschlichen Hand angepasst sein.

Durchspielen von Aufgaben. Bei dieser Methode, im Englischen "Walk-Through" genannt, führt der Designer seinen Kollegen/-innen oder Usability-Experten die Erfüllung einer typischen Aufgabe anhand des entworfenen Prototypen schrittweise vor. Die Experten achten auf Probleme bzw. beurteilen das Design nach zuvor spezifizierten Richtlinien. Domänenspezifisches Wissen wird bei dieser Methode nicht eingebracht bzw. ist auf das Wissen der am Durchgang beteiligten Personen limitiert.

Testmethoden

In diesem Abschnitt behandeln wir die bedeutendsten Testmethoden zur Usability-Evaluierung.

Ethnographische Studien. Ethnographische Studien zielen darauf ab, potentielle Nutzergruppen bei typischen Tätigkeiten im typischen Nutzungskontext zu beobachten. Basierend auf diesen Beobachtungsstudien können Nutzerprofile und Aufgabenbeschreibungen angefertigt werden, die als Ausgangsbasis für den Entwicklungsprozess dienen. Diese Methode wird üblicherweise vor der eigentlichen Implementierung des Produkts eingesetzt. Erst durch die Erkenntnisse dieser Studien können erste Designs und Entwicklungen erfolgreich initiiert werden.

Umfragen. Umfragen ermöglichen es, die Rückmeldung einer großen Anzahl von Nutzern in den weiteren Entwicklungsprozess einfließen zu lassen. Der große Vorteil dieser Methode ist somit die Einbringung von Fachwissen einer breiten Masse.

Zielgruppendiskussion. Bei der Zielgruppendiskussion werden zukünftigen Nutzern bereits in einer sehr frühen Entwicklungsphase Designs und Konzepte vorgelegt, zu denen diese Stellung nehmen. Dabei kann bereits sehr früh Einblick in die Sichtweise von potentiellen Nutzern gewonnen werden. Allerdings besteht hier die Gefahr, dass eine Lücke zwischen der Einschätzung der Nutzer und der Realität bestehen bleibt. Diese Lücke kann oft durch reine Diskussion, ohne Beobachtung der Nutzer bei der praktischen Erfüllung ihrer Aufgaben, nicht geschlossen werden.

Usability-Tests. Bei dieser Methode werden sowohl qualitative als auch quantitative Daten über das Design gesammelt, indem Testnutzer vorgegebene Aufgabenstellungen mit dem entwickelten Prototypen oder Produkt bearbeiten. Der große Unterschied zu anderen Methoden, wie beispielsweise der Zielgruppendiskussion oder den Umfragen, ist die Möglichkeit Nutzer tatsächlich bei der Aufgabenbewältigung zu beobachten anstatt auf ihre bloße Einschätzung zu vertrauen. Die Einbringung von Fachwissen ist daher besonders stark ausgeprägt. Ein Einsatz von Usability-Tests vor der eigentlichen Implementierungsarbeit ist möglich. Dazu wird

der Software-Prototyp durch Papier-Prototypen ersetzt. Speziell für die Entwicklung mobiler Gesundheitsanwendungen empfehlen einige wissenschaftliche Arbeiten die iterative Evaluierung anhand von Prototypen, oftmals beginnend bei Papier-Prototypen, wie beispielsweise in [10, 19, 23] ersichtlich.

Häufig eingesetzt wird im Rahmen von Usability-Tests die sogenannte "Thinking Aloud"-Methode, bei der die Testperson dazu aufgefordert wird, ihre Gedanken zu verbalisieren, während sie eine Aufgabenstellung zu bewältigen versucht. Auch wenn das Verbalisieren von Gedanken für die Testperson eventuell unnatürlich ist und ihr Vorgehen verlangsamt, ist es somit möglich zu erkennen, warum bestimmte Aktionen durchgeführt bzw. Fehler gemacht oder vermieden werden.

Während Holzinger [18] die Kombination von Usability-Methoden für Softwareentwicklungen jeglicher Art betont, wird diese Notwendigkeit verstärkt bei mobilen Anwendungen gesehen. Durch die Kombination unterschiedlicher Methoden des Usability Engineerings soll in Summe den komplexen Rahmenbedingungen für die Usability-Evaluierung mobiler Anwendungen entgegengewirkt werden. Der Einsatz von Untersuchungs- und Testmethoden über den Entwicklungsprozess der Anwendung hinweg soll möglichst gewinnbringend aufeinander abgestimmt werden. Beispielsweise zeigen Lee und Grice [27] sowie Fiotakis u. a. [15] in ihren Arbeiten die erfolgreiche Kombination von heuristischer Evaluierung und Usability-Tests zur Evaluierung mobiler Anwendungen.

2.2 Webservice-Architekturen

Ein Schlüsselfaktor jeder mobilen Geschäftsanwendung ist das Interaktionskonzept von mobiler Applikation und Backend-Server. Webservices stellen in diesem Zusammenhang eine populäre Architekturmöglichkeit dar, deren theoretische Grundlagen wir in diesem Kapitel erläutern. Nachdem wir in Abschnitt 2.2.1 Webservices definieren und ihre Charakteristiken erläutern, definieren wir in Abschnitt 2.2.2 besondere Anforderungen von Geschäftsanwendungen, die bei der Architektur von Webservices zu beachten sind. Danach beschreiben wir in den Abschnitten 2.2.3 und 2.2.4 die Grundprinzipien von SOAP- und REST-basierten Webservices mit dem Fokus auf die zuvor definierten Anforderungen.

2.2.1 Einleitung

Das Grundkonzept von Webservices geht auf Dave Winer zurück, der das Protokoll XML-RPC (Remote Procedure Call) entworfen hat [16]. Dieses spezifiziert simple Methodenaufrufe mittels XML-Nachrichten über HTTP. Mittlerweile wurde dieses einfache Protokoll erfolgreich zu unterschiedlichen Webservice-Architekturen weiterentwickelt. Die grundlegende Idee ist immer dieselbe: Webservices bieten heterogenen Systemen eine standardisierte Möglichkeit der Kommunikation und Interaktion. Die Dienste von Anwendungen können somit unabhängig von ihrer Programmiersprache oder ihrem Betriebssystem von anderen Anwendungen genutzt werden [30]. Zahlreiche Unternehmen bieten über diese Architektur ihre Dienste bzw. Services anderen Unternehmen an. Voraussetzung für diese plattformunabhängige Zusammenarbeit von verteilten Anwendungen sind das Internet und internetbasierte Protokolle, über die Nachrichten versandt werden. Das Format dieser Nachrichten baute lange Zeit vorrangig auf XML auf. Besonders bekannt ist beispielsweise das XML-basierte Nachrichtenformat SOAP (Simple Object Access Protocol). In letzter Zeit werden allerdings auch simplere Formate, wie beispielsweise JSON (Java Simple Object Notation), immer populärer. Unabhängig vom gewählten Format kapseln diese, zwischen unterschiedlichen Anwendungen versandten, Nachrichten, stets Methodenaufrufe auf einer fremden Anwendung bzw. das Resultat eines solchen Aufrufes.

Während über die Jahre hinweg viele Spezifikationen für SOAP-basierte Webser-

vices entstanden, wurde besonders in letzter Zeit die REST-basierte Architektur immer populärer. Das World Wide Web Consortium (W3C) [92] beschreibt in diesem Zusammenhang zwei große Klassen von Webservices [86]:

- **SOAP-basierte Webservices:** Diese stellen dem Client beliebige Methoden zur Verfügung und orientieren sich bei ihrer Architektur an den angebotenen Services der Geschäftsprozesse.

- **REST-basierte Webservices:** Diese orientieren sich an den hinter den Geschäftsprozessen stehenden Ressourcen und manipulieren diese über eine homogene Schnittstelle zustandsloser Operationen.

2.2.2 Besondere Anforderungen von Geschäftsanwendungen

Bevor eine Analyse der eingangs erwähnten Webservice-Architekturen angestellt wird, definieren wir besondere Anforderungen von Geschäftsanwendungen, die Einfluss auf die Wahl der Architektur haben. Mobile Geschäftsanwendungen haben im Vergleich zu privaten Anwendungen oftmals erhöhte Anforderungen im nichtfunktionalen Bereich. Ein Großteil der Anforderungen betrifft die Sicherheit der ausgetauschten Daten sowie die Qualität der Kommunikation mit dem Backend-Server. Die nächsten Abschnitte geben einen kurzen Überblick über diese Anforderungen.

Qualitätsanforderungen

Mobile Geschäftsanwendungen sind von zahlreichen Anforderungen flankiert, die die Kommunikation und den Datenabgleich mit dem Backend-Server betreffen. Beispielsweise muss die Zustellung von Nachrichten garantiert oder die Durchführung von Transaktionen ermöglicht werden, um die Datenkonsistenz zu gewährleisten. Für unsere geplante Applikation ist eine sogenannte "Occasionally-Connected"-Architektur geplant, was impliziert, dass Daten lokal gespeichert werden, um eine Nutzung der Applikation auch bei schlechter Netzabdeckung zu ermöglichen. Ein sicherer und konsistenter Datenabgleich mit dem Server ist daher essentiell.

Sicherheitsanforderungen

Geschäftsanwendungen verarbeiten oftmals sensible Daten und haben daher hohe Sicherheitsanforderungen, um diese vor unerlaubten Zugriffen zu schützen. Gesundheitsapplikationen unterliegen in diesem Zusammenhang besonderen europäischen Richtlinien und darauf basierenden nationalen Gesetzen. Die rechtlichen Grundlagen sind beispielsweise das Datenschutzgesetz (DSG) und das Gesundheitstelematikgesetz (GTelG). Im Rahmen dieser Arbeit soll keine detaillierte, rechtliche Beurteilung der Datenschutzerfordernisse unserer Anwendung vorgenommen werden. Dennoch muss beachtet werden, dass Gesundheitsdaten besonders schutzwürdige Daten darstellen (§ 4 Z2 DSG) und der Gesetzgeber besondere Datenschutzvorschriften für diese vorsieht, wie sie bspw. in § 14 DSG genannt werden.

Zusammengefasst tangieren folgende datenschutzrechtliche Aspekte die Umsetzung mobiler Geschäftsanwendungen:

- **Authentifizierung:** Um sensible Daten vor unbefugten Zugriffen zu schützen, müssen Authentifierungsmechanismen eingesetzt werden. Diese reichen von der simplen Authentifizierung des Hypertext Transfer Protocols (HTTP) bis hin zur Verwendung elektronischer Zertifikate.

- **Vertraulichkeit:** Sowohl die gespeicherten als auch die zu übertragenden Daten müssen verschlüsselt werden. Bei der Übertragung kann beispielsweise auf die Verschlüsselungsmechanismen der Transport Layer Security (TLS) gebaut werden.

- **Integrität:** Die Integrität der übertragenen Daten kann über elektronische Signaturen erfolgen. Alternativ könnten beispielsweise Message Authentication Codes (MACs) verwendet werden.

2.2.3 SOAP-basierte Webservices

Die in diesem Kapitel vorgestellten Webservices verfolgen eine SOAP-basierte Architektur. Sie stellen Services über ein Netzwerk zur Verfügung, die plattformübergreifend von Anwendungen genutzt werden [29]. Im krassen Unterschied zum Representational State Transfer (REST), den wir später näher vorstellen, sieht die Schnittstelle

hier für jeden Webservice anders aus. Jeder Webservice veröffentlicht basierend auf einer Dienstbeschreibung Methoden, die sich an den von ihm angebotenen Diensten orientieren. In den nächsten Abschnitten werden wir die grundlegenden Komponenten der SOAP-basierten Webservices detailliert erläutern.

SOAP. Wie bereits erwähnt, kommunizieren Anwendungen und Webservices oder Webservices untereinander über speziell formatierte Nachrichten. Heutzutage ist der populärste Nachrichtenstandard das auf der Extensible Markup Language (XML) basierende Simple Object Access Pattern, kurz SOAP [37]. SOAP ist aus XML-RPC hervorgegangen. Einer der größten Nachteile dieses Protokolls war das Fehlen von Metadaten zur Beschreibung der Methodenaufrufe [16]. Während die XML-Syntax aus XML-RPC zur Erhaltung der Plattformunabhängigkeit beibehalten wurde, nahm sich der Nachfolger aber der fehlenden Metadaten an, und definierte den SOAP-Envelope, der alle ausgetauschten Nachrichten inklusive Metadaten kapselt. Er besteht nach Hein aus [16]:

- **Header:** Dieser optionale Nachrichtenteil enthält die Metadaten der Nachricht. Durch die Inkludierung von Metadaten erreicht SOAP Unabhängigkeit vom Transportprotokoll. Denn der Kopf von SOAP-Nachrichten enthält selbst die Informationen, die ansonsten vom Übertragungsprotokoll getragen werden [16]. Die Metadaten in den Köpfen der SOAP-Nachrichten, die von Transaktions-IDs bis hin zu Signaturen reichen, werden vor allem von höheren Middleware-Diensten genutzt, die entsprechend Transaktionen durchführen oder die Integrität der Nachricht überprüfen [13]. In der Praxis wird die Unabhängigkeit vom Transportprotokoll allerdings nur in den seltensten Fällen genutzt. Nahezu alle Nachrichten werden über HTTP übertragen [29, 37].

- **Body:** Dieser enthält die eigentlichen Nutzdaten, also die Informationen für den Methodenaufruf am Server, wie Methodenname und Parameter, beziehungsweise das Resultat des Methodenaufrufes.

Listings 2.1 und 2.2 zeigen, basierend auf [81], zwei Beispiele für den Aufbau von SOAP-Nachrichten. Der Webservice bietet hier eine Methode zum Suchen von Patienten anhand ihrer ID an. Die Übertragung der Nachrichten aus Listing 2.1 und

2.2 erfolgen über das HTTP-Protokoll. SOAP sieht für die Übertragung über HTTP vor, die POST-Methode zu nutzen [13] .

```
1 POST /patienten-service HTTP/1.1
2 Host: beispiel.org
3 Content-Type: application/soap+xml; charset=utf-8
4 Content-Length: nnn
5
6 <?xml version="1.0"?>
7 <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-
      envelope"
8     xmlns:s="http://www.beispiel.org/patienten-service">
9     <env:Body>
10       <s:GetPatient>
11             <s:ID>2</s:ID>
12       </s:GetPatient>
13    </env:Body>
14 </env:Envelope>
```

Listing 2.1: SOAP-Anfrage an einen Patienten-Service.

```
1 HTTP/1.1 200 OK
2 Content-Type: application/soap+xml; charset=utf-8
3 Content-Length: nnn
4
5 <?xml version="1.0"?>
6 <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-
      envelope"
7     xmlns:s="http://www.beispiel.org/patienten-service">
8     <env:Body>
9       <s:GetPatientResponse>
10            <s:Vorname>Marianne</s:Vorname>
11            <s:Nachname>Musterfrau</s:Nachname>
12            <s:Alter>75</s:Alter>
13      </s:GetPatientResponse>
14    </env:Body>
```

```
15 </env:Envelope>
```

Listing 2.2: SOAP-Antwort eines Patienten-Services.

Webservice Description Language. Die Webservice Description Language, kurz WSDL, stellt eine Möglichkeit zur Dienstbeschreibung von Webservices dar. Auf Basis dieser Schnittstellenbeschreibung kann der Client seine Anfragen an den Server formulieren. WSDL-Beschreibungen werden zum Erhalt der Sprachenunabhängigkeit mit einer eigenen XML-Grammatik verfasst und umfassen, nach Hein und Zeller, vier Bereiche [16]:

1. Schnittstellendefinitionen aller öffentlichen Methoden.
2. Beschreibung der auszutauschenden Datentypen.
3. Informationen zum genutzten Transportprotokoll.
4. Informationen zum Auffinden des Services, wie beispielsweise die Server-Adresse.

Aus den WSDL-Beschreibungen ist es mit speziellen Tools möglich, automatisiert Client-Proxies für unterschiedliche Programmiersprachen zu erstellen. Diese Client-Proxies abstrahieren und kapseln die Kommunikation mit dem Webservice [13].

Universal Description, Discovery and Integration. Das Universal Description, Discovery and Integration Protokoll (UDDI) definiert eine Verzeichnisstruktur für Webservices. Relevante Informationen zu Webservices können hier publiziert werden, um interessierten Clients das Auffinden zu erleichtern [16]. Wir erwähnen UDDI nur am Rande, da dieser Verzeichnisdienst keine essentielle Rolle spielt, solange die benötigten Webservices der Anwendung ohnehin bekannt sind.

SOAP-basierte Webservices für Geschäftsanwendungen

Konzepte, die besonders für komplexe Geschäftsanwendungen relevant sein können, wie etwa Transaktionen, Sicherheit oder zuverlässige Nachrichtenübertragungen, werden von SOAP selbst nicht spezifiziert. Allerdings behandeln die entwickelten Erweiterungen zu SOAP, das Webservice Framework (WS-Framework), diese

Problematiken. Sehr vereinfacht dargestellt, erweitern die WS-Spezifikationen die SOAP-Nachrichtenköpfe um spezielle Metadaten, mit denen spezielle Services bei Bedarf interagieren. Beispielsweise interagiert ein intermediärer Sicherheitsservice, der die Integrität der SOAP-Nachrichten überprüft, mit einer speziellen Signatur im SOAP-Nachrichtenkopf [13]. Das W3C hat eine Vielzahl an WS-Spezifikationen veröffentlicht, von denen einige in den folgenden Abschnitten diskutiert werden.

Zustandsorientierte Kommunikation. Grundsätzlich sind SOAP-Webservices zustandslos. Über die Erweiterung WS-Addressing ist es aber möglich die Kommunikation zustandsorientiert zu gestalten. Dabei nutzt WS-Addressing das bekannte Konzept der Sessions. Der Webservice erzeugt für einen Client eine sogenannte "EndpointReference", die in den folgenden Anfragen des Clients mitgesandt wird und dem Server die Zuordnung des anfragenden Clients zum gespeicherten Zustand ermöglicht [50].

Asynchrone Kommunikation. SOAP selbst schreibt keine synchrone oder asynchrone Kommunikation vor, sondern legt nur den Aufbau einer Nachricht fest. Wird ein synchrones Kommunikationsprotokoll, wie etwa HTTP, genutzt, so muss SOAP um die Spezifikation WS-Addressing erweitert werden, um eine asynchrone Kommunikation zu unterstützen. Der Kern von WS-Addressing ist die Eigenschaft "ReplyTo", die dem SOAP-Nachrichtenkopf hinzugefügt wird und dem Webservice die Adresse bekannt gibt, an die die Antwort zu senden ist. Wird "asynchron" in dem Sinne verstanden, dass der Client während der Server-Operation nicht blockiert, die Antwort aber im Kontext der initialen HTTP-Verbindung erhält, so ergibt sich für den Webservice selbst kein Unterschied zu synchronen Aufrufen [42]. Erfolgt die Antwort asynchron im Sinne einer für die Antwortnachricht eigens erstellten HTTP-Verbindung zum Client, so wird der Client in diesem Moment selbst zum Server und muss einen adressierbaren ServiceEndpoint bereitstellen [42].

Zuverlässige Nachrichtenübertragung. Wird SOAP über HTTP genutzt, kann keine zuverlässige Nachrichtenübertragung garantiert werden. Das HTTP-Protokoll basiert auf TCP, das die Zustellung von Nachrichten nicht gewährleisten kann. Bei einer Zeitüberschreitung erklärt TCP die Verbindung als unterbrochen und die Kom-

munikationspartner wissen nicht, ob die letzten Nachrichten angekommen sind oder nicht. Über die Spezifikation WS-ReliableMessaging, die je einen zusätzlichen Vermittler auf Client- und Serverseite vorsieht, kann die Zustellung von Nachrichten garantiert werden. Die zur Verfügung stehenden Optionen sind “AtLeastOnce”, “AtMostOnce”, “ExactlyOnce” und “InOrder” [13, 37]. Für eine detailliertere Beschreibung dieser Spezifikation verweisen wir auf [74].

Transaktionen. Arbeiten mehrere Webservices zusammen, um eine Aufgabe zu bewältigen, werden Algorithmen für verteilte Transaktionen benötigt, um einen konsistenten Zustand aller Webservices zu gewährleisten. Das typische Beispiel, das in diesem Zusammenhang genannt wird, ist der Webservice des Reiseveranstalters, der mit anderen Webservices zur Hotel-, Flug- und Mietwagenbuchung zusammenarbeitet. Nur wenn sowohl Flug als auch Hotel und Mietwagen für einen bestimmten Zeitraum verfügbar sind, soll die Buchung durchgeführt werden. Zur Lösung derartiger Probleme wurde das Web Services Transaction Framework entwickelt, das hier basierend auf der Arbeit von Melzer u. a. beschrieben wird [29]:

- **WS-Coordination:** Diese Erweiterung spezifiziert den für verteilte Transaktionen nötigen Koordinator. Er ist für die Erzeugung einer neuen Transaktion zuständig und ermöglicht angesprochenen Komponenten das Beitreten zu dieser. WS-Coordination bildet die Basis für die zwei nachfolgend beschriebenen Arten von Koordinationen.

- **WS-AtomicTransaction:** Diese Spezifikation implementiert das Zwei-Phasen-Commit-Protokoll für verteilte Transaktionen, bei dem die einzelnen Komponenten über einen Koordinator zusammenarbeiten. Grundsätzlich erfüllt dieses Protokoll alle ACID-Eigenschaften, also Atomarität, Konsistenz, Isolation und Durabilität. Ein vollständiges Rollback ist beim Fehlschlagen einer Operation möglich.

- **WS-BusinessActivity:** Während WS-AtomicTransaction bei kurzlebigen Transaktionen angewandt wird, die relevante Ressourcen für die Dauer der Transaktion sperren, werden mit WS-BusinessActivity lang andauernde Transaktionen behandelt. Statt Ressourcen über eine lange Zeit zu sperren, werden Änderungen durchgeführt ohne notwendigerweise die Zustimmung aller an der

Transaktion beteiligten Komponenten zuvor einzuholen. Im Fehlerfall werden Operationen ausgeführt, die Teile der Transaktion wieder zurücksetzen.

Sicherheit. In der grundlegenden Spezifikation von SOAP werden sicherheitsrelevante Themen, wie etwa die Vertrauenswürdigkeit oder Integrität der ausgetauschten Nachrichten, nicht angesprochen. Wählt man als Transportprotokoll HTTPS, baut also auf TLS auf, so kann man auf die Sicherheitsmechanismen dieses Protokolls zurückgreifen. Allerdings ist dieses limitiert. TLS kann nur eine Punkt-zu-Punkt-Sicherheit gewährleisten. Das bedeutet, dass zwar der Pfad zwischen zwei beliebigen Intermediären gesichert wird, nicht aber der gesamte Kommunikationspfad von einem Ende zum anderen. Diese Limitierung war ein Grund zur Ausarbeitung der Spezifikation WS-Security. Durch den Einsatz von WS-Security werden die sicherheitsrelevanten Aspekte aus der Transportebene in die Nachrichtenebene transferiert [29]. Hierzu wurden keine neuen Sicherheitsverfahren entwickelt, sondern der standardisierte Einsatz bereits bewährter Konzepte vorgeschlagen [29]. Nachdem SOAP-Nachrichten auf XML basieren, war der Einsatz von XML-Signaturen und XML-Verschlüsselung naheliegend. WS-Security setzt in diesem Zusammenhang auch fest, wie Teile einer Nachricht verschlüsselt bzw. signiert werden können, um verschiedenen Intermediären Zugriff auf Teile der Nachricht zu gewähren bzw. die Integrität eines bestimmten Nachrichtenteils zu überprüfen [29]. WS-Security ist aber nicht auf XML-Signaturen und -Verschlüsselungen beschränkt, sondern stellt eine Grundlage für weitere Sicherheitskonzepte, wie beispielsweise WS-Policy oder WS-Trust, dar [29].

2.2.4 REST-basierte Webservices

Der Begriff Representational State Transfer, kurz REST, wurde im Jahr 2000 von Roy Fielding [14] geprägt. Fielding beschreibt mit REST eine Softwarearchitektur, die der des Webs entspricht. Daten werden dabei über ein homogenes Interface als Ressourcen bereitgestellt und verarbeitet. REST stellt somit eine einfache und einheitliche Alternative zur SOAP-basierten Architektur dar, deren Daten über ein komplexes, heterogenes Interface verarbeitet werden [37]. Eine ressourcenorientierte Architektur ist grundsätzlich vom Applikationsprotokoll unabhängig. Da aber

nahezu alle REST-basierten Webservices auf HTTP aufbauen und dessen homogenes Interface, die HTTP-Methoden, nutzen, wird die Architektur in den folgenden Abschnitten basierend auf dem HTTP-Protokoll erläutert.

Die folgenden Abschnitte behandeln, basierend auf der Arbeit von Richardson und Ruby [37], die wichtigsten Merkmale REST-basierter Webservices.

Adressierbarkeit. Ein REST-basierter Webservice wird als adressierbar bezeichnet, stellt er allen anfragenden Anwendungen relevante Informationen über einen eindeutigen Namen und eine eindeutige Adresse bereit. Dies geschieht typischerweise über Uniform Resource Identifiers (URIs). Eine relevante Information kann dabei jede speicherbare Information sein, von einem Datenbankeintrag bis hin zum Ergebnis einer Berechnung. Derartige, über URIs adressierbare Informationen werden als Ressourcen bezeichnet. Im Zusammenhang mit Ressourcen stehen deren Repräsentationen. Eine Repräsentation ist eine Darstellung des aktuellen Zustands einer Ressource in einem bestimmten Format. Typische Formate für Repräsentationen sind beispielsweise XML oder die JavaScript Object Notation (JSON). Listing 2.3 zeigt zwei typische Beispiele für URIs eines REST-basierten Webservices. Während die erste URI alle Patienten adressiert, adressiert die zweite URI den Patienten mit der ID 2. Eine beispielhafte JSON-Repräsentation der Daten dieses Patienten zeigt Listing 2.4.

```
1 http://www.beispiel.com/patienten
2 http://www.beispiel.com/patienten/2
```

Listing 2.3: Beispiele für REST-typische URIs.

```
1 {
2     "id": 2
3     "vorname": "Marianne",
4     "nachname": "Musterfrau",
5     "alter": 75,
6     "adresse": {
7         "strasse": "Musterstrasse 4",
8         "stadt": "Graz",
9         "plz": "8010"
```

```
10       }
11 }
```

Listing 2.4: JSON-Repräsentation einer Patienten-Ressource.

Zustandslosigkeit. Das Konzept der Zustandslosigkeit drückt aus, dass der Server HTTP-Anfragen unabhängig von einander betrachtet. Er macht die Verarbeitung einer Anfrage zu keinem Zeitpunkt abhängig vom Resultat einer vorhergehenden. Alle Informationen, die für die Verarbeitung einer Anfrage nötig sind, sind vom Client mitzuliefern. Beispielsweise entspricht eine Session-ID nicht dem REST-Prinzip der Zustandslosigkeit. Eine Session-ID knüpft an einen bestimmten Anwendungszustand am Server an. Bei der Nutzung von Session-IDs ist also der Server für das Speichern des Anwendungszustandes verantwortlich. REST sieht hingegen vor, dem Client diese Verantwortung zu übergeben. Dazu muss der Server dem Client mögliche, zukünftige Zustände als adressierbare Ressourcen mitteilen. Der Client kann diese dann aktiv ansteuern. Auch wenn die Zustandslosigkeit die Verantwortung des Clients erhöht, ist es durch die Unabhängigkeit der einzelnen HTTP-Anfragen beispielsweise einfacher möglich eine Anwendung über Lastausgleichsserver zu verteilen oder Daten zu cachen.

Konnektivität. Das Merkmal der Konnektivität beschreibt, dass Repräsentationen von Ressourcen auf andere Ressourcen verlinken sollen, um dem Client so potentielle Pfade durch die Ressourcen aufzuzeigen. Fragt ein Client beispielsweise die Patienten-Ressource `http://www.beispiel.com/patienten` ab, wird ihn die Repräsentation auf die Ressourcen einzelner Patienten verweisen, wie in Listing 2.5 gezeigt.

```
1 [
2   {"id": 1,
3     "uri": "http://www.beispiel.com/patienten/1"} ,
4   {"id": 2,
5     "uri": "http://www.beispiel.com/patienten/2"}
6 ]
```

Listing 2.5: Verlinkung auf Ressourcen innerhalb einer REST-Repräsentation.

Homogenes Interface. HTTP bietet, entsprechend der REST-basierten Architektur, ein homogenes Interface an, um auf Ressourcen zuzugreifen:

- Abfragen einer Ressource: `GET`.
- Erzeugen einer Ressource: `POST` oder `PUT` (abhängig davon, ob die zu erstellende URI vom Server oder vom Client bestimmt wird).
- Ändern einer Ressource: `PUT`.
- Löschen einer Ressource: `DELETE`.
- Abfragen der Meta-Daten einer Ressource: `HEAD`.
- Abfragen der unterstützen HTTP-Methoden einer Ressource: `OPTIONS`.

Während die SOAP-basierte Architektur für jeden Service unterschiedliche Interfaces anbietet, ist das Interface für REST-basierte Architekturen immer dasselbe. Eine GET-Anfrage bedeutet beispielsweise immer, dass die Repräsentation einer Ressource gelesen werden soll, unabhängig davon, auf welche Ressource die Anfrage abzielt.

Für datenbankbasierte Webservices bietet das Interface den zusätzlichen Vorteil, dass die einzelnen Methoden mit den CRUD-Operationen (**C**reate, **R**ead, **U**pdate, **D**elete) von relationalen Datenbanken konform sind und die Datenbank-Anbindung somit intuitiv erfolgen kann:

- `POST` oder `PUT` $\Longrightarrow$ `CREATE`.
- `GET` $\Longrightarrow$ `READ`.
- `PUT` $\Longrightarrow$ `UPDATE`.
- `DELETE` $\Longrightarrow$ `DELETE`.

Durch das homogene Interface der REST-basierten Architektur ist das Verhalten jeder Methode bekannt und das Verhalten des Clients kann darauf abgestimmt werden:

- GET ist sicher, d.h. eine Anfrage kann beliebig oft abgesetzt werden und das Resultat bleibt dasselbe. Die Ressource wird durch GET nie verändert, sondern nur gelesen. Sollte eine GET-Anfrage also aufgrund eines Fehlers kein Resultat liefern, kann der Client die Anfrage gefahrlos erneut senden.

- PUT und DELETE sind idempotent, d.h. eine Anfrage kann ein bis x Mal abgesetzt werden und das Resultat ist immer dasselbe. Wird zum Beispiel der Patient `http://www.beispiel.com/patienten/1` gelöscht, ändert eine zweite oder dritte DELETE-Anfrage auf dieselbe URI nichts am Ergebnis. Schließlich wurde der Patient mit der ID 1 bereits gelöscht. Sollte also ein PUT oder DELETE keine Antwort liefern, kann der Client auch diese Anfragen ohne Bedenken erneut absetzen.

- POST ist weder sicher noch idempotent. Ein POST ein zweites oder drittes Mal abzusetzen, würde das Ergebnis verändern. Eine mögliche Lösung, um auch POST idempotent zu gestalten, ist die Implementierung von Post Once Exactly (POE), die in einem späteren Abschnitt noch detailliert behandelt wird.

REST für Geschäftsanwendungen

Ein großer Vorteil REST-basierter Webservices liegt in der Möglichkeit den Bandbreitenverbrauch zu limitieren. Einerseits wird das durch bedingte HTTP-Anfragen und optimiertes Caching, andererseits durch frei wählbare Datenformate ermöglicht. Zu den populärsten zählen sicher XML und JSON, wobei das schlankere JSON-Format in den meisten Fällen zu bevorzugen ist. Ein weiterer, wichtiger Vorteil ist die Einfachheit der Architektur und somit die Möglichkeit REST-Clients schnell und einfach zu entwickeln und zu erweitern. Obwohl vieles für REST spricht und der Trend zu dieser Architektur auch bei großen Unternehmen, wie Amazon [47], Google [54] oder Microsoft [71], steigt, haftet REST dennoch der Ruf an, auf die Abbildung simpler Datenbankoperationen und die Sicherheit von TLS limitiert und daher für große Geschäftsanwendungen ungeeignet zu sein. Während SOAP zahlreiche WS-Spezifikationen anbietet, um bspw. hohen Sicherheitsanforderungen gerecht zu werden, soll in den nächsten Abschnitt gezeigt werden, dass mit den einfachen

Grundprinzipien von REST ebenfalls eine hohe Sicherheit und Qualität der Kommunikation erzielt werden kann.

Zustandsorientierte Kommunikation. Wie bereits beschrieben, ist ein Grundprinzip von REST die Zustandslosigkeit des Servers. Das bedeutet aber nicht, dass eine zustandsorientierte Kommunikation nicht möglich ist. Sie muss nur vom Client aus steuerbar sein. Als typisches Beispiel nennen Richardson und Ruby [37] den Warenkorb eines Online-Shops. Der Warenkorb eines bestimmten Kunden kann am Server als eine eigene Ressource angelegt werden. Die URI des Warenkorbs wird dem Client mitgeteilt. Der Kunde fügt dieser Ressource so lange neue Produkte hinzu oder entfernt diese wieder bis er sich entschließt alle in der Warenkorb-Ressource enthaltenen Artikel zu bestellen. Somit geht die Zustandsinformation des Warenkorbs von einer Anfrage bis zur nächsten nicht verloren, allerdings ist der Client dafür verantwortlich seine Warenkorb-Ressource zu verwalten.

Asynchrone Kommunikation. Grundsätzlich stellt HTTP ein synchrones Anfrage-Antwort-Protokoll dar. Doch für HTTP-Anfragen, deren Ausführung längere Zeit beansprucht, muss die Möglichkeit einer asynchronen Kommunikation geschaffen werden. Um diese mit der REST-Architektur konform aufzubauen, muss die asynchrone Kommunikation laut Richardson und Ruby [37] in mindestens zwei synchrone Anfragen aufgeteilt werden. Die erste Anfrage startet die Operation. Der Server liefert in diesem Fall den Antwort-Code 202 "Accepted“ sowie eine URI an den Client zurück, auf der der Status der Operation abefragt werden kann. Hat der Client das Resultat der Operation auf dieser URI abgefragt, kann sie mit einer DELETE-Anfrage wieder gelöscht werden.

Zuverlässige Nachrichtenübertragung. Während SOAP mittels WS-Reliable-Messaging sicherstellen kann, dass eine Nachricht den Server erreicht hat, setzt REST auf die Idempotenz seiner HTTP-Methoden. Denn, wie bereits zuvor beschrieben, sind die Methoden GET, PUT und DELETE sicher bzw. idempotent. D.h. sollte eine dieser Methoden kein Ergebnis liefern, kann dieselbe Anfrage gefahrlos noch einmal versandt werden. Ein mehrmaliger Versand hat keine Auswirkungen auf die Ressourcen. Die einzige Methode, die noch zusätzlich abgesichert werden

muss, um eine zuverlässige Nachrichtenübertragung zu gewährleisten, ist POST. Richardson und Ruby [37] schlagen hier vor Post Once Exactly (POE) einzusetzen. Unterstützt eine Ressource POE, akzeptiert sie ein einziges Mal ein POST. Alle darauffolgenden POST-Anfragen werden mit dem HTTP-Error-Code 405 "Method Not Allowed" beantwortet und haben keinen Einfluss mehr auf die Ressource. Erhält ein Client nun keine Antwort auf seine POST-Anfrage, kann er gefahrlos dieselbe Anfrage erneut senden. POST wird mittels POE also idempotent gestaltet.

Transaktionen. Die REST-basierte Architektur manipuliert in einer Anfrage exakt eine Ressource. Es gibt allerdings Fälle, in denen mehrere Ressourcen gleichzeitig manipuliert werden müssen, um die Datenkonsistenz zu gewährleisten. Das typische Beispiel für diesen Fall ist ein Geldtransfer von Konto A auf Konto B. Entweder das Geld wird von Konto A abgebucht und Konto B gutgeschrieben oder nichts davon geschieht. Eine teilweise Ausführung dieser Operation führt zu inkonsistenten Daten. Nachdem eine einzelne HTTP-Anfrage nach dem REST-Prinzip nur eine Ressource, also entweder Konto A oder Konto B manipulieren kann, muss für Transaktionen eine eigene Transaktionsressource eingeführt werden. Dieser speziellen Ressource können Bestandteile der Transaktion, in unserem Fall "Abbuchung Konto A" und "Gutschrift Konto B", hinzugefügt werden, bevor die gesamte Transaktion schlussendlich gelöscht oder ausgeführt wird. Das exakte Vorgehen wird basierend auf dem Beispiel von Richardson und Ruby [37] erklärt. Sowohl Konto `A` als auch Konto `B` weisen in diesem Beispiel einen Kontostand von jeweils $ 200 auf und eine Überweisung von $ 50 von Konto `A` auf Konto `B` wird vorgenommen. Zuerst wird, wie in Listing 2.6 gezeigt, eine neue Transaktions-Ressource erzeugt. Der Server vergibt der neuen Ressource die ID 123, wie in Listing 2.7 gezeigt. Dieser Ressource werden nun die Abbuchungs- bzw. Gutschriftsaufträge mitgeteilt, wie in den Listings 2.8 und 2.9 dargestellt. Listing 2.10 zeigt abschließend die Anfrage zur Durchführung der Transaktion. Durch diese Vorgehensweise kann den ACID-Anforderungen auch mit einer REST-basierten Architektur entsprochen werden.

Die Spezifikationen bzgl. verteilter Transaktionen, also Transaktionen, die über mehrere Webservices hinweg verteilt sind, sind für SOAP-basierte Architekturen ausgereifter, wie in Abschnitt 2.2.3 beschrieben wurde. Allerdings gibt es bereits Arbeiten, die sich mit verteilten Transaktionen auf REST-basierten Systemen be-

schäftigen. Da für unsere Applikation derartige Transaktionen vorläufig keine Rolle spielen, gehen wir an dieser Stelle nicht näher auf derartige Konzepte ein und verweisen den interessierten Leser auf die Arbeit von Pautasso [32].

```
1 POST /transactions/account-transfer
```

Listing 2.6: Erzeugung einer neuen Transaktions-Ressource.

```
1 201 Created
2 Location:
3 /transactions/account-transfer/123
```

Listing 2.7: Server-Antwort beinhaltet erzeugte Transaktions-Ressource.

```
1 PUT /transactions/account-transfer/123/accounts/A
2 "balance":150
```

Listing 2.8: Abbuchung Konto A.

```
1 PUT /transactions/account-transfer/123/accounts/B
2 "balance":250
```

Listing 2.9: Gutschrift Konto B.

```
1 PUT /transactions/account-transfer/123
2 "commited":1
```

Listing 2.10: Durchführung der Transaktion.

Sicherheit. Die Sicherheit REST-basierter Webservices baut auf der Sicherheit von HTTPS, d.h. von HTTP über TLS, auf. Während SOAP die Sicherheitskonzepte in den Nachrichten selbst implementiert, überlässt die REST-Architektur die Verschlüsselung der Transportebene. TLS kann eine gegenseitige Authentifizierung von Client und Server vornehmen. Die Authentifizierung des Users selbst kann über die HTTP Authentication erfolgen, die beispielsweise eine Authentifizierung über Benutzername und Passwort (Basic Authentication) oder Hashwert

(Digest Authentication) ermöglicht. Die Grenzen von TLS werden klar, wenn Intermediäre ins Spiel kommen. TLS bietet nicht, wie beispielsweise WS-Security, eine Ende-Zu-Ende-Sicherheit. Weiters muss man der WS-Security zugute halten, dass sie den standardisierten Einsatz eines umfangreichen Sets an Sicherheitskonzepten ermöglicht, angefangen von simplen Passwörtern über X.509 Zertifikate bis hin zu Kerberos Tokens. Allerdings spricht nichts dagegen diese Konzepte auch auf REST zu übertragen. Schließlich ist die HTTP Authentication erweiterbar bzw. sind XML-Verschlüsselung und -Signatur bei einer REST-Architektur mit einem XML-basierten Nachrichtenformat ebenfalls einsetzbar. Sollten für eine Anwendung aber keine außergewöhnlichen Sicherheitsanforderungen, wie beispielsweise die teilweise Verschlüsselung einer Nachricht für einen speziellen Intermediär notwendig sein, sind die Sicherheitsmechanismen von HTTP über TLS ausreichend [1, 37].

3. Analyse und Vergleich mobiler Technologien

Bei der momentan heterogenen Landschaft an mobilen Betriebssystemen steht man zu Beginn des Entwicklungsprozesses stets vor der Entscheidung, welche Plattformen unterstützt werden sollen. Die Entscheidung für eine bestimmte Technologie sollte bei der Einführung eines kommerziellen Produktes neben technologischen Argumenten auch wesentlich durch Trends innerhalb einer Branche geprägt sein, um eine möglichst hohe Akzeptanz bei den Anwendern zu erzielen. Gerade der Gesundheitsbereich ist geprägt von Personal, das bzgl. Ausbildungsstand und Wissen stark differiert und unterschiedliche Anforderungen an die geplante Applikation stellt. Um die Widerstände gegen die neue Technologie möglichst gering zu halten, müssen Plattform und Endgerät so gewählt werden, dass sich die Benutzergruppen bestmöglich damit identifizieren können. Vor diesem Hintergrund werden in Abschnitt 3.1 aktuelle Markttrends bei mobilen Plattformen im Gesundheitsbereich analysiert. Die dabei identifizierten Marktführer werden in Abschnitt 3.2 im Hinblick auf technologische Anforderungen verglichen. Abschnitt 3.3 fasst die wichtigsten Punkte der Analyse zusammen und begründet die Wahl der Plattform für unsere Gesundheitsanwendung.

3.1 Markttrends bei mobilen Gesundheitsanwendungen

Eine Umfrage, die 2012 von Manhattan Research [70] für den US-amerikanischen Markt durchgeführt wurde, zeigt, dass für Mediziner Apples iPhone [48] eindeutiger Marktführer ist, gefolgt von RIMs Blackberry [79] und Googles Android [56]. Als

Hauptgrund für die Verbreitung des iOS-Betriebssystems wird unter den Medizinern die hohe Benutzerfreundlichkeit genannt [53, 84]. Liu [28] zeigt, dass zum heutigen Stand in Apples App Store weitaus mehr Gesundheitsapplikationen vorhanden sind als in vergleichbaren Märkten anderer Plattformen. So scheint sich der mHealth-Markt in den letzten Jahren vermehrt auf iOS zu konzentrieren.

Allerdings wurde im Sommer 2010 unter dem Titel "Global mHealth Developer Survey" [82] eine weltweite Umfrage unter Unternehmen des mHealth-Bereichs durchgeführt. Die Prognose dieser Studie besagt, dass Android den derzeitigen Plattform-Marktführer für mobile Gesundheitsapplikationen, iOS, bis 2015 überholt haben wird. Android und iOS sind also derzeit im Bereich der mobilen Gesundheitsapplikationen die vorherrschenden mobilen Betriebssysteme, wobei Android eine bessere Zukunftsprognose ausgestellt wird.

Die Bedeutung von Android wird noch größer, wenn man bedenkt, dass die Umfrage von Manhatten Research [70] sich ausschließlich auf den US-amerikanischen Gesundheitsmarkt und die Benutzergruppe der Mediziner konzentriert. Der globale Marktanteil von iOS ist bereits wesentlich kleiner als jener von Android, schränkt man die Sicht nicht auf die Gruppe US-amerikanischer Mediziner ein. Die von der International Data Corporation [63] für das zweite Quartal 2012 veröffentlichte Statistik zeigt, dass iOS mit einem weltweiten Marktanteil von 16,9% deutlich hinter Android mit 68,1% liegt [64]. Gut situierte Mediziner nutzen, die im Hochpreissegment angesiedelten Produkte von Apple außerdem sicher häufiger als Therapeuten und Pflegepersonal mit geringerem Einkommen. Für die geplante Gesundheitsapplikation sind alle drei Berufsgruppen potentielle Nutzer, wobei Therapeuten und Pflegepersonal eine größere Rolle spielen werden, was die Android-Plattform noch interessanter macht. Ein häufig diskutiertes Thema im Zusammenhang mit der beruflichen Nutzung mobiler Technologien ist das BYOD-Prinzip oder "Bring Your Own Device". Nach dieser Philosophie sollen Mitarbeiter ihr privates Endgerät auch beruflich einsetzen können. Für Mitarbeiter bringt dies zwar viele Vorteile, für die Unternehmung aber ebenso viele Risiken mit sich. Denn so kommt es zur Mischung privater und sensibler Unternehmensdaten auf oft unzureichend gesicherten Endgeräten. Vor dem Hintergrund vermehrter Hacker-Angriffe auf mobile Plattformen und besonders schutzwürdigen Gesundheitsdaten, schließen wir "BYOD" für unsere mobile Anwendung vorerst aus. Dennoch glauben wir, dass Mitarbeitern die Nut-

zung der mobilen Geschäftsanwendung wesentlich leichter fällt, ist ihnen das mobile Betriebssystem bereits aus dem privaten Bereich bekannt.

Abgesehen von der Vorherrschaft von Android und iOS hat die "Global mHealth Developer Survey" [82] weiters gezeigt, dass die Technologie der Zukunft in diesem Bereich HTML5 sein wird. Diese Technologie bringt den enormen Vorteil, dass mit der Entwicklung einer einzigen Anwendung und unter Wegfall einer Installation alle mobilen Plattformen unterstützt werden können. Die einzige Voraussetzung, die die Plattform erfüllen muss, ist das Ausführen eines Browser.

Ein weiterer Aspekt, der neben den Trends mobiler Plattformen zu beachten ist, ist die Ergonomie des Endgeräts. Es stellt sich die Frage, ob Tablet oder Smartphone im professionellen Gesundheitsbereich bevorzugt werden. Laut der Studie von global mHealth [82] liegt das größte Potential in Smartphones gefolgt von Tablets, wobei hier erwähnt werden muss, dass Gesundheitsapplikationen allgemein und nicht Gesundheitsapplikationen für den professionellen Bereich im Speziellen untersucht wurden. Butz und Kruger [11] bestätigen dieses Ergebnis für professionelle Gesundheitsapplikationen, wobei als größter Vorteil im Vergleich zu Tablets und Laptops das geringere Gewicht angesehen wird.

Für die geplante Gesundheitsapplikation sind somit Android, iOS und HTML5, als die derzeit stärksten Technologien am mobilen Gesundheitsmarkt, verstärkt zu untersuchen.

3.2 Technologische Charakteristiken

Die folgenden Abschnitte vergleichen die aktuell wichtigsten mobilen Technologien des Gesundheitsbereiches - Android, iOS und HTML5 - hinsichtlich relevanter technologischer Charakteristiken. Für einen Vergleich dieser Plattformen hinsichtlich der Unterstützung unterschiedlicher Bildschirmgrößen bzw. für Informationen zur Entwicklung plattformübergreifender Anwendungen mittels sogenannter Cross-Platform-Development-Tools verweisen wir den interessierten Leser auf die Arbeit von Holzinger u. a. [21].

Zum Zeitpunkt unserer Recherche und Analyse ist das Betriebssystem iOS in der Version 5 bzw. Betaversion 6 verfügbar. Die aktuelle Version der Android-Plattform

ist 4.1 (Jelly Bean). Der HTML5-Standard ist zum Zeitpunkt der Recherche noch nicht abgeschlossen.

Offenheit der Plattform

Die mobile Plattform soll bei der Entwicklung und Distribution der Anwendung möglichst wenig Einschränkungen, wie beispielsweise obligatorische Designrichtlinien, eingeschränkte Distributionskanäle oder standardisierte Softwaretests, auferlegen.

Einen sehr offenen Ansatz verfolgt hier Googles Android. Lizenzen werden an Mobiltelefon-Hersteller vergeben, um Android auf einer großen Anzahl verschiedener Endgeräte verfügbar zu machen. Design-Richtlinien werden zwar vorgeschlagen, ihre Einhaltung aber nicht geprüft. Softwaretests der im Google Playstore (vormals Android Market) veröffentlichten Applikationen werden nicht durchgeführt. Ebenso ist eine Distribution der Anwendungen ohne Zwischenschaltung des Playstores, beispielsweise über Email oder Webserver, möglich. Die Entwicklung von Anwendungen unter Android ist somit keinen Einschränkungen unterworfen.

Anders als Google vergibt Apple keine iOS Lizenzen an Mobiltelefon-Hersteller. Das Betriebssystem wird nur auf unternehmenseigener Hardware vertrieben, was die Vielfalt an Endgeräten einschränkt. Auch bei der Distribution von Applikationen verfolgt Apple einen sehr restriktiven Ansatz im Vergleich zur Konkurrenz, indem die Verteilung von Applikationen nur über Apples AppStore möglich ist. Bevor eine Applikation zum Download bereitgestellt wird, untersucht Apple die Applikation auf Einhaltung gewisser Richtlinien. Details über die durchgeführten Tests und geprüften Anforderungen sind nicht bekannt. Entspricht eine Applikation den geprüften Anforderungen allerdings nicht, verweigert Apple die Veröffentlichung dieser. Für Unternehmen bietet Apple einen Distributionskanal an, der am AppStore vorbeiführt, das sogenannte “iOS Enterprise Program”. Über dieses Programm können Unternehmen ihre Anwendungen exklusiv auf die Endgeräte ihrer Mitarbeiter verteilen [61]. Die Verteilung der unternehmenseigenen Applikationen kann somit über einen einfachen Webserver oder über eine Mobile Device Management-Lösung erfolgen [60].

HTML5-Webapps werden über einen Browser am mobilen Endgerät angesteuert

und laufen somit auf allen mobilen Plattformen. Dadurch sind sie von der Unternehmenspolitik der Plattformanbieter unabhängig.

Hardwarezugriffe und Applikationsübergreifende Kommunikation

Eine mobile Applikation entwickelt vor allem durch die Integration gerätespezifischer Hardware, wie beispielsweise der Kamera, einen großen Mehrwert für die Nutzer. Um eine einfache und rasche Erweiterbarkeit einer mobilen Applikation zu gewährleisten, ist weiters die Kommunikation mit und die Integration von bereits bestehenden Applikationen unabdingbar. Daher sollte ein Zugriff auf Funktionalitäten anderer Applikationen, wie bspw. die eines Barcode-Scanners zur Medikamentenzuordnung, möglich sein.

Auf den mobilen Plattformen Android und iOS sind sowohl Hardware-Zugriffe als auch die Integration dritter Anwendungen möglich. Während Android für Hardware-Zugriffe ein Berechtigungssystem vorsieht, haben bei iOS alle Anwendungen grundsätzlich Zugriff auf jegliche Hardware. Bisher erforderten lediglich sogenannte "Location Services" eine explizite Berechtigung durch den Benutzer, zusätzliche Berechtigungsmechanismen sind aber für künfitge Versionen von iOS angedacht [41]. Die Integration dritter Apps erfolgt in iOS über eine URL-basierte Methode zur Kommunikation zwischen Anwendungen[65], während Android hierfür eigene Komponenten (Intents und ContentProvider) bzw. ebenfalls Berechtigungen zur Informationsabfrage vorsieht [55].

Hardwarezugriffe einer HTML5-Webapplikation sind ebenfalls möglich. Dazu muss der Browser der jeweiligen Plattform allerdings die sogenannte Media Capture API [85] unterstützen, die beispielsweise Zugriffe auf Kamera und Audio erlaubt. Nicht geplant ist eine Unterstützung von Bluetooth. Das bedeutet eine Einschränkung für unsere Anwendung, vor allem was den Zugriff auf externe medizinische Geräte betrifft. Weiters ist eine Kommunikation mit anderen Anwendungen des Geräts bei einer HTML5-Webapplikation nicht möglich, was eine zusätzliche Einschränkung darstellt.

Offline-Funktionalität

Um den Nutzern auch bei Netzausfall die Arbeit mit der Applikation zu ermöglichen, muss zumindest eine eingeschränkte Offline-Funktionalität sichergestellt werden. Dazu ist die lokale Speicherung von Daten nötig. Sowohl iOS als auch Android nutzen die leichtgewichtige Datenbank SQLite. Apples iOS bietet das Framework "Core Data" als "Object Relational Mapping"-Framework (ORM) an [90]. Android stellt nativ kein ORM-Tool zur Verfügung, sondern setzt auf einen Cursor-basierten Ansatz, mit dem Informationen aus der Datenbank abgefragt werden.

Webapplikationen haben bisher eine bestehende Internetverbindung vorausgesetzt. Das ändert sich allerdings mit dem HTML5-Standard, der offline Webapplikationen unterstützt. Dazu werden in der sogenannten Cache-Manifest-Datei Ressourcen festgelegt, die offline verfügbar sein sollen [35]. Weiters ermöglicht HTML5 die lokale Speicherung von Daten innerhalb des Webbrowsers. Diese Daten sind über die Existenz des Browser-Prozesses hinaus persistent und werden – anders als Cookies – nie zum Webserver transportiert. Der Standard bietet hier drei Möglichkeiten [35]:

- **HTML5 Storage:** Speicherung von Schlüssel-Wert-Paaren.
- **Web SQL Database:** Wrapper um eine SQL-Datenbank. Abfagen sind über JavaScript möglich.
- **Indexed Database API:** Die Indexed Database API, vormals WebSimpleDB, stellt einen Object Store bereit, der aktuell aber nur von wenigen Browsern unterstützt wird.

Multithreading

Nachdem die Client-Applikation eine generelle Kommunikation sowie eine Synchronisation der lokalen Datenbestände zum Backend-Server erfordert, ist die Fähigkeit der Client-Plattform bzw. -Technologie zum Multithreading unabdingbar. Ohne diese Möglichkeit wäre die Benutzeroberfläche der Applikation beispielsweise für die Dauer des Datenabgleichs blockiert. Während moderne Betriebssysteme, wie

iOS und Android, Multithreading vorsehen, war dies mit Webapplikationen bisher schwierig. HTML5 führt das Konzept der sogenannten “WebWorker” ein. Diese arbeiten unabhängig vom Thread des User Interfaces und können mit diesem kommunizieren [80]. Somit erfüllt auch eine HTML5-Webapplikation die Anforderung des Multithreadings.

Mobile Device Management (MDM)

Zur Integration mobiler Endgeräte und Anwendungen in die Geschäftsprozesse ist die Konfiguration und Überwachung dieser über ein sogenanntes Mobile Device Management (MDM) unerlässlich.

Weder Android noch iOS bieten derzeit eine native MDM-Lösung an. Allerdings existieren für beide Plattformen Lösungen von Drittanbietern, über die beispielsweise die zwingende Nutzung eines Passwortes oder eines Gerätelocks konfigurierbar ist. Die Konfiguration von Endgeräten über eine HTML5-Webapplikation ist nicht möglich.

Sicherheitsaspekte

Gesundheitsapplikationen verarbeiten und speichern sensible Daten. Die Plattform der Anwendung muss daher grundlegende Sicherheitsaspekte zum Schutz dieser Daten aufweisen.

Die sichere Speicherung von Passwörtern oder privaten Schlüsseln wird in iOS durch sogenannte Keychains gelöst. Die sichere Speicherung von Authentifizierungsdaten auf iOS-Geräten ist somit gewährleistet [62, 73]. In Android ermöglicht der Credential Storage die sichere, verschlüsselte Speicherung von Zertifikaten. Benutzernamen und Passwörter werden typischerweise auf den internen Speicher einer Applikation geschrieben, wodurch sie, auf nicht gerooteten Geräten, nur für diese Applikation sichtbar sind [45].

iOS sieht weiters eine zwingende Device-Verschlüsselung vor [62]. In Android 3.0 wurde ebenfalls eine Verschlüsselung des Dateisystems realisiert, die optional aktiviert werden kann [44]. Der HTML5-Standard sieht derzeit keine Verschlüsselung von lokal gespeicherten Daten vor.

3.3 Entscheidungsfindung

Tabelle 3.1 zeigt eine stark vereinfachten Übersicht der untersuchten Plattformen und Technologien. Die Entscheidung zwischen iOS und Android ist eine schwierige. Beide Plattformen erfüllen die für die geplante Applikation geforderten Kriterien. Durch die große Marktrelevanz beider Plattformen wird eventuell eine Unterstützung beider notwendig sein. Diese Entscheidung ist schlussendlich eine, die mit der Unternehmensstrategie Hand in Hand gehen muss. HTML5 ist für unsere geplante Applikation vor allem aus Gründen fehlender, applikationsübergreifender Kommunikation und geringer Sicherheitsmechanismen lokal gespeicherter Daten auszuschließen. Auch Bloice u. a. [7] empfehlen die Entwicklung einer nativen Applikation, werden plattformspezifische Features benötigt. Nachdem die Markttrends zeigen, dass Android derzeit Marktführer ist und von der mHealth-Studie [82] die größere Bedeutung für die Zukunft zugesprochen bekommen hat, werden wir im Rahmen dieser Arbeit die Entwicklung für Android fokussieren. Die Unterstützung von iOS ist für einen späteren Zeitpunkt angedacht.

	iOS	Android	HTML5
Offenheit der Plattform	teilweise	gegeben	gegeben
Hardwarezugriff	gegeben	gegeben	teilweise
Interapplikationskommunikation	gegeben	gegeben	nicht
Offline-Funktionalität	gegeben	gegeben	teilweise
Multithreading	gegeben	gegeben	gegeben
Mobile Device Management	teilweise	teilweise	nicht
Sicherheitsaspekte	gegeben	gegeben	teilweise

Legende
gegeben | teilweise gegeben | nicht gegeben

Tabelle 3.1: Gegenüberstellung der Client-Plattformen und -Technologien.

4. Nutzerzentriertes Design im Softwareentwicklungsprozess

Nachdem wir in Kapitel 2.1 die wichtigsten Methoden des Usability Engineerings definiert haben, zeigen wir im Folgenden, auf welche Weise der Faktor "Usability" in unseren Softwareentwicklungsprozess integriert wurde. Dazu argumentieren wir in Abschnitt 4.1 die von uns gewählte Vorgehensweise. Danach beschreiben wir in Abschnitt 4.2 die praktische Durchführung von Usability-Tests im Rahmen unseres Entwicklungsprozesses.

4.1 Vorgehensweise und Methoden

Obwohl die enorme Bedeutung des Usability Engineerings für Softwareentwicklungen jeder Art bekannt ist, ist der Einsatz formeller Usability-Methoden in der Praxis immer noch mangelhaft. Zahlreiche Forschungsarbeiten beschäftigen sich mit dieser Problematik und identifizieren bspw. straffe Zeitpläne oder hohe Kosten als Ursachen. Den interessierten Leser verweisen wir hier auf [8, 9, 12, 26]. Um dem Faktor der Benutzerfreundlichkeit trotz begrenzter Ressourcen Rechnung zu tragen, integrierten wir eine Usability-Evaluierung von bereits sehr frühen Prototypen, sogenannten Papier-Prototypen, in den Softwareentwicklungsprozess. Usability-Tests erschienen uns als die effektivste Methode, da sie die Einbeziehung der künftigen Nutzer fokussieren und deren Beobachtung bei der praktischen Aufgabenbewältigung ermöglichen. Zudem können Usability-Tests mit Papier-Prototypen bereits vor der eigentlichen Implementierung durchgeführt werden. Besonders in einem schnelllebigen, innovativen Umfeld, das von unklaren Anforderungen geprägt ist, sind Rückmeldungen von Anwendern zu einem sehr frühen Zeitpunkt in der Entwicklung notwendig.

So können Entwicklungsarbeiten verhindert werden, die an den Anforderungen des Marktes vorbeigehen.

Zahlreiche Arbeiten aus dem mobilen Bereich generell (siehe bspw. [34]) und aus dem mHealth-Bereich speziell propagieren den Einsatz von Usability-Tests basierend auf Prototypen. So schreiben beispielsweise Höll u. a. [23] in einer Arbeit im Gesundheitsbereich, dass das nutzerzentrierte Design optimalerweise bei Prototypen ansetzt, da es technisch nicht versierten Personen erst durch das Aufzeigen eines Lösungsweges möglich ist, ein Verständnis für die Problematik zu entwickeln und basierend darauf eigene Vorschläge und Ideen zur Verbesserung einzubringen. Butz u. a. [10] schreiben wiederum, dass eine erstmalige Einbeziehung der Nutzer nach Implementierung eines ersten Prototypen ihrer Krankenhaus-Management-Applikation das Entwicklungsteam unnötig Zeit gekostet hat. Rückblickend empfehlen sie den Einsatz von Papier-Prototypen für eine schnelle, erste Rückmeldung der Nutzer. Auch Holzinger u. a. [20] zeigen ein iteratives Design beginnend bei einfachen Papierprototypen bis hin zur ersten Software-Implementierung eines mobilen Krankenhaus-Fragebogens für Patienten.

Im Rahmen dieser Arbeit wurde die erste Iteration an Tests durchgeführt. Weitere Usability-Tests, beispielsweise basierend auf einer ersten Software-Implementierung der Anwendung, sind, im Sinne eines iterativen Designprozesses, weiterführenden Arbeiten durch FERK Systems vorbehalten.

Nachdem im Rahmen dieser Arbeit nur eine Iteration an Usability-Tests durchgeführt werden konnte, war es umso wichtiger den Teilnehmern bereits möglichst durchdachte Designs zu präsentieren. Auch Rubin und Chisnell [38] betonen, dass die Durchführung von Usability-Tests erst zweckmäßig ist, wenn offensichtliche Usability-Fehler bereits behoben sind. Aus diesem Grund wurden mit den Mitarbeitern von FERK Systems vor der Durchführung der Usability-Tests zusätzlich einige Iterationen an Walk-Throughs durchgeführt. Weiters orientierte sich das Design an den in Abschnitt 2.1.2 beschriebenen Usability-Richtlinien für mobile Anwendungen von Inostroza u. a. [24] sowie an den Design-Richtlinien von Android bzw. iOS.

Das Kernelement unserer Usability-Integration in den Softwareentwicklungsprozess stellten also Usability-Tests in Kombination mit Papier-Prototypen dar. Ziel und Zweck von Usability-Tests unterscheiden sich je nach Entwicklungsstand des Produkts. Nachdem wir uns noch am Beginn der Entwicklung befanden, führten

wir, der Einteilung von Rubin und Chisnell [38] folgend, mit den Usability-Tests eine *explorative* bzw. *formative* Studie durch. Die explorative Komponente unserer Usability-Tests versuchte zu klären, ob die angenommenen Nutzerprofile der Realität entsprachen und ob das entworfene Produkt für den Nutzer tatsächlich eine Arbeitserleichterung bedeutete. Dieser Aspekt wurde durch die Interviews zum Stand der Gesundheitsdatenerfassung in der Praxis zusätzlich betont. Die formative Komponente der Tests sollte vor allem die Frage klären, ob die Designs die Aufgaben der Nutzer optimal unterstützten, wie einfach der Nutzer relevante Informationen auffinden konnte, etc.

Im Rahmen von explorativen Studien sind Usability-Tests sehr informal gestaltet. Die Interaktion zwischen Testperson und Moderator ist sehr intensiv. Häufig eingesetzt wird dabei die sogenannte "Thinking Aloud"-Methode, bei der die Testperson dazu aufgefordert wird, ihre Gedanken zu verbalisieren, während sie eine Aufgabenstellung zu bewältigen versucht. Auch wenn das Verbalisieren von Gedanken für die Testperson eventuell unnatürlich ist und ihr Vorgehen verlangsamt, ist es dem Moderator dadurch möglich zu erkennen, warum bestimmte Aktionen durchgeführt bzw. Fehler gemacht oder vermieden werden [38].

4.2 Usability-Tests mit Papier-Prototypen

In den folgenden Abschnitten beschreiben wir die, im Rahmen dieser Arbeit, durchgeführten Usability-Tests basierend auf Papier-Prototypen. Dazu definieren wir zu Beginn in Abschnitt 4.2.1 Forschungsfragen sowie unsere Testgruppen in Abschhnitt 4.2.2. Anschließend beschreiben wir den Testablauf in Abschnitt 4.2.3 und stellen die Ergebnisse der Tests in Abschnitt 4.2.4 detailliert dar.

4.2.1 Forschungsfragen

Nachdem wir die Usability-Tests mit einem Interview über die gängige Praxis der Gesundheitsdatenerfassung verknüpfen, nehmen wir auch eine entsprechende Einteilung unserer Forschungsfragen vor. Das zusätzliche Interview soll die explorative Komponente unserer Evaluierung stärken. Das heißt, dass wir typische Arbeitsvorgänge im Rahmen der Gesundheitsdatenerfassung aufdecken und unsere angenom-

menen Nutzerprofile sowie den Nutzungskontext bestätigen wollen.

Die Fragen, die es im Rahmen der Interviews zu beantworten gilt, sind folgende:

- Was sind die typischen Arbeitsvorgänge im Zusammenhang mit der Gesundheitsdokumentation eines Patienten im Rahmen der Pflege bzw. Therapie?
- Inwieweit sind die Vorgehensweisen standardisiert? Werden beispielsweise standardisierte Fragebögen eingesetzt?
- Ist der Begriff ICF bekannt bzw. erfolgt ein Einsatz von ICF?
- Inwieweit erfolgt die Dokumentation EDV-gestützt? Werden eventuell bereits mobile Technologien eingesetzt?
- Inwieweit befassen sich Mediziner mit der Pflege- und Therapiedokumentation eines Patienten?

Nachdem die Usability-Tests zu einem sehr frühen Zeitpunkt der Produktentwicklung durchgeführt werden, fokussieren wir die formative Komponente, das heißt grundsätzliche Fragen zur Qualität unseres Designkonzepts. Folgende Fragen sollen durch die Usability-Tests beantwortet werden:

- Sieht der Nutzer in dem geplanten Einsatz von ICF einen Nutzen, d.h. eine Erleichterung für seine Arbeitsvorgänge?
- Werden alle für den Nutzer relevanten Informationen verarbeitet?
- Sind die allgemeine Benutzbarkeit bzw. Navigation für den Nutzer intuitiv?
- Welches Vorwissen bzw. welche Form der Einschulung wird nötig sein, um das Produkt im Berufsalltag einsetzen zu können?

4.2.2 Nutzergruppen

Die potentiellen Nutzer unserer Anwendung werden im Gesundheitsbereich tätig sein. Berufserfahrung oder eine entsprechende Ausbildung aus diesem Bereich waren Grundvoraussetzung für die Teilnahme am Usability-Test. Nachdem ICF unseren Recherchen zufolge in der Praxis noch nicht angewandt wird, ist es schwierig alle

Nutzergruppen exakt zu bestimmen. Aus diesem Grund konzentrieren wir uns in der ersten Iteration der Usability-Tests auf die zwei vermutlich wichtigsten Nutzergruppen, Pflegepersonal und Physiotherapeuten/-innen, und inkludieren als Randgruppe die Mediziner in unsere Untersuchung. Wir kategorisieren unsere Nutzergruppen entsprechend ihrer beruflichen Tätigkeit daher in:

- **Diplomierte Krankenpfleger/-innen:** Zum Berufsalltag von Krankenpflegern/innen gehört die Dokumentation des Gesundheitszustandes von Patienten. Diese Berufsgruppe ist somit eine der wichtigsten Zielgruppen für unsere geplante Applikation, speziell für die Vorgänge der Gesundheitsdatenerfassung und -analyse.

- **Physiotherapeuten/-innen:** Physiotherapeuten/-innen sind für die Mobilisierung von Patienten verantwortlich. Eine entsprechende Erfassungsmöglichkeit des Gesundheitszustandes ist bei der Ausübung dieser Tätigkeit genauso relevant wie die Möglichkeit den Therapiefortschritt zu analysieren.

- **Mediziner:** Mediziner tragen die generelle Verantwortung für die Genesung des Patienten. Sie geben unter Umständen die Anweisung zur Dokumentation relevanter Gesundheitsdaten, werden sie jedoch üblicherweise nicht selbst durchführen.

Eine zusätzliche Kategorisierung der Nutzergruppen anhand der Faktoren "Alter" oder "Erfahrungen mit mobilen Endgeräten" wird an dieser Stelle vernachlässigt, um das Testdesign zu Beginn des Entwicklungsprozesses möglichst simpel zu halten. Das Design der Anwendung wird noch von unklaren Anforderungen und einem unklaren Nutzungskontext tangiert. Im Sinne eines iterativen Prototypings in unserem Entwicklungsprozess soll diese erste Usability-Evaluierung in einem sehr frühen Stadium grundlegende Rückmeldungen zur Tauglichkeit und Einschätzung des Designs bringen, was unserer Meinung nach bereits durch eine geringere Anzahl an Testpersonen erreicht werden kann. Auch in der Wissenschaft ist die Frage der notwendigen Anzahl an Testpersonen für Usability-Evaluierungen intensiv behandelt worden. Diese Anzahl hat sowohl ökonomische als auch wissenschaftliche Auswirkungen [4]. Je mehr Testpersonen, desto höher sind die Kosten für die Unternehmung und desto relevanter sind die Ergebnisse der durchgeführten Tests. Die

Herausforderung besteht somit darin, mit der geringsten Anzahl an Testpersonen so viele Usability-Probleme wie möglich aufzudecken. Laut Bastien [4] wird die optimale Anzahl an Testpersonen seit den Neunziger Jahren diskutiert, wobei nach damaliger Meinung vier bis fünf Testpersonen 80-85% aller Usability-Probleme einer Anwendung aufdecken können. Mittlerweile wurden Studien veröffentlicht, die eine weitaus höhere Zahl an Testpersonen empfehlen [4]. Die Frage der optimalen Anzahl wird stark durch die Art der Anwendung und die Komplexität der durchzuführenden Aufgaben der Usability-Tests beeinflusst und eine endgültige Antwort ist bislang ausständig. Bis heute orientieren sich viele erfolgreiche Forschungs- und Praxis-Arbeiten, wie beispielsweise [19, 22], an der in den Neunziger Jahren vorgeschlagenen Anzahl von vier bis fünf Testpersonen. Auch Rubin [38] ist der Meinung mit dieser Anzahl einen Großteil der Usability-Probleme aufdecken zu können, wobei sie seiner Meinung nach bei einer iterativen Durchführung von Usability-Tests auch unterschritten werden darf. User-Tests, die die Thinking-Aloud-Methode einsetzen, bedürfen laut Holzinger [18] sogar nur drei Testpersonen pro Nutzergruppe als Minimum. Für unsere wichtigsten Nutzergruppen, die Krankenpfleger/-innen und Physiotherapeut/-innen, setzten wir diese Anzahl als Maßstab an, wie Tabelle 4.1 zeigt. Die Benutzergruppe der Mediziner wurde als Randgruppe in die Usability-Evaluierung integriert.

Nutzergruppen	**Anzahl der Testpersonen**
Diplomierte Krankenpfleger/-innen	4
Physiotherapeuten/-innen	3
Mediziner/-innen	1

Tabelle 4.1: Anzahl der Testpersonen pro Nutzergruppe.

4.2.3 Testablauf und -aufbau

Nach einem Interview zur gängigen Praxis der Gesundheitsdatenerfassung, werden die grundlegende Idee unserer Anwendung sowie das Konzept von ICF erklärt. Nach einer Demonstration zu Papierprototypen-Tests und der Thinking-Aloud-Methode, erfolgt die eigentliche Durchführung des Usability-Tests. Abbildung 4.1 zeigt die Testumgebung.

Abbildung 4.1: Testumgebung der Usability-Tests.

Die ersten Aufgabenstellungen des Tests sollten den Teilnehmern das im Vorgespräch geklärte Konzept von ICF noch einmal in Erinnerung rufen bzw. Zusammenhänge zu ICD klären. Danach wurden Probleme gewählt, die unserer Meinung nach die gängisten Arbeitsvorgänge im Berufsalltag widerspiegeln. Die Aufgaben wurden sowohl sequentiell - das heißt der Ausführungsreihenfolge im typischen Berufsalltag entsprechend - als auch nach Schwierigkeitsgrad gereiht. Nach der Durchführung des eigentlichen Usability-Tests wurde im Rahmen einer Diskussion auf kritische Punkte aus dem Test und persönliche Rückmeldungen zum Design eingegangen.

Im Folgenden werden die Problemstellungen und Abschlusskriterien der einzelnen Testaufgaben zusammengefasst.

Aufgabe 1. Die erste Aufgabe hielt die Teilnehmer dazu an das Konzept der ICF in Erinnerung zu rufen und gedanklich auf unsere Anwendung umzulegen. Somit sollten bestmögliche Voraussetzungen für den restlichen Testverlauf geschaffen werden. Die Testteilnehmer erhielten bei dieser Aufgabe das Design eines Patientenblattes. Sie sollten basierend auf diesem Design verbal beschreiben, welche Funktionalitäten sie hinter den einzelnen ICF- und ICD-Schaltflächen vermuteten.

Abschlusskriterien: Beendigung der verbalen Erläuterung, nachdem die Be-

griffe ”Krankheitsbild“, ”Diagnosen“, sowie ”ICF-Erfassung“ und ”ICF-Auswertung“ den entsprechenden Funktionalitäten zugeordnet wurden.

Aufgabe 2. Bei der zweiten Aufgabe wurden die Testpersonen dazu aufgefordert, dem Patienten eine neue ICD-Diagnose sowie ein entsprechendes Krankheitsbild zuzuordnen. Obwohl diese Aufgabe keinen typischen Arbeitsvorgang von diplomierten Pflegekräften oder Physiotherapeuten/-innen darstellt, sollte so der Zusammenhang zwischen ICD-Diagnosen und Krankheitsbildern zur späteren Gesundheitsdokumentation verdeutlicht werden. Der Begriff des ”Krankheitsbildes“ war bereits Gegenstand des Vorgesprächs zum Test.

Abschlusskriterien: Sowohl die geforderte Diagnose als auch das geforderte Krankheitsbild sind der fiktiven Patientin zugeordnet worden.

Aufgabe 3. Diese Aufgabe befasste sich mit dem Kern der geplanten Anwendung, der Gesundheitsdatenerfassung basierend auf ICF. Die Teilnehmer sollten die ICF-Beurteilung des Codes ”d450 Gehen“ vornehmen. Dazu erhielten sie sowohl Angaben zur Beurteilung durch das Pflegepersonal als auch zur Einschätzung des Patienten selbst.

Abschlusskriterien: Die Detail-Informationen zu den möglichen Beurteilungswerten wurden abgefragt. Der Code ”d450 Gehen“ wurde mit dem Wert 1 bzw. 2 beurteilt.

Aufgabe 4. Die letzte Aufgabe beschäftigte sich mit der Visualisierung und Auswertung der ICF-Erfassungen. In der ersten Teilaufgabe sollten die ICF-Codes gefunden werden, deren Bewertung sich in letzter Zeit verbessert hatte. In der zweiten Teilaufgabe sollten detaillierte Informationen über den Verlauf des ICF-Codes ”d450 Gehen“ abgefragt werden.

Abschlusskriterien:

1. Nennung und Anzeige der Codes, die über die letzten drei ICF-Erfassungen eine positive Entwicklung genommen haben.

2. Anzeige des Detail-Diagramms zum Code ”d450 Gehen“ über die letzten sechs Monate hinweg.

4.2.4 Ergebnisse und Empfehlungen

Dieses Kapitel beschreibt die Ergebnisse der durchgeführten Usability-Evaluierung. Zu Beginn fassen wir die wichtigsten Ergebnisse der duchgeführten Interviews zusammen. Im Anschluss daran stellen wir die Ergebnisse der Usability-Tests sowie die darauf basierenden Design-Empfehlungen detailliert dar.

Erkenntnisse aus den Interviews

Die Interviews bestätigten größtenteils unsere Recherchen über die gängige Praxis und die typischen Arbeitsvorgänge der potentiellen Nutzer unserer Anwendung. Die folgenden Abschnitte fassen die Ergebnisse angelehnt an die in Abschnitt 4.2.1 definierten Forschungsfragen zusammen.

Praxis der Gesundheitsdokumentation. Die typischen Arbeitsvorgänge im Zusammenhang mit der Gesundheitsdokumentation werden vom Großteil der Befragten als sehr langwierig und schreibintensiv empfunden. Es existieren sowohl in der Pflege als auch in der Therapie standardisierte Skalen und Fragebögen. Als Beispiele in der Pflege wurden die Braden-Skala, der Barthel-Index, das Exsikkose-Risiko, die Morse-Fall-Skala und die Pflegeabhängigkeitsskala genannt. Die Physiotherapeuten hoben neben standardisierten Fragebögen vor allem den Einsatz der visuellen Analogskala (VAS) hervor, die zur subjektiven Einschätzung der Schmerzintensität verwendet wird. Diese standardisierten Werkzeuge erfassen jeweils einen Teilaspekt des Zustands eines Patienten. Relevante Zusatzinformationen, die von den Fragebögen nicht erfasst werden, werden als Freitext notiert. Diverse Fragebögen zum Gesundheitszustand werden in regelmäßigen Abständen erfasst, die Periode der Erfassung variiert aber je nach Patient sehr stark. Neben diesen Fragebögen und Skalen ist der Großteil der Dokumentationsarbeit als Freitext gestaltet. Beispielsweise wird die gesamte Pflegedokumentation, also die Dokumentation der geplanten und durchgeführten pflegerischen Maßnahmen, als Freitext erfasst. Ebenso werden die durchgeführten Maßnahmen sowie die Therapieziele eines Patienten im Bereich der Physiotherapie als Freitext dokumentiert.

Abgrenzung der Aufgabengebiete der Nutzergruppen. Während diplomierte Krankenpfleger/-innen und Therapeuten/-innen die ICD-Diagnosen der Mediziner teilweise als Ausgangspunkt ihrer Arbeit ansehen, hat der befragte Arzt im Interview betont, dass Mediziner mit den Skalen und Fragebögen der Pflege bzw. Therapie in der Regel nicht arbeiten. Unser Hauptaugenmerk im Rahmen der Usability-Tests ist also eindeutig auf die Gruppen der diplomierten Krankenpfleger/-innen und Physiotherapeuten/-innen zu legen.

ICF in der Praxis. Die ICF wird von den Befragten im Berufsalltag nicht angewandt. Ein Begriff ist sie den meisten nur aus der Ausbildungszeit. Das trifft vor allem auf die Gruppe der Physiotherapeuten/-innen zu, bei denen die ICF in der Ausbildung ein großes Thema war. Dennoch wurde keiner der Befragten in der Praxis mit dieser Klassifizierung konfrontiert.

IT-Unterstützung. Mobile Technologien kommen im Berufsalltag der Befragten noch nicht zum Einsatz. Die mobilste Lösung wurde am Beispiel des LKH Graz genannt, wo es üblich ist, einen Laptop auf einem Rollwagen bei der Visite mitzuführen. Doch auch der IT-Einsatz abseits mobiler Technologien scheint im Bereich der Gesundheitsdokumentation nicht stark ausgeprägt zu sein. So wird von Befragten teilweise beschrieben, dass standardisierte Fragebögen nicht automationsunterstützt verarbeitet, sondern vor der Datenerhebung ausgedruckt und handschriftlich ausgefüllt werden. Für jeden Patienten häufen sich so große Papierakten an Pflege- und Gesundheitsdokumentation an.

Erkenntnisse aus den Usability-Tests

Zu Beginn dieses Abschnitts zeigen wir eine kurze, quantitative Zusammenfassung der Testergebnisse. Unser Fokus liegt aber auf den qualitativen Rückmeldungen der Testpersonen zu unserem Designkonzept, auf die wir im Anschluss detailliert eingehen werden.

Abbildung 4.2 zeigt eine prozentuelle Aufstellung über die erfüllten Testaufgaben mit und ohne Hilfestellung. Im Schnitt konnten sechs von acht Teilnehmern eine Aufgabe erfüllen. Tabelle 4.2 zeigt uns die Gegenüberstellung der Testergebnisse unserer potentiell wichtigsten Zielgruppen, während in Tabelle 4.3 die Ergebnisse

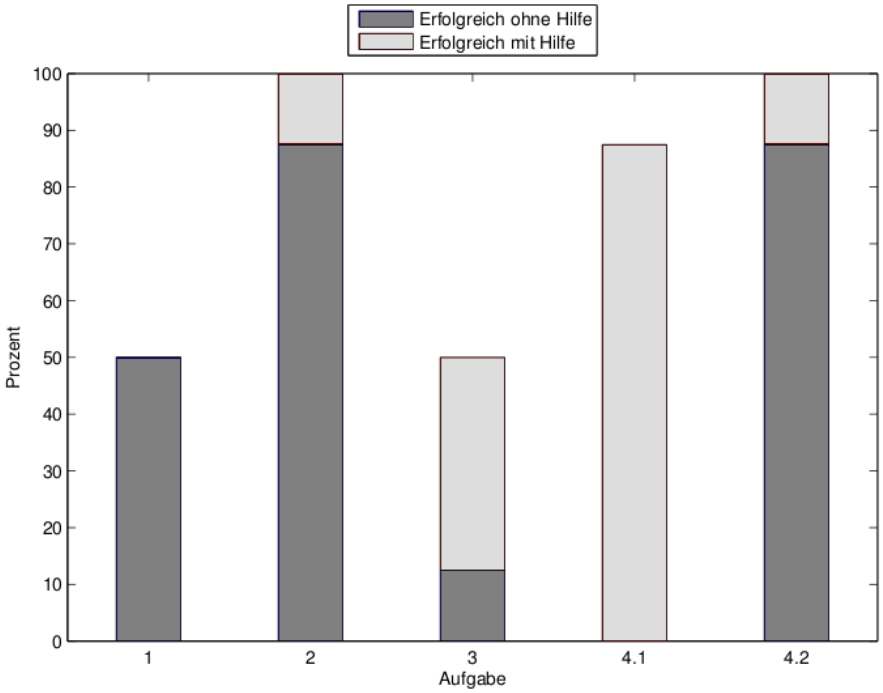

Abbildung 4.2: Erfolgsraten bei den einzelnen Testaufgaben.

nach Smartphone-Besitz der Testpersonen aufgeschlüsselt sind. Auffallend ist, dass die Physiotherapeuten im Vergleich zum Pflegepersonal durchwegs bessere Ergebnisse lieferten. Dasselbe kann für die Smartphone-Besitzer unter den Testpersonen im Vergleich zu Nutzern herkömmlicher Handys beobachtet werden. Zu beachten ist allerdings, dass 100 % der getesteten Physiotherapeuten ein Smartphone besitzen, während das nur für 50 % der getesteten Krankenpfleger der Fall ist. Dieser Zusammenhang darf bei der Interpretation der Tabellen 4.2 und 4.3 nicht außer Acht gelassen werden.

	Dipl. Pflegepersonal		**Physiotherapeuten/-innen**	
Aufgabe	**ohne Hilfe**	**mit Hilfe**	**ohne Hilfe**	**mit Hilfe**
Aufgabe 1	50 %	0 %	75 %	0 %
Aufgabe 2	75 %	25 %	100 %	0 %
Aufgabe 3	25 %	25 %	75 %	0 %
Aufgabe 4.1	0 %	75 %	0 %	100 %
Aufgabe 4.2	100 %	0 %	100 %	0 %

Tabelle 4.2: Ergebnisse nach Berufsgruppen.

Nach der kurzen Übersicht über die Testergebnisse gehen wir in den folgenden Abschnitten näher auf die Rückmeldungen zu den einzelnen Aufgabenstellungen ein.

	Smartphone		kein Smartphone	
Aufgabe	**ohne Hilfe**	**mit Hilfe**	**ohne Hilfe**	**mit Hilfe**
Aufgabe 1	75 %	0 %	0 %	0 %
Aufgabe 2	100 %	0 %	67 %	33 %
Aufgabe 3	20 %	60 %	0 %	0 %
Aufgabe 4.1	0 %	100 %	0 %	67 %
Aufgabe 4.2	100 %	0 %	67 %	33 %

Tabelle 4.3: Ergebnisse nach Smartphone-Nutzung.

Aufgabe 1. Abbildung 4.3 zeigt das Design eines Patientenblattes. Das schlechte Ergebnis einer 50-prozentigen Erfolgsrate ist hier vor allem auf den Begriff des Krankheitsbildes zurückzuführen. Dieser Begriff konnte von den Testnutzern nicht der dahinterstehenden Funktionalität unserer Anwendung zugeordnet werden. Im Vorgespräch wurde unser Verständnis dieses Begriffes grundsätzlich geklärt. Er bezeichnet spezielle Sets an ICF-Codes, die abgestimmt auf Diagnosen bzw. Diagnosegruppen erstellt werden. Für die Testpersonen scheint der Begriff "Krankheitsbild" derart verwirrend gewesen zu sein, dass viele bereits Minuten nach der Abklärung keine Assoziation mehr dazu hatten. Die meisten setzten den Begriff mit dem der Diagnose gleich. Im Vorgespräch gab ein Großteil der Personen bereits an, "Krankheitsbild" in diesem Zusammenhang nicht als passend zu empfinden. Einen Gegenvorschlag konnten die Testpersonen, vermutlich aufgrund der mangelnden Kenntnis über ICF, nicht bringen. Allerdings empfand der Großteil den Begriff "ICF-Diagnosegruppen" als passender.

Empfehlung: An dieser Stelle sollte künftig noch an einer eindeutigen Begriffswahl gearbeitet werden, um auch bei Personen mit geringen Kenntnissen zu ICF eine eindeutige Assoziation zu wecken.

Aufgabe 2. Die Designs für die Erfassung einer Diagnose und eines Krankheitsbildes werden ausschnittsweise in den Abbildungen 4.4 und 4.5 dargestellt. Die Aufgabe konnte von allen Testteilnehmern erfolgreich durchgeführt werden. Die einzige Hilfestellung, die gegeben werden musste, war auf fehlende Erfahrung mit Smartphones zurückzuführen und unabhängig von unserem spezifischen Design. Zu Beginn des Designprozesses war nicht klar, wie eng die Erfassung von ICD-Diagnosen und die Erfassung von *ICF-CoreSets* im Berufsalltag zusammenhängen werden. Die Inter-

Abbildung 4.3: Patientenblatt.

views haben gezeigt, dass Pflegekräfte und Physiotherapeuten/-innen ihre Dokumentationsarbeit teilweise basierend auf den medizinischen Diagnosen, aber nicht vollständig an diese geknüpft, durchführen. Daher scheint uns die Erfassung von *ICF-CoreSets* unabhängig von Diagnosen, aber mit einem entsprechenden Hinweis (siehe Abbildung 4.5) passend. Die guten Ergebnisse dieser Aufgabenstellung unterstreichen zudem, dass der Zusammenhang der beiden Kategorien durch das Design klar wird und die Handhabung bzw. Navigation intuitiv ist.

Aufgabe 3. Abbildung 4.6 zeigt das Design für die ICF-Dokumentation. Die schlechte Erfolgsquote von 50 % bei dieser Aufgabe ist vor allem darauf zurückzuführen, dass die Hälfte der Testpersonen die hinterlegte Zusatzinformation für die Beurteilungswerte des ICF-Codes nicht beachtete. Die Abfrage dieser Zusatzinformation war ein Abschlusskriterium für diese Aufgabe. Bei einer ICF-Erfassung ohne dieses Kriterium läge die Erfolgsquote bei 100 %. Das Design für die Zusatzinformation ist in Abbildung 4.7 dargestellt. Jedem Beurteilungswert werden konkrete Kriterien zugeordnet, die der Patient erfüllen muss. Aufgerufen wird die Zusatzinformation über die mittig positionierten Buttons mit den Zahlenwerten.

Abbildung 4.4: Erfassung einer ICD-Diagnose.

Abbildung 4.5: Verweis auf die Erfassung eines ICF-Krankheitsbildes.

Wir konnten vier Ursachen für die Nichtbeachtung der hinterlegten Informationen identifizieren:

- **Unklare Bedeutung des Beurteilungswertes:** Ein Teil der Testpersonen verstand die Bedeutung des angezeigten Beurteilungswertes nicht. In Abbildung 4.6 ist der Beurteilungswert für den Code ”d450 Gehen“ zwei Mal als eine grüne 1 dargestellt. Obwohl im Vorgespräch geklärt wurde, dass die Bewertung der ICF-Codes anhand einer fünfstufigen Skala erfolgt, konnte der Zusammenhang zwischen dem Zahlenwert und der ICF-Skala nicht hergestellt werden.

  Empfehlung: Dieses Problem kann unserer Meinung nach auf zwei Arten gelöst werden. Einerseits ist eine Einschulung zu ICF Voraussetzung für den Einsatz unserer Anwendung. Ist den Nutzern klar, wie eine ICF-Beurteilung zu erfolgen hat, sollte die Interpretation der oben angesprochenen Zahlenwerte kein Problem darstellen. Andererseits könnte man die Beurteilung eines ICF-Codes anhand eines Zahlenwertes vor dem Nutzer verstecken. Statt eines

Abbildung 4.6: ICF-Erfassungsdialog.

Abbildung 4.7: Information zu den Beurteilungswerten des Codes "d450 Gehen".

Zahlenwertes könnte man die Einführung einer Farbskala andenken, die allein auf Basis von Farben den Stand der ICF-Beurteilung darstellt. Aufgrund der nachfolgend beschriebenen Problematik der Farbgestaltung unserer Applikation, werden wir vorerst die Zahlenwerte im Design erhalten und von einer zusätzlichen Farbskala Abstand nehmen. Die Beobachtung dieser Problematik wird aber fester Bestandteil künftiger Usability-Tests sein.

- **Zusammenhang zwischen Beurteilungswert und Beurteilungs-Buttons:** Einige Testpersonen erkannten den Zusammenhang zwischen den Beurteilungs-Buttons und dem Zahlenwert nicht. Obwohl verstanden wurde, dass die rot-gelb-grünen Buttons eine Verschlechterung, eine konstante Entwicklung bzw. eine Verbesserung im Vergleich zur letzten ICF-Beurteilung symbolisierten, war nicht klar, dass das Betätigen dieser Buttons Auswirkungen auf den Zahlenwert hat. Dieses Problem bestand bereits in den Vortests, woraufhin wir den Zahlenwert in der Mitte unserer Beurteilungs-Buttons positionierten. Der Großteil der Testpersonen der Usability-Tests bevorzugte allerdings die

Positionierung des Zahlenwertes links der Beurteilungs-Buttons.

Empfehlung: Aufgrund dieser Ergebnisse werden wir den Zahlenwert wieder links der Beurteilungs-Buttons positionieren. Außerdem werden wir durch visuelle Hilfsmittel, wie beispielsweise ein kurzes Aufblinken des Zahlenwertes bei Betätigen eines Buttons, versuchen, den Zusammenhang zu betonen.

- **Doppelte Farbsymbolik:** Die dritte Problematik bei dieser Aufgabe war auf eine Schwachstelle unseres Farbkonzepts zurückzuführen. Einige Testpersonen verunsicherte die doppelte Farbgebung innerhalb der ICF-Erfassung. Einerseits wurden die Farben rot, gelb, grün für die Beurteilungs-Buttons zur Darstellung einer Verschlechterung, einer konstanten Entwicklung bzw. einer Verbesserung verwendet. Andererseits haben die Zahlenwerte der ICF-Beurteilung ebenfalls Farben zugeordnet. Diese stehen allerdings nicht für eine Entwicklung, sondern für eine statische Wertung. Die Werte 0-1 sind beispielsweise mit Grüntönen hinterlegt, da diese Werte für keine bzw. eine leichte Beeinträchtigung des Patienten stehen und somit eine positive Assoziation wecken sollen. Dahingegen sind die Werte 3-4 mit roten Farbtönen hinterlegt, um die Bedeutung einer starken bis kompletten Beeinträchtigung des Patienten zu verdeutlichen.

  Empfehlung: Um diese Problematik zu lösen, verzichten wir in Zukunft auf die Hinterlegung der einzelnen Beurteilungswerte, wodurch die Farben rot, gelb, grün nur noch die Veränderung der Beurteilung symbolisieren.

- **Auffindbarkeit der Information zu den Beurteilungswerten:** Die Testpersonen hatten teilweise Probleme die hinterlegte Information zu den Beurteilungswerten zu finden. Oftmals wurde bei der Suche die Information zum ICF-Code oder zur Maßnahmen- und Zieleplanung aufgerufen, in Abbildung 4.6 als grüne Zeile bzw. grauer Button dargestellt.

  Empfehlung: Um dieses Problem zu beseitigen, werden wir die Position der hinterlegten Information zwar nicht verändern, aber mit visuellen Mitteln versuchen, die versteckte Hintergrundinformation zu betonen.

Aufgabe 4. Diese Aufgabe befasste sich mit der Auswertung der ICF-Erfassungen. Besonders positiv hervorzuheben ist hier, dass die Lernkurve der Testnutzer ausgehend von Teilaufgabe 4.1 im Vergleich zu Teilaufgabe 4.2 stark ansteigt, wie in Abbildung 4.2 ersichtlich. Während Teilaufgabe 4.1 keine Testperson ohne Hilfestellung lösen konnte, führten bereits 87,5 % der Testpersonen Teilaufgabe 4.2 selbständig durch. Dies führen wir teilweise auf einen schnellen Lerneffekt durch unsere simple Menüführung zurück, teilweise aber auch auf das problematische Design der Teilaufgabe 4.1.

Teilaufgabe 4.1: Abbildung 4.8 zeigt die Veränderung einzelner ICF-Codes über einen bestimmten Zeitraum hinweg. In dieser Abbildung wird beispielsweise die aktuellste ICF-Bewertung eines Patienten mit der ICF-Bewertung vor 100 Erfassungen verglichen. Die Testpersonen waren nicht in der Lage die Tabelle korrekt zu interpretieren. Die folgenden Punkte fassen die aufgetretenen Probleme zusammen:

- **Allgemeine Informationsaufbereitung:** Den Testpersonen war nicht klar, dass zwischen, den in der Tabelle dargestellten Werten, 100 Erfassungen liegen. Es wurde intuitiv vermutet, dass die aktuellste Beurteilung mit der vorherigen Beurteilung verglichen wird. Die Informationen, die auf einen längeren Zeitraum des Beurteilungsvergleichs hinwiesen, wie beispielsweise die Anzeige ”100 Erfassungen“ im oberen Bereich des Bildschirms wurden nicht beachtet.

  Empfehlung: Hier muss visuell deutlicher hervorgehoben werden, über welchen Zeitraum sich der Trend der Verbesserung bzw. Verschlechterung oder der konstanten Entwicklung abspielt. Wenn möglich sollte die Informations- und Datenmenge in diesem Design reduziert werden, um die Interpretation der dargestellten Daten einfacher zu machen.

- **Doppelte Farbsymbolik:** Abermals wurden in diesem Design, wie bereits im Design der ICF-Erfassung, die Farben rot, gelb und grün mit einer doppelten Bedeutung belegt.

  Empfehlung: Auch hier werden wir in Zukunft auf eine farbliche Markierung der ICF-Beurteilungswerte verzichten und die Farben ausschließlich den Be-

deutungen ”Verbesserung“, ”Verschlechterung“ oder ”konstante Entwicklung“ zuordnen.

Abbildung 4.8: Liste ICF-Trendübersicht.

Abbildung 4.9: Menü der ICF-Auswertung

Ein weiteres Problem bei dieser Aufgabenstellung stellte das Menü dar, das Abbildung 4.9 zeigt. Das Menü der ICF-Auswertungen stellt Optionen zur Verfügung, um zwischen verschiedenen Auswertungsansichten zu wechseln und um die Einstellungen der Ansichten zu konfigurieren. Nachdem die Einstellungen für alle Ansichten der ICF-Auswertung übernommen werden, kann man mit dem ”Zurück“-Button zur letzten Ansicht zurückkehren. Die Einstellungsänderungen werden automatisch übernommen. Aus dem Desktop-Bereich waren aber die meisten Nutzer daran gewöhnt, ihre getroffenen Einstellungen zu bestätigen. Die wenigsten nutzten den ”Zurück“-Button ohne zuvor einen Bestätigungs-Button zu suchen.

Empfehlung: Hier stellt die einfachste Lösung aus unserer Sicht die Einführung eines ”Bestätigen“-Buttons dar. Bereits in den Vortests haben wir die Problematik des Bestätigens am Smartphone im Rahmen der Aufgabe 2 bemerkt. Durch das Hinzufügen eines Bestätigen-Buttons in diesem Bereich konnten die Ergebnisse verbessert werden.

Teilaufgabe 4.2: Wie bereits erwähnt, war der Lerneffekt von der 1. zur 2. Teilaufgabe enorm. Die Menüführung stellte für die wenigsten Testpersonen noch ein Problem dar. Auch die Anwendung des ”Zurück“-Buttons statt des Bestätigen-Buttons war einem Großteil der Testpersonen mittlerweile klar. Abbildung 4.10 zeigt die Detailansicht zum ICF-Code ”d450 Gehen“, der mit entsprechenden Einstellungen aufzurufen war.

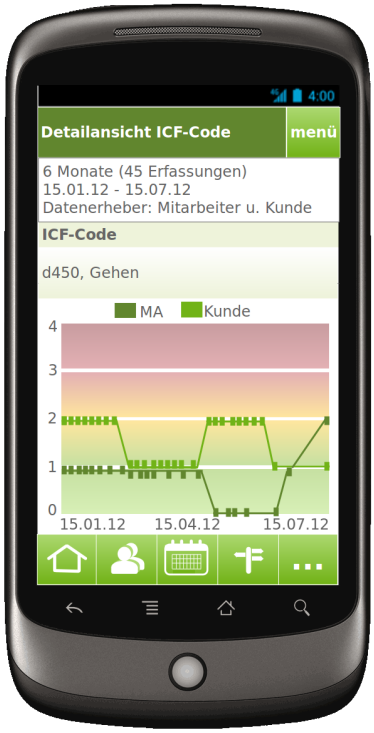

Abbildung 4.10: Detailansicht ICF-Code ”d450 Gehen“.

Weitere Anmerkungen zu den Designs. Abschließend sollen noch zwei generelle Anmerkungen, unabhängig von einer konkreten Testaufgabe, gemacht werden.

Navigationsleiste: Ein allgemeines Problem stellte die exponierte Position der grünen Navigationsleiste im unteren Bildschirmbereich dar. Vor Durchführung der Tests wurde darauf hingewiesen, dass die Navigationsleiste Bestandteil des Designkonzepts ist, für unsere Usability-Tests aber keine Rolle spielt und daher nicht benutzt werden soll. Trotz dieses Hinweises waren alle Testteilnehmer zu unterschiedlichen Zeitpunkten dazu verleitet, die Navigationsleiste anzusteuern.

Empfehlung: Eventuell ist für die Navigationsleiste eine weniger exponierte Po-

sition sinnvoller, da ein Wechseln zwischen grundlegenden Funktionalitäten der Anwendung selten vorkommen sollte. Durch eine weniger offensichtliche Position der Leiste wäre die Versuchung geringer, diese oft zu nutzen. Darüber hinaus könnte somit wertvoller Platz am Display gewonnen werden.

Farbkarte: Abbildungen 4.11 und 4.12 zeigen eine Übersicht über alle erfassten ICF-Beurteilungen eines Patienten. Auf der horizontalen Achse verlaufen die ICF-Codes, während die vertikale Achse die Zeit darstellt. Die Farbpunkte deuten die Veränderung des Gesundheitszustandes an (Verbesserung, Verschlechterung, gleichbleibende Entwicklung). Dieses Design wurde aufgrund der begrenzten Interaktionsmöglichkeit mit Papier-Prototypen im Vergleich zu tatsächlichen Smartphone-Anwendungen nicht im Rahmen des Usability-Tests behandelt. Allerdings war es Gegenstand der anschließenden Interviews und Diskussionen mit den Teilnehmern. Der Großteil der Testpersonen hielt die Notwendigkeit einer derartigen Übersicht am Smartphone für gering. Der Großteil vermutete, dass ein Langzeitverlauf eher am Desktop-PC ausgewertet wird, während im Berufsalltag die aktuellen Daten eine Rolle spielen. Einige Testteilnehmer schätzten die Notwendigkeit hingegen als hoch ein, um vor allem dem Patienten auch über lange Zeit seine Fortschritte auf einen Blick zeigen zu können. Unser Lösungsansatz für eine derartige Informationsdarstellung wurde im Detaillierungsgrad von Abbildung 4.11 größtenteils als komplex und unübersichtlich angesehen. Viele Testpersonen merkten an, dass vor allem Anhaltspunkte an den Achsen fehlten, um ein effektives Navigieren in der Darstellung zu ermöglichen. Die Farbcodierung allein war für die meisten nicht Anhaltspunkt genug. Der Detaillierungsgrad in Abbildung 4.12 wurde als weitaus angenehmer empfunden.

Empfehlung: Die Darstellung einer großen Anzahl an ICF-Beurteilungen muss weiter verbessert werden. Allerdings stufen wir dies zum jetzigen Zeitpunkt als nachrangig ein, da der Großteil der Testpersonen den Bedarf an einer derartigen Darstellung als gering einstuft.

Basierend auf den Ergebnissen der Tests und der Diskussion mit den Testpersonen beantworten wir in den nächsten Abschnitten die, zu Beginn definierten, Forschungsfragen.

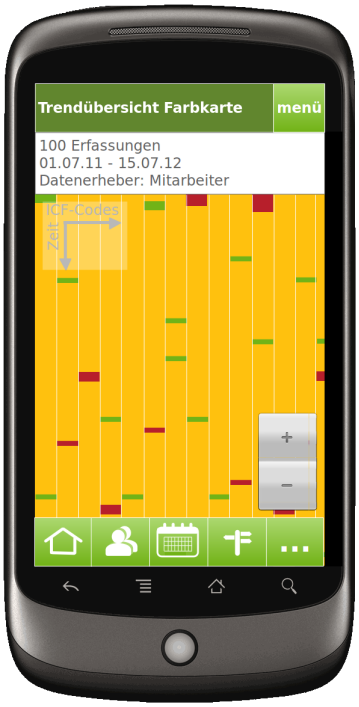

Abbildung 4.11: Farbkarte Übersicht.

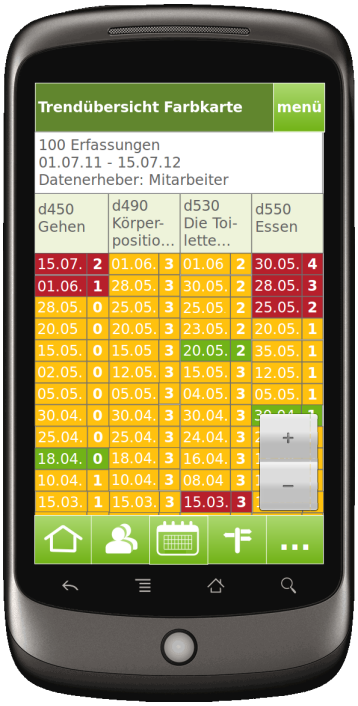

Abbildung 4.12: Farbkarte Details.

Beurteilung des Nutzens der Anwendung. 100 % der Testteilnehmer beurteilen die Praxistauglichkeit und den Nutzen der Applikation nach der Durchführung der Usability-Tests als hoch. Diese Beurteilung gründet auf mehreren Faktoren:

- Das Design wurde überwiegend als sehr übersichtlich und klar strukturiert empfunden.
- Mobile Technologien werden als sehr praktisch für den Berufsalltag eingeschätzt. Die Testpersonen erwarten durch den Einsatz mobiler Anwendungen eine große Zeitersparnis und ein effizienteres Arbeiten.
- Der Einsatz von ICF zur Gesundheitsdokumentation wird positiv beurteilt. Die Testpersonen schätzen dieses Klassifikationssystem als sehr umfangreich ein. Einige Teilnehmer erkannten Parallelen - sowohl in den Begrifflichkeiten als auch in der Art der Beurteilung - zu ihnen bekannten Skalen und Fragebögen. Dies erleichtert ihrer Meinung nach den Umstieg auf ICF.

Verarbeitung aller relevanter Informationen. Bezüglich der reinen Gesundheitsdokumentation vermissten die Testpersonen keine Informationen. Aber sowohl

Teilnehmer aus der Berufsgruppe der Physiotherapeuten als auch der Krankenpfleger betonten, dass ein angeschlossener Ursachen-, Maßnahmen- und Zielekatalog für die Praxis relevant sei. Eine derartige Funktionalität ist für die Anwendung geplant, jedoch nicht Gegenstand dieser Arbeit. Ein weiterer Punkt, der von einigen Teilnehmern angesprochen wurde, war die Möglichkeit frei formulierte Notizen hinterlegen zu können. Dieser starke Wunsch resultiert vermutlich aus der derzeitigen Praxis den Großteil der Gesundheitsdokumentation als Freitext zu notieren. Für die reine Erfassung von ICF-Beurteilungen werden wir keine Möglichkeiten für Freitexte vorsehen, für den später angeschlossenen Ursachen-, Maßnahmen- und Zielekatalog werden diese aber eingeplant. Als eine weitere, wichtige Information sahen viele Testteilnehmer die übersichtliche Darstellung aller aktueller ICF-Werte eines Patienten an. Ein derartiges Design wurde im Rahmen des Usability-Tests nicht verwendet, wurde aber bereits geplant, wie Abbildung 4.13 beispielhaft zeigt. Eine mögliche Einschränkung befürchteten einige Teilnehmer im Zusammenhang mit den ICF-Codesets. Die Befürchtung war, dass jede Berufsgruppe, also vor allem Pflegepersonal und Physiotherapeuten, einen anderen Zugang zu der Gesundheitsdokumentation eines Patienten hat. Beispielsweise benötigt ein Pfleger eventuell nur die Information darüber, ob ein Patient im Stande ist selbständig 10 Meter zu gehen, während ein Physiotherapeut diese Information weiter unterteilt in Trittsicherheit, Gleichgewichtsgefühl, etc. Derartige berufsspezifische Anforderungen im Rahmen der ICF-Codesets zu berücksichtigen, ist sicher ein ausschlaggebendes Kriterium für den Erfolg dieser Anwendung bzw. den Einsatz von ICF generell. Allerdings fällt die Zusammensetzung von *ICF-CoreSets* nicht in den Rahmen dieser Arbeit.

Allgemeine Benutzbarkeit und Navigation. Wie bereits im obigen Abschnitt erwähnt, wurde das Design der Anwendung von allen Testteilnehmern durchgehend als sehr ansprechend und übersichtlich beurteilt. Die Navigation war für den Großteil der Teilnehmer einfach und intuitiv gestaltet. Einziger größerer Kritikpunkt war die doppelte Farbsymbolik. Die restlichen Schwierigkeiten wurden auf die mangelnden ICF- bzw. Smartphone-Kenntnisse zurückgeführt.

Beurteilung des benötigten Vorwissens. Trotz des intuitiven Aufbaus der mobilen Anwendung halten alle Testteilnehmer eine Einschulung für notwendig. Die

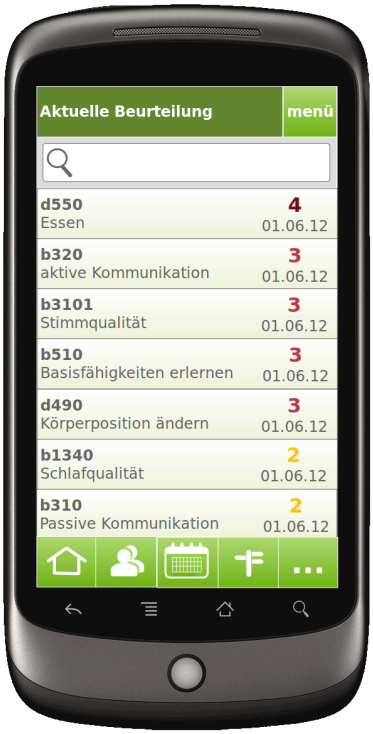

Abbildung 4.13: Aktuelle ICF-Beurteilung.

Einschulung muss einerseits die Grundlagen zu ICF und andererseits den Umgang mit der mobilen Anwendung selbst umfassen. Allerdings betonen die meisten Teilnehmer, dass bereits eine kurze Einschulung mit den wichtigsten Hintergrundinformationen ausreichen würde, um sich in der Anwendung zurecht zu finden.

Zusammenfassend kann also festgehalten werden, dass die Testteilnehmer der geplanten Anwendung unter der Voraussetzung einer kurzen Einschulung eine hohe Praxistauglichkeit zusprechen. Das Design wurde bereits zu diesem frühen Entwicklungszeitpunkt als ansprechend, durchdacht und intuitiv beurteilt. Die größten Kritikpunkte waren die doppelte Farbsymbolik und das Fehlen eines Bestätigen-Buttons, wie er aus Desktop-Anwendungen bekannt ist.

5. Experimentelle Implementierung der ICF-Anwendung

Basierend auf den Erkenntnissen der Usability-Tests aus Abschnitt 4.2.4 wurde eine prototypische Anwendung für die Android-Plattform entwickelt. Einen Einblick in den grundlegenden Funktionsumfang der implementierten Anwendung geben wir zu Beginn in Abschnitt 5.1. Danach gehen wir in Abschnitt 5.2 detaillierter auf die Software-Architektur und typische Anwendungsfälle ein. Grundsätzlich ist die Applikation als sogenannte "Occassionally Connected"-Anwendung konzipiert. Derartige Anwendungen können auch ohne Netzanbindung eingesetzt werden, was beispielsweise bei der Arbeit im mobilen Pflegedienst notwendig ist. Dieser Umstand führt allerdings zu einer erhöhten Komplexität, denn sowohl Teile der Geschäftslogik als auch des zentralen Datenbestandes des Backends müssen auf die mobile Anwendung bzw. das Gerät übertragen werden. Die damit zusammenhängende Synchronisationsproblematik klären wir in Abschnitt 5.3.

5.1 Funktionsumfang

In diesem Abschnitt zeigen wir den Funktionsumfang des implementierten Prototypen anhand mehrerer UML-Anwendungsfalldiagramme (Use-Case-Diagramme).

5.1.1 Authentifizierung und Synchronisation

Abbildung 5.1 zeigt die Anwendungsfälle zur Benutzer-Authentifizierung und Synchronisation zwischen lokaler Anwendung und Backend-Server. Beim ersten Login eines Benutzers muss die Anwendung initialisiert werden. Dazu muss sich der Benutzer am zentralen Server authentifizieren. Die Authentifizierung erfolgt mit Benutzer-

name und Passwort. Ist sie erfolgreich, werden, die dem Nutzer zugeordneten Daten, in die lokale Datenbank geladen. Der Benutzer kann die Synchronisation neuer oder aktualisierter Daten jederzeit manuell auslösen. Weiters kann er konfigurieren, ob die Datensynchronisation in regelmäßigen Zeitabständen automatisch erfolgen soll. Dazu muss er entsprechende Einstellungen bezüglich der Synchronisationsrichtlinie treffen. Der Benutzer kann sich explizit aus der mobilen Anwendung ausloggen oder wird aus Sicherheitsgründen nach einer gewissen Zeit der Inaktivität automatisch ausgeloggt.

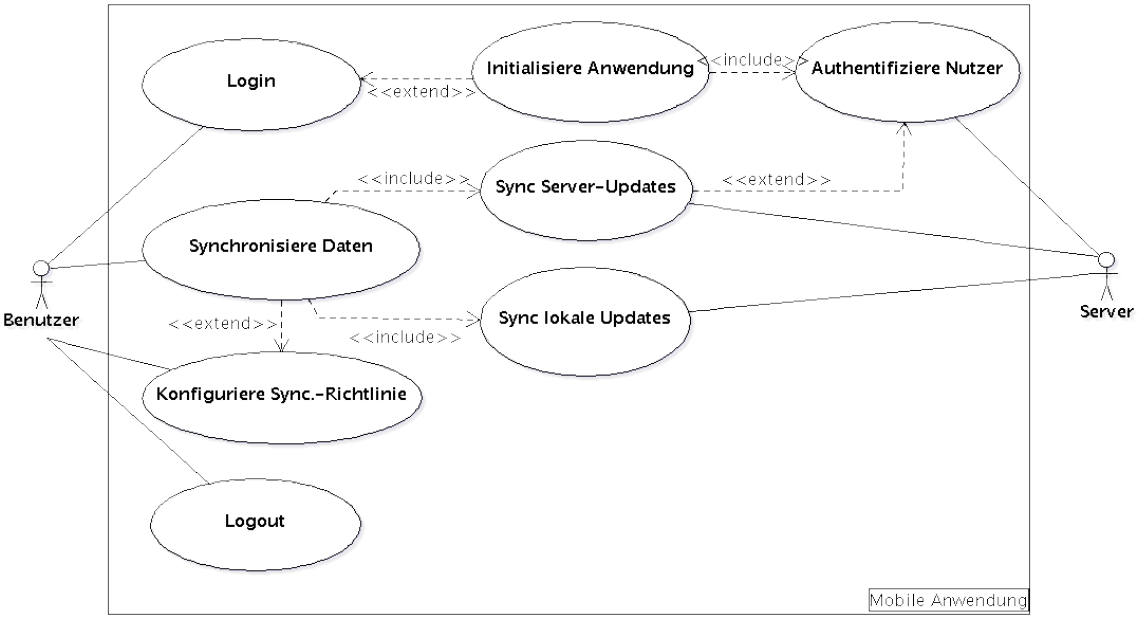

Abbildung 5.1: Use-Case-Diagramm zur Authentifizierung und Synchronisation.

5.1.2 ICD- und ICF-Datenmanipulation

In Abbildung 5.2 zeigen wir die Möglichkeiten zur Datenerfassung und -manipulation in Bezug auf ICD-Diagnosen, ICF-Diagnosegruppen und ICF-Codes. Ausgehend vom Informationsblatt eines Patienten können für diesen ICD-Diagnosen und ICF-Diagnosegruppen angezeigt, neu erfasst oder bearbeitet werden. Die ICF-Diagnosegruppen enthalten ICF-Codes für eine Gruppe zusammengehöriger Diagnosen. Wird einem Patienten eine ICF-Diagnosegruppe zugeordnet, werden die entsprechenden ICF-Codes zur Dokumentation seines Gesundheitszustandes angezeigt. Wird eine neue Diagnose erfasst, die bisher keiner ICF-Diagnosegruppe des Patienten zuge-

ordnet ist, erhält der Benutzer einen Hinweis zur Erfassung der entsprechenden Gruppe. Ebenso verhält es sich bei der Erfassung einer neuen ICF-Diagnosegruppe ohne entsprechende ICD-Diagnose. Bei der Erfassung von ICF-Codes können zusätzliche Informationen zum Code oder den Beurteilungswerten angezeigt werden. Weiters kann die letzte Erfassung rückgängig gemacht werden.

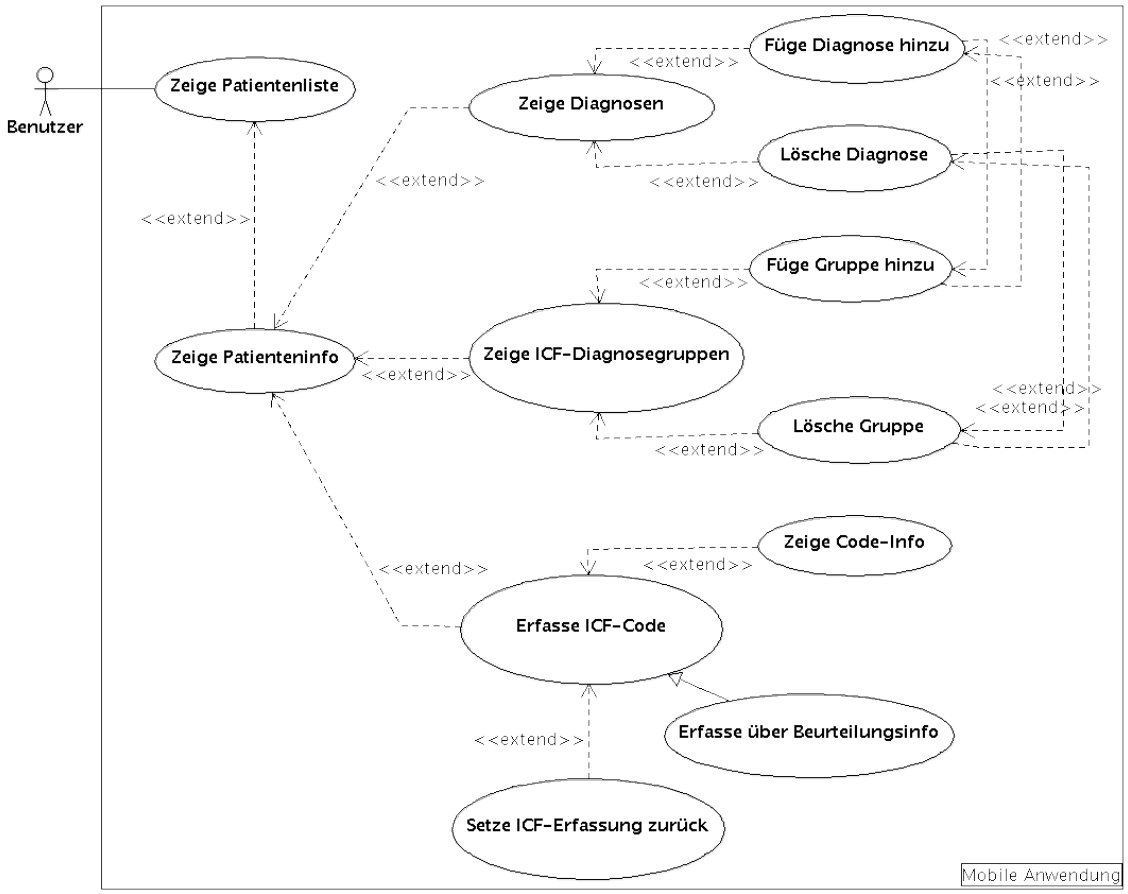

Abbildung 5.2: Use-Case-Diagramm zur ICD- und ICF-Datenmanipulation.

5.1.3 ICF-Auswertung

Das Use-Case-Diagramm zur ICF-Auswertung in Abbildung 5.3 zeigt, wie die laufende ICF-Dokumentation eines Patienten analysiert werden kann. Dem Benutzer steht eine Trendliste zur Verfügung, die ICF-Codes nach ihrer aktuellen Entwicklung sortiert. Somit können schnell Verbesserungen und Verschlechterungen des Gesundheitszustandes festgestellt werden. Weiters hat der Benutzer die Möglichkeit nur die aktuellste Erfassung aller ICF-Codes eines Patienten abzufragen. Außerdem können Details zu allen ICF-Codes als Linien- bzw. Balkendiagramm dargestellt werden.

Über die Optionen kann der Benutzer Einstellungen zur Ansicht der Auswertungen treffen.

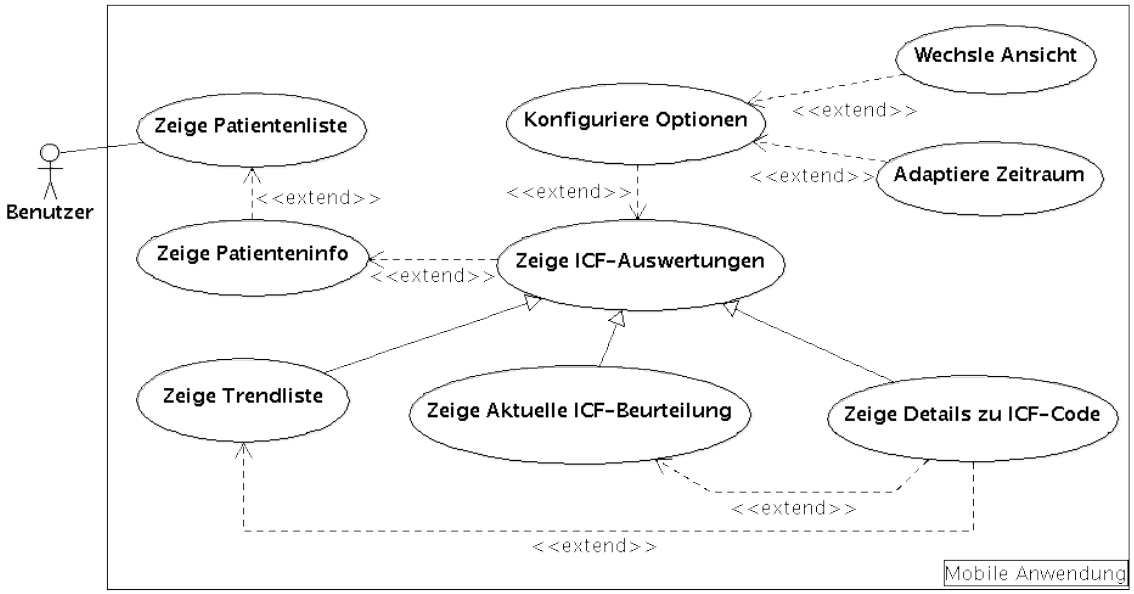

Abbildung 5.3: Use-Case-Diagramm zur ICF-Auswertung.

5.2 Software-Architektur

Bevor wir in Abschnitt 5.2.2 auf den Aufbau und die Komponenten unserer Software-Architektur eingehen, argumentieren wir unsere Entscheidung für eine REST-basierte Backend-Anbindung in Abschnitt 5.2.1.

5.2.1 REST-basierte Backend-Anbindung

Nachdem wir in Kapitel 2.2 SOAP- und REST-basierte Webservices ausführlich analysiert haben, argumentieren wir hier unsere Entscheidung zugunsten der REST-basierten Architektur für die Implementierung unserer Backend-Anbindung. Dass die Entscheidung zwischen REST und SOAP keine einfache ist, zeigt beispielsweise schon die Arbeit von Pautasso u. a. [33]. Sie entwickelten ein komplexes Modell zum Vergleich aller relevanten Aspekte von SOAP und REST, das den Entscheidungsprozess für oder gegen eine Architektur strukturieren soll. Unabhängig von derart wissenschaftlichen Systemen zur Entscheidungsfindung genügt eine kurze Literaturrecherche, um zu erkennen, dass die Welt der Webservices in zwei Lager geteilt ist.

Die einen halten die REST-Architektur nur für kleine Anwendungen, die über simple CRUD-Operationen nicht hinausgehen, geeignet und warnen vor einem Einsatz in Geschäftsanwendungen. Die anderen kritisieren die unnötige Komplexität von SOAP, die gegen die vorgegebene Architektur des Webs arbeitet, statt die Vorteile dieser zu nutzen. Das in diesem Zusammenhang oft genannte Argument der Unabhängigkeit vom Transportprotokoll, wird durch die Praxis aufgehoben, die zeigt, dass in einer überwältigend großen Zahl der Fälle SOAP auf HTTP aufsetzt. Dennoch berücksichtigen SOAP-basierte Webservices die Architektur des Webs nicht. Denn sie stellen kein homogenes Interface bereit, agieren nicht zustandslos und machen somit einen effektiven Einsatz von Caching schwierig [37].

Der Trend der letzten Jahre geht eindeutig in Richtung REST. Beispielsweise benennt Coulouris [13] bereits im Jahr 2003 den Anteil der REST-basierten Anfragen an den Webservice von *amazon.com* mit 80%, während nur 20% SOAP nutzen. Diese Entwicklung ist sicher nicht zuletzt durch die einfachere Integration und schnellere Entwicklung von REST-basierten Anwendungen bedingt. In letzter Zeit wurden außerdem immer mehr Stimmen laut, die sich für REST im Umfeld von Geschäftsanwendungen aussprechen. Hier sei beispielsweise auf Tilkov [83] verwiesen.

Was SOAP sicher zu Gute gehalten werden muss, ist die umfangreiche Spezifikation und Unterstützung komplexer Szenarien bezogen auf Sicherheit und Transaktionen. Dem entgegengehalten werden kann nur, dass Szenarien, die den unbedingten Einsatz dieser Spezifikationen erfordern, sehr selten sind und praxisrelevantere Szenarien, wie in Abschnitt 2.2.4 gezeigt wurde, auch mit einer REST-basierten Architektur handhabbar sind.

Sollte man sich beispielsweise aufgrund der umfangreichen Sicherheitsspezifikationen für den Einsatz von SOAP entscheiden, so ist zuvor zu prüfen, ob eine Unterstützung dieser Spezifikationen auf der gewählten Plattform überhaupt verfügbar ist. Denn ohne entsprechende Tool- und Bibliotheksunterstützung wird die Entwicklung von SOAP-Applikationen sehr umfangreich und komplex. Wir mussten in diesem Zusammenhang feststellen, dass der SOAP-Support für die Android-Plattform derzeit noch gering ist. Nativ wird SOAP nicht unterstützt. Open-Source-Bibliotheken, wie *kSOAP2* [67] für Java ME unterstützen den Standard in zu geringem Umfang für einen professionellen Einsatz [67, 68]. Eine Alternative wäre beispielsweise die C/C++-basierte Bibliothek *gSOAP* [57]. Sie unterstützt neben der Basisspezifikati-

on für SOAP WS-Policy, WS-Addressing, WS-ReliableMessaging und WS-Security [6]. Für die kommerzielle Nutzung dieses Produkts fallen allerdings Lizenzgebühren an. Nachdem der mobile Markt derzeit noch stark auf den Endverbraucher konzentriert ist, wird mit einer nativen Unterstützung von SOAP laut Bertram und Kleiner [6] in nächster Zukunft nicht zu rechnen sein. Die Unterstützung von REST-Architekturen auf Android stellt im Unterschied zu SOAP kein Problem dar. Zahlreiche Bibliotheken für REST-basierte Android-Clients stehen zur Verfügung. Als Beispiele seien Restlet [77], Resting [76] und der Spring RESTClient [78] genannt.

Aus den bisher getroffenen Feststellungen lässt sich unsere Tendenz zur REST-basierten Architektur bereits ablesen. Dennoch muss festgehalten werden, dass unsere Applikation sensible Gesundheitsdaten verarbeitet, was die Einhaltung strenger Datenschutzvorschriften impliziert. Andry u. a. [2] waren in ihrer Arbeit mit ebenso sensiblen Patientendaten konfrontiert und beschreiben ihre Umsetzung einer REST-Architektur zur Interaktion mit diesen Daten auf einem mobilen Endgerät. Die durchwegs positiven Ergebnisse, die Andry u. a. durch den REST-basierten Ansatz erzielen konnten – besonders hervorgehoben wurden die einfache Wartung und Erweiterbarkeit der Applikation – sowie die einfache Realisierung von Sicherheitsmechanismen über REST motivierten uns zusätzlich für unsere Applikation ebenfalls diese Art der Architektur zu wählen.

5.2.2 Software-Komponenten im Schichtenmodell

Abbildung 5.4 zeigt die Software-Komponenten der mobilen Anwendung und des Backend-Servers in einem Schichtenmodell. Die mobile Anwendung wurde, wie bereits erwähnt, für die Android-Plattform entwickelt. Das Backend wurde ebenfalls in Java programmiert und läuft auf einem Apache-Tomcat-Applikationsserver. Für die REST-basierte Kommunikation zwischen mobiler Anwendung und Backend wird die Restlet API [77] in Verbindung mit der Jackson Library [66] zur automatischen Generierung von JSON-Strings aus Java Beans verwendet. Die Komponenten der mobilen Anwendung wurden in Abbildung 5.4 drei Ebenen zugeordnet, die in den folgenden Abschnitten detailliert betrachtet werden.

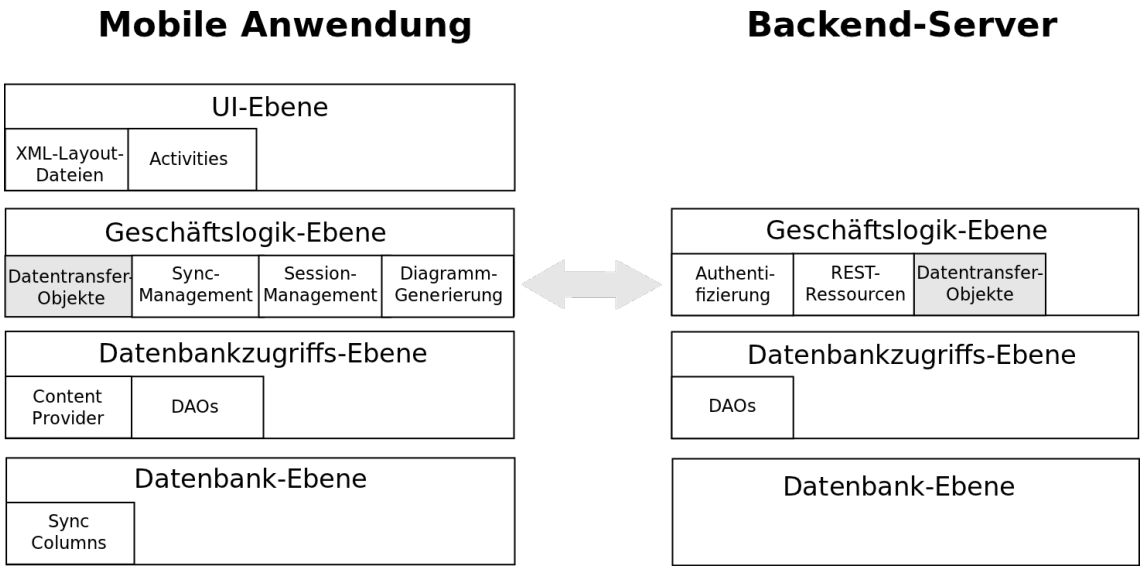

Abbildung 5.4: Software-Architektur.

UI-Ebene. Die Komponenten dieser Ebene dienen der Interaktion mit dem Benutzer. Zentrales Element ist die *Activity*. Sie stellt dem Anwender eine Benutzeroberfläche zur Verfügung. Der Aufbau dieser wird durch *XML-Layout-Dateien* definiert. Da Android-Endgeräte mit unterschiedlichen Bildschirmgrößen und -auflösungen existieren, sind darauf abgestimmte Layouts zu entwickeln. Für den Prototypen konzentrierten wir uns auf große und hoch auflösende Smartphone-Bildschirme. Die Entwicklung eines Layouts für Tablet-Bildschirme ist geplant. Die Ergebnisse der Usability-Tests aus Abschnitt 4.2.4 waren maßgebend für das Design der Activities.

Geschäftslogik-Ebene. Die Ebene der Geschäftslogik beinhaltet sowohl client- als auch serverseitig *Datentransfer-Objekte*. Dabei handelt es sich um simple Java Beans, die dem Datenaustausch mit dem Backend-Server dienen. Sie kapseln den Zustand einer Ressource, wie beispielsweise die ICF-Erfassung eines Patienten, und werden in JSON-Strings serialisiert und übers Netz versendet. Auf der Gegenseite werden sie entsprechend deserialisiert. Zur De-/Serialisierung der *Datentransfer-Objekte* wird die Jackson Library [66] eingesetzt. Den Austausch dieser Komponenten und somit die Synchronisation der Datenbestände übernimmt clientseitig die Komponente *Sync-Management*. Hier können Daten registriert werden, deren Synchronisation durchgeführt werden soll. Dabei werden sowohl zentrale Aktualisierungen des Backend-Servers als auch lokal veränderte Daten abgeglichen. Bei entsprechender Konfiguration der Synchronisationsrichtlinie wird der Aufruf des Syn-

chronisationsmechanismus regelmäßig vom Betriebssystem initiiert. Der Vorteil hier ist, dass das Betriebssystem den Datenabgleich auf einen Zeitpunkt legen kann, zu dem die Ressourcen des Endgerätes anderweitig nicht benötigt werden. Soll das Betriebssystem nicht über die Datensynchronisation entscheiden, kann die Synchronisationsrichtlinie so konfiguriert werden, dass der Synchronisationsmechanismus nur durch den Benutzer angestoßen werden kann. Für die Überwachung und die Handhabung von Änderungen des Netzwerkstatus oder der Synchronisationsrichtlinie, die Auswirkungen auf das Synchronisationsverhalten haben, ist ein eigener Listener zuständig. Dem clientseitigen Handling der Synchronisation steht serverseitig die Komponente der *REST-Ressourcen* gegenüber. Diese werden entsprechend der REST-basierten Architektur über spezielle ressourcenorientierte URIs über HTTPS adressiert. Die serverseitigen *REST-Ressourcen* nehmen *Datentransfer-Objekte* entgegen oder liefern diese auf eine Anfrage zurück. Die dazu notwendigen Datenbank-Zugriffe werden, eine Ebene tiefer im Schichtenmodell, über die serverseitigen Datenzugriffsobjekte (Data Access Objects, *DAOs*) ausgeführt. Eine weitere Komponente der Geschäftslogik-Ebene auf der Serverseite ist die Komponente zur *Authentifizierung*. Diese ist den *REST-Ressourcen* vorgeschalten und verhindert einen Zugriff durch nicht berechtigte Benutzer. Die Authentifizierung erfolgt über Benutzername und Passwort bzw. sogenannten AuthTokens mittels HTTP Basic Authentication. Auch der clientseitige Zugriff auf Informationen innerhalb der mobilen Anwendung ist durch Authentifizierungsmechanismen geschützt. Diese gehören zur *Session-Management*-Komponente des Clients. Diese Komponente schützt die Anwendung vor unberechtigten Zugriffen durch ein simples Login-Interface. Bei der ersten Anmeldung eines Benutzers werden Benutzername und Passwort sofort ans Backend weitergeleitet, um die Richtigkeit zu überprüfen. Danach wird die Anwendung initialisiert, indem, die dem Nutzer zugeordneten Daten, vom Backend in die lokale Datenbank geschrieben werden. Um auch nach der erstmaligen Authentifizierung des Benutzers die Datensicherheit zu gewährleisten, hat der Nutzer die Möglichkeit sich auszuloggen. Tut er dies nicht, wird er von der Anwendung nach einer bestimmten Dauer der Inaktivität automatisch ausgeloggt. Die letzte Komponenete dieser Ebene ist die *Diagramm-Generierung* zur graphischen Aufbereitung der ICF-Dokumentation. Die Open-Source-Bibliothek *achartengine* [40] wird von dieser Komponente verwendet, um Linien- und Balkendiagramme zu generieren.

Datenbankzugriffs-Ebene. Die Komponenten dieser Ebene abstrahieren und kapseln den Zugriff auf die darunterliegende Datenbank. Der *ContentProvider* ist der Mechanismus der Android-Plattform, um den Datenbank-Zugriff grundlegend zu kapseln bzw. um anderen Applikationen Daten zur Verfügung zu stellen. Die Data Access Objects (*DAOs*) implementieren die Datenbank-Zugriffslogik für *Datentransfer-Objekte*. Somit können über die DAOs *Datentransfer-Objekte* in der Datenbank abgelegt bzw. aus dieser generiert werden. Eine spezielle Klasse an *DAOs* stellt die Funktioanlität bereit, zu synchronisierende Daten in der Datenbank aufzufinden und entsprechende *Datentransfer-Objekte* zu generieren, die diese Daten kapseln. Dazu werden spezielle Synchronisations-Flags in der Datenbank benötigt. Dieses Thema wird in Abschnitt 5.3 ausführlich behandelt.

Datenbank-Ebene. Bei der lokalen Datenbank am mobilen Gerät handelt es sich um eine SQLite-Datenbank. Ihr gegenüber steht eine MySQL-Datenbank auf der Serverseite. Jede zu synchronisierende Tabelle der lokalen Datenbank wird um spezielle Synchronisations-Flags, die *SyncColumns* erweitert. Sie spiegeln den Synchronisationszustand des Tabelleneintrags wider, was in Abschnitt 5.3 näher erläutert wird.

5.3 Synchronisationsproblematik

Wie bereits erwähnt, wurde die ICF-Anwendung unter der Prämisse entwickelt, die Kernfunktionalität unabhängig von der Netzverfügbarkeit nutzbar zu machen. Die Entwicklung solcher gelegentlich verbundener Anwendungen ist komplexer als die Entwicklung reiner Online-Anwendungen, die nur bei bestehender Netzverbindung einsetzbar sind. Soll die Anwendung auch ohne Netzanbindung einsetzbar sein, müssen Daten lokal gespeichert und zu einem späteren Zeitpunkt mit der zentralen Datenbank synchronisiert werden. Trotz der erhöhten Komplexität kann eine gelegentlich verbundene Anwendung in bestimmten Szenarien von großem Vorteil sein. Man denke hier beispielsweise an einen Mitarbeiter der mobilen Pflege bei schlechter Netzabdeckung am Land. Doch nicht nur derartige Szenarien sprechen für die lokale Speicherung von Daten. Auch im Hinblick auf die begrenzten Ressourcen mobiler Endgeräte ist die lokale Speicherung von Daten - selbst bei reinen Online-Anwendungen - sinnvoll. Dem Vortrag von Virgil Dobjanschi im Rahmen der

Google I/O 2010 [52] folgend, sprechen auf der Android-Plattform folgende Gründe für die lokale Speicherung von Daten:

- **Reduzierung des Datenverkehrs:** Wird das Ergebnis einer REST-Anfrage lokal gespeichert, hat man die Möglichkeit jeweils nur aktuellere Daten vom Server abzufragen. Dadurch können der Datenverkehr des Geräts reduziert sowie Performance und Akkulaufzeit erheblich gesteigert werden.

- **Vermeidung von Datenverlust:** Typischerweise laufen länger andauernde Operationen, wie REST-Anfragen, in Services im Hintergrund ab. Somit blockiert die Benutzeroberfläche nicht während die REST-Anfrage ausgeführt wird. Eine zuständige Activity wird durch den Service über das Einlangen des REST-Ergebnisses informiert, um dieses weiterverarbeiten zu können. Allerdings kann nur eine Activity informiert werden, die gerade aktiv ist. Wird beispielsweise für die Anzeige in der Patienten-Activity eine Patientenliste angefordert und erreicht den Nutzer ein Anruf bevor die Antwort vom Server einlangt, pausiert die Patienten-Activity. Die Activity zur Beantwortung des Anrufs ist nun aktiv. Wenn die Patientenliste vom Backend-Server die mobile Anwendung erreicht, verständigt der Hintergrundprozess die Patienten-Activity darüber. Doch diese kann auf die Verständigung nicht reagieren, da sie noch immer pausiert. Das Ergebnis der REST-Anfrage geht somit verloren. Navigiert der Nutzer nach seinem Telefonat zurück zur Ansicht der Patientenliste, muss die REST-Anfrage erneut abgesetzt werden. Ein derartiges Design bringt erhöhten Netzwerkverkehr und unnötige CPU-Auslastung mit sich. Wird die Patientenliste dagegen lokal abgespeichert, kann die Patienten-Activity die Daten im Nachhinein in der lokalen Datenbank abfragen, statt die REST-Anfrage erneut abzusetzen.

Die obigen Punkte zeigen also, dass ein Verzicht der lokalen Datenspeicherung auf Kosten der Performance, des Netzwerkverkehrs und der Akkulaufzeit geht.

Datenpartitionierung. Die ICF-Anwendung verfügt somit aus den oben genannten Gründen über eine lokale Datenbank, die mit dem zentralen Datenbestand am Server synchronisiert werden muss. An dieser Stelle muss festgehalten werden, dass

die Synchronisation verteilter Datenbanksysteme ein komplexes Themengebiet darstellt. Zahlreiche Verfahren und Tools wurden in diesem Zusammenhang entwickelt. Typisch für mobile Anwendungen, die offline verfügbar sind, ist die sogenannte Mergereplikation. Bei dieser Form der Datenbankreplikation dürfen die Daten an mehreren Replikaten verändert werden. Datenänderungen in diesen Replikaten werden typischerweise verzögert synchronisiert [72]. Hier kann es zu Inkonsistenzen und Konflikten kommen, sollten in mindestens zwei der Replikate dieselben Daten verändert worden sein [72]. Der Server muss entsprechende Konfliktlösungsverfahren implementieren, um die Daten wieder in einen konsistenten Zustand zu versetzen [72]. Zahlreiche Tools am Markt bieten hier vorgefertigte Lösungen an. Als Beispiel kann der Oracle Database Mobile Server 11g [75] genannt werden. Allerdings sind diese Tools kostenpflichtig und für die Implementierung einer prototypischen Anwendung nicht angemessen. Aus diesem Grund wurde für den Rahmen dieser Arbeit die Prämisse getroffen, dass zwei mobile Anwendungen zu keinem Zeitpunkt auf dieselben Daten schreibend zugreifen. Konflikte bei der Synchronisation werden also durch organisatorische Maßnahmen verhindert. Derartige Prämissen sind ein gängiges Verfahren, um Konflikte zu vermeiden. Man spricht in diesem Zusammenhang von einer Partitionierung der Daten. In unserem Fall würde dies beispielsweise bedeuten, dass die Diagnose eines Patienten nur von seinem zuständigen Arzt geändert werden darf. Die prototypische Anwendung wurde hinsichtlich der Datenpartitionierung für die Berufsgruppe der Krankenpfleger konfiguriert. Diese haben lesenden Zugriff auf die Diagnose-Daten ihrer Patienten, während sie schreibenden auf die ICF-Daten zugreifen können.

Synchronisations-Richtlinien. Wir haben drei Synchronisations-Richtlinien festgelegt, die definieren, wann eine Synchronisation der lokalen Daten mit den Daten des Backends stattfindet. Für den Datenbestand der gesamten Anwendung kann global eine der folgenden Optionen festgelegt werden:

- **ALWAYS:** Bei dieser Einstellung erfolgt die Synchronisation sofort. Einzige Voraussetzung ist das Vorhandensein einer Verbindung zum Netzwerk. Veränderte Daten werden in regelmäßigen Abständen an den Server übermittelt beziehungsweise von diesem abgefragt und in die lokale Datenbank eingepflegt. Ist keine Netzverbindung verfügbar, bleiben die Daten lokal gespeichert und

der Synchronisationsmechanismus greift, sobald die Netzverbindung wieder besteht.

- **MANUAL:** Der Synchronisationsmechanismus wird manuell durch den Nutzer gestartet. Bis dahin werden alle Datenänderungen ausschließlich in der lokalen Datenbank abgespeichert.

- **WLAN:** Voraussetzung für die Durchführung einer Synchronisation ist das Vorhandensein eines WLAN-Netzes. Somit sollen die Kosten, die bei großem Datenverkehr über mobiles Internet entstehen, vermieden werden. Ist also eine WLAN-Verbindung vorhanden, entspricht die Synchronisation der von *ALWAYS*. Ist keine vorhanden, so entspricht der Mechanismus dem von *MANUAL* bis wieder eine WLAN-Verbindung aufgebaut werden kann.

Synchronisations-Flags. Um den Synchronisationszustand der einzelnen Ressourcen, in unserem Fall also der Tabelleneinträge in der Datenbank, zu beschreiben, wurden zusätzliche Tabellenspalten eingeführt:

- **SID:** Die Server-ID entspricht dem Primary Key der Ressource am Server und wird zur serverseitigen Zuordnung benötigt.

- **TIMESTAMP:** Der Timestamp einer Ressource wird bei jedem schreibenden Zugriff auf diese aktualisiert, um den Zeitpunkt der letzten Änderung verfolgen zu können.

- **DIRTY:** Dieses Flag markiert Ressourcen, die lokal verändert wurden. Sie sind in keinem konsistenten Zustand mit dem Backend-Server und müssen synchronisiert werden.

- **DELETED:** Dieses Flag markiert eine gelöschte Ressource. Erst wenn die Lösch-Operation dem Server erfolgreich mitgeteilt wurde, wird die Ressource am Client tatsächlich gelöscht.

- **SYNC:** Mit diesem Flag werden Ressourcen markiert, die sich in einem Synchronisationsvorgang befinden.

- **LAST_RESULT:** Hier wird das Ergebnis der letzten REST-Anfrage dieser Ressource gespeichert. Das Ergebnis ist stets ein HTTP-Code. Sollte beispielsweise das Ergebnis der letzten Anfrage der Code "401 Forbidden" sein, so kann die Anwendungslogik beispielsweise vorsehen, dass dem Nutzer eine Oberfläche zur Eingabe seines Passworts gezeigt und daraufhin die Anfrage noch einmal versendet wird.

Um Klarheit über die Synchronisationslogik zu schaffen, werden die Vorgänge in den nächsten Abschnitten anhand von Beispielen detailliert beschrieben.

Einfügen eines neuen Datensatzes. Das Einfügen eines neuen Datensatzes in die lokale Datenbank entspricht bei einer REST-basierten Architektur einer POST-Anfrage am Server, um den neuen Datensatz auch am Backend anzulegen. Diesen Vorgang zeigen wir am Beispiel der Erfassung einer neuen ICF-Beurteilung für einen einzelnen ICF-Code. Die Tabellen 5.1 und 5.2 zeigen den lokale Tabellenausschnitt vor und nach der Synchronisation mit dem Server. Der Eintrag der Spalte *sync* in der Tabelle 5.1 wird mit dem Start der POST-Anfrage auf *true* gesetzt. In diesem Fall weiß die Routine, dass ein POST auszuführen ist, da für diesen Tabelleneintrag noch keine *sid* des Servers bekannt ist und *dirty* gesetzt ist. Nach der erfolgreichen Ausführung der POST-Methode am Server, wird die *sid* 864 zurückgeliefert. Mit dieser ID, die dem Primary Key des Tabelleneintrags am Backend-Server entspricht, kann diese Tabellenzeile von nun an am Server zugeordnet werden. Weiters wird noch *timestamp* auf den Zeitpunkt der Aktualisierung gesetzt, *sync* und *dirty* werden wieder auf *false* gesetzt und *last_result* war in diesem Fall der HTTP-Code *201* Created.

id	icf_code	wert	sid	timestamp	dirty	deleted	sync	last_result
1	54	3	null	2012-11-01 13:00	true	false	true	null

Tabelle 5.1: Lokales Insert während POST-Anfrage.

id	icf_code	wert	sid	timestamp	dirty	deleted	sync	last_result
1	54	3	864	2012-11-01 13:01	false	false	false	201

Tabelle 5.2: Lokales Insert nach POST-Anfrage.

Änderung eines bestehenden Datensatzes. In folgendem Beispiel wird der Wert der zuvor durchgeführten ICF-Beurteilung von 3 auf 4 geändert. Die Tabellen 5.3 und 5.4 zeigen wiederum den Ausschnitt aus der lokalen Datenbanktabelle vor und nach Durchführung der Synchronisation. Die Synchronisationsroutine weiß in diesem Fall, dass ein Update auf dem entsprechenden Eintrag am Server auszuführen ist, da *dirty* gesetzt ist. Eine POST-Anfrage kommt nicht in Frage, da die *sid* bereits gesetzt und der Eintrag dem Backend-Server daher bereits bekannt ist. Wie in Tabelle 5.4 ersichtlich, war das Ergebnis der UPDATE-Anfrage erfolgreich, nachdem *sync* und *dirty* wieder auf *false* gesetzt wurden, der *timestamp* aktualisiert wurde und als *last_result* der HTTP-Code *200* OK eingetragen wurde.

id	icf_code	wert	sid	timestamp	dirty	deleted	sync	last_result
1	54	4	864	2012-11-01 14:00	true	false	true	201

Tabelle 5.3: Lokales Update während UPDATE-Anfrage.

id	icf_code	wert	sid	timestamp	dirty	deleted	sync	last_result
1	54	4	864	2012-11-01 14:01	false	false	false	200

Tabelle 5.4: Lokales Update nach UPDATE-Anfrage.

Löschen eines bestehenden Datensatzes. In diesem Beispiel wird die ICF-Erfassung gelöscht. Tabelle 5.5 zeigt nun den entsprechenden Tabelleneintrag vor Durchführung der DELETE-Anfrage an das Backend. Muss das Löschen eines Tabelleneintrages erst mit dem Server synchronisiert werden, so darf der Eintrag nicht sofort gelöscht werden. Er wird nur als gelöscht markiert. Dies geschieht anhand des Flags *deleted*. Erst wenn die DELETE-Anfrage erfolgreich vom Server zurückkehrt, wird der Tabelleneintrag lokal tatsächlich gelöscht.

id	icf_code	wert	sid	timestamp	dirty	deleted	sync	last_result
1	54	4	864	2012-11-01 15:00	true	true	true	200

Tabelle 5.5: Lokales Delete während DELETE-Anfrage.

Behandlung von Fehlerfällen. Nachdem die grundsätzliche Synchronisationslogik beschrieben wurde, stellen wir nun exemplarisch die Behandlung eines Fehlerfalles vor. Gehen wir erneut von der Erfassung einer ICF-Beurteilung aus. Tabelle 5.6

zeigt die Erfassung mit dem *timestamp* um *13:00* Uhr. Die aktuelle Synchronisations-Policy sieht *WLAN* vor, das zu diesem Zeitpunkt nicht vorhanden ist. Daher wird die Synchronisation angestoßen, als der Mitarbeiter später am Tag wieder ein WLAN-Netz zur Verfügung hat. Doch diesmal schlägt die Synchronisation fehl, da der AuthToken des Nutzers nicht mehr gültig ist. Der Server liefert als *last_result 403* Forbidden zurück, wie in Tabelle 5.7 dargestellt. Dieser Fehler tritt auf, wenn der Nutzer keine Berechtigung zur Durchführung dieser Operation am Backend-Server hat. Anscheinend schlug also die Authentifizierung des Nutzers am Backend fehl. Die Synchronisationslogik benachrichtigt daraufhin den Nutzer über den Fehler und fordert ihn zur Eingabe seines Passworts auf. Daraufhin wird erneut versucht die ICF-Erfassung zu synchronisieren. Diesmal erfolgreich, wie in Tabelle 5.8 ersichtlich.

id	icf_code	wert	sid	timestamp	dirty	deleted	sync	last_result
1	54	3	null	2012-11-01 13:00	true	false	false	null

Tabelle 5.6: Lokales Insert vor POST-Anfrage.

id	icf_code	wert	sid	timestamp	dirty	deleted	sync	last_result
1	54	3	null	2012-11-01 15:08	true	false	false	403

Tabelle 5.7: POST-Anfrage schlug fehl.

id	icf_code	wert	sid	timestamp	dirty	deleted	sync	last_result
1	54	3	864	2012-11-01 15:10	false	false	false	200

Tabelle 5.8: POST-Anfrage erfolgreich.

Limitierungen der Synchronisationslogik. Wie bereits angedeutet, stößt die implementierte Synchronisationslogik beim Auftreten von Konflikten an ihre Grenzen. Dies kann der Fall sein, wenn dieselben Datensätze gleichzeitig auf mehreren lokalen Datenbanken verändert werden. Auch für diesen Fall möchten wir zum besseren Verständnis ein Beispiel bringen. Sowohl Client 1 als auch Client 2 holen sich mit einer GET-Anfrage den zentral gespeicherten Patienten-Eintrag des Patienten Huber. Der Server-Eintrag ist in Tabelle 5.9 dargestellt. Client 2 ändert, wie Tabelle 5.10 zeigt, die Adresse seines lokalen Datensatzes von *Wien* auf *Graz* und synchronisiert das UPDATE zum Server. Dieser aktualisiert den Eintrag in der zentralen Datenbank entsprechend. Client 1 ändert indes den Namen des lokalen Datenbankeintrages 5.11 von *Leopold* auf *Leo* und synchronisiert erst abends das UPDATE an

den Server. Dieser überschreibt abermals den Eintrag in der zentralen Datenbank mit den Daten von Client 1. Das Problem in diesem Fall ist, dass Client 1 seine Änderungen auf einem ungültigen Datenbankeintrag vornimmt. Denn der Eintrag für den Patienten enthält zum Zeitpunkt des Updates von Client 1 als *Stadt* nicht mehr *Wien*, sondern bereits *Graz*. Doch wie der endgültige Server-Eintrag in Tabelle 5.12 zeigt, hat das UPDATE des Client 1 die Städte-Änderung durch Client 2 überschrieben.

id	vorname	name	stadt	timestamp
864	Leopold	Huber	Wien	2012-11-01 13:00

Tabelle 5.9: Server-Eintrag der Ausgangssituation.

id	vorname	name	stadt	sid	timestamp	dirty	deleted	sync	last_result
1	Leopold	Huber	Graz	864	2012-11-01 13:10	false	false	false	200

Tabelle 5.10: Eintrag in Client 2 nach lokalem Update und UPDATE-Anfrage am Server.

id	vorname	name	stadt	sid	timestamp	dirty	deleted	sync	last_result
1	Leo	Huber	Wien	864	2012-11-01 18:00	false	false	false	200

Tabelle 5.11: Eintrag in Client 1 nach lokalem Update und UPDATE-Anfrage am Server.

id	vorname	name	stadt	timestamp
864	Leo	Huber	Wien	2012-11-01 18:00

Tabelle 5.12: Server-Eintrag nach Synchronisation.

Unserer Meinung nach gibt es drei Ansätze zur Lösung dieser Problematik:

1. **Datenpartitionierung:** Der von uns derzeit gewählte Ansatz ist, wie bereits beschrieben, die Daten durch organisatorische Maßnahmen so zu partitionieren, dass ein bestimmter Datensatz nur durch eine Person verändert werden darf. Dadurch werden Konflikte, wie oben beschrieben, von Vornherein vermieden und müssen nicht durch aufwändige Konfliktlösungsverfahren behandelt werden.

2. **Implementierung einer "Always-On"-Anwendung:** Würde das Update des Client 2 im vorigen Beispiel sofort mit dem Server synchronisiert werden,

könnte dieser durch ein sogenanntes Conditional PUT dem Client rückmelden, dass sein Update auf einem veralteten Datensatz beruht und daher ungültig ist. Der Client könnte sofort reagieren, indem die aktuelle Version des Datensatzes angefordert und das Update, falls erforderlich, noch einmal durchgeführt wird. Diese Vorgehensweise ist bei unserer "Occassionally-Connected"-Anwendung jedoch nicht möglich. Denn wird der Datenbankeintrag nicht sofort, sondern erst im Nachhinein aufgrund der Tabelleninformation synchronisiert, ist nicht mehr ersichtlich, welche Daten der Tabellenzeile geändert wurden und wie die veraltete, lokale Kopie mit dem aktuellen Datensatz am Server zusammengeführt werden muss. Für derartige Konfliktsituationen wäre eine detailliertere Protokollierung der Änderungen notwendig.

3. **Verwendung einer vollständigen Mergereplikation:** Kann eine entsprechende Datenpartitionierung zur Konfliktvermeidung nicht vorausgesetzt werden. Und ist weiters eine "Always-On"-Anwendung nicht erwünscht, so muss eine vollständige Logik zur Synchronisation und Konfliktauflösung von Mergereplikationen implementiert werden. Wie bereits erwähnt, gibt es zahlreiche Tools, die vorgefertigte Lösungen für diese Synchronisationsproblematik anbieten. Für die Umsetzung dieser prototypischen Anwendung ist die Verwendung einer allumfassenden Synchronisationslogik bzw. eines entsprechenden Tools zum jetzigen Zeitpunkt jedoch nicht zweckmäßig.

Transaktionen. In der derzeitigen Anwendung spielen Transaktionen keine Rolle. Die Datenbankoperationen, die der Nutzer im Prototypen durchführen kann, sind voneinander unabhängig und erfordern somit keine Transaktion. Allerdings spielen Transaktionen sicher eine wichtige Rolle für künftige Erweiterungen der Anwendung. Aus diesem Grund wird im Folgenden beschrieben, wie die derzeitige Architektur auf einfache Weise um Transaktionen erweitert werden kann.

Die mobile Anwendung fasst zusammengehörigen Operationen zu einer Transaktion zusammen und fügt sie als solche in die lokale Datenbank ein. Für die spätere Synchronisation der lokalen Transaktion zum Server muss aus der Datenbank ersichtlich sein, welche Operationen zu einer Transaktion zusammengefasst werden müssen. Hierzu muss das Datenbankmodell um eine Tabelle "Transaktionen" erweitert werden. Diese Tabelle ordnet eine eindeutige Transaktions-ID allen zu einer

Transaktion gehörenden Operationen zu. Es wird also eine Transaktions-ID den Datenbanktabellen und den relevanten Zeilen dieser Tabellen zugeordnet. Somit können die Operationen zum Synchronisationszeitpunkt zu einer Transaktion zusammengefasst werden. Die Informationen für die Ausführung der einzelnen Operationen der Transaktion sind aus den einzelnen Einträgen der Tabellen ersichtlich. Nachdem bereits in Abschnitt 2.2.4 ausführlich beschrieben wurde, wie Transaktionen mit einer REST-basierten Architektur zu gestalten sind, verweisen wir für die serverseitige Implementierung auf unsere vorherigen Ausführungen zu diesem Thema.

5.3.1 Typische Anwendungsfälle

Nachdem die Synchronisationslogik auf Datenbank-Ebene erklärt wurde, veranschaulichen wir die Interaktion der einzelnen Anwendungskomponenten bei der Synchronisation anhand detaillierter Sequenzdiagramme.

Übermittlung neuer Daten an das Backend. Abbildung 5.5 zeigt die Übermittlung lokal erfasster Daten an den Backend-Server. Dazu läuft, je nach konfigurierter Synchronisations-Richtlinie, in periodischen Abständen oder auf explizite Anfrage, der *SyncAdapter* als Hintergrundprozess. Um neue Daten ans Backend zu übermitteln, werden in den *SyncConfigs* die für den aktuellen Nutzer zu synchronisierenden *SyncDAOs* abgefragt. Für diese werden die zu synchronisierenden Daten – markiert durch die Synchronisations-Flags *dirty*=1 und *sid*=NULL – ermittelt und *Datentransfer-Objekte* generiert, die mittels einer POST-Anfrage an das Backend gesendet werden. *Datentransfer-Objekte* sind, wie bereits erwähnt, Java Beans, die den Zustand einer Ressource kapseln und übers Netzwerk übertragen werden können. Dort werden die neuen Daten in die zentrale Datenbank eingefügt. Als Antwort kommt eine *PostResponse* zurück, die die Server-IDs (*sids*) der Datensätze liefert. Abschließend werden die *sids* und die Synchronisations-Flags aktualisiert.

Abfrage serverseitiger Aktualisierungen. Neben der Übermittlung lokaler Änderungen an das Backend, ist die regelmäßige Abfrage von Aktualisierungen am Backend ein typisches Anwendungsszenario. Dieses ist in Abbildung 5.6 dargestellt. Der *SyncAdapter* kennt die abzufragenden Backend-Ressourcen durch die *SyncConfigs*. Für alle abzufragenden Ressourcen wird der Zeitpunkt der letzten

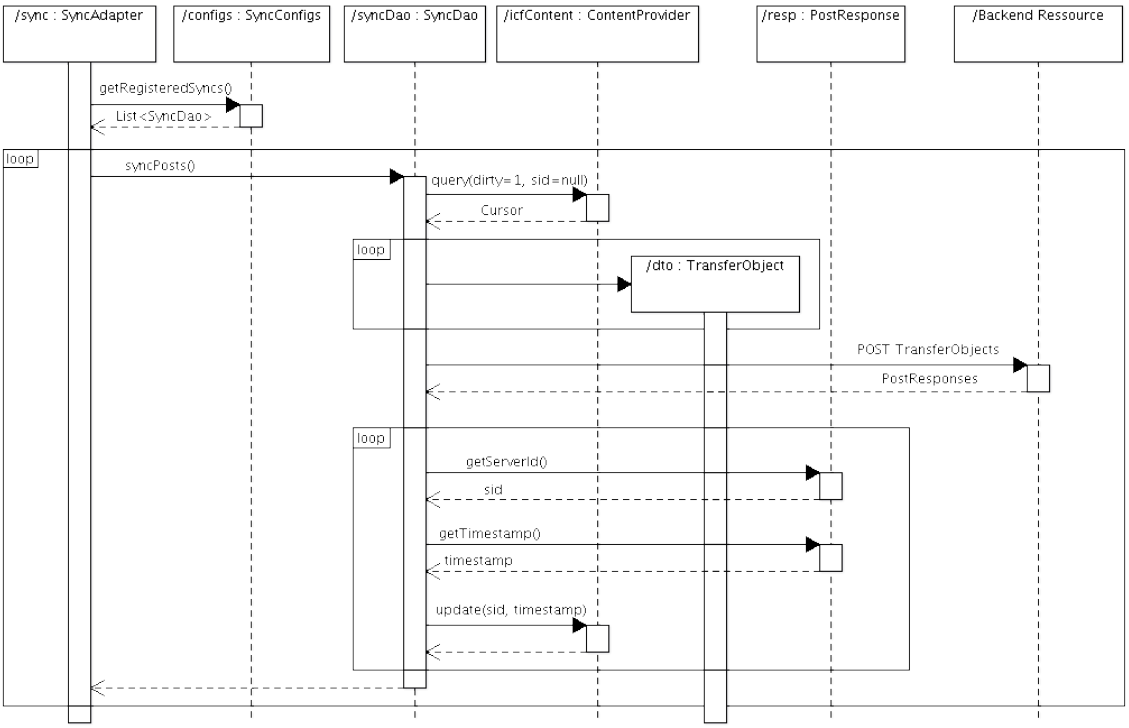

Abbildung 5.5: Sequenzdiagramm zur Übermittlung neuer lokaler Daten an das Backend.

Synchronisation ermittelt, um jeweils nur neuere Daten abzufragen. Mit dem ermittelten *Timestamp* wird eine GET-Anfrage abgesetzt. Diese liefert alle seit der letzten Abfrage aktualisierten Daten in Form von *Datentransfer-Objekten* zurück. Diese *Datentransfer-Objekte* werden anschließend, entsprechend ihrer Zustandsflags, in die lokale Datenbank eingepflegt.

5.4 Sicherheitsmechanismen

Nachdem die mobile Gesundheitsanwendung sensible Gesundheitsdaten verarbeitet, fassen wir hier die, bereits in der Implementierung des Prototypen vorhandenen, Sicherheitsmechanismen zum Schutz dieser Daten zusammen. Die Daten der lokalen Datenbank werden bereits durch das sogenannte Sandboxing des Android-Betriebssystems geschützt. Solange das mobile Endgerät nicht gerootet ist, hat nur die ICF-Anwendung Zugriff auf die Daten der Datenbank. Ein Dritter kann sie nicht einsehen. Zusätzlich wäre eine Verschlüsselung der Datenbank denkbar. Dies ist derzeit nicht realisiert. Beim Datentransport selbst wird auf die Sicherheit von TLS gesetzt, indem die Daten ausschließlich über HTTPS übertragen werden. Weiters

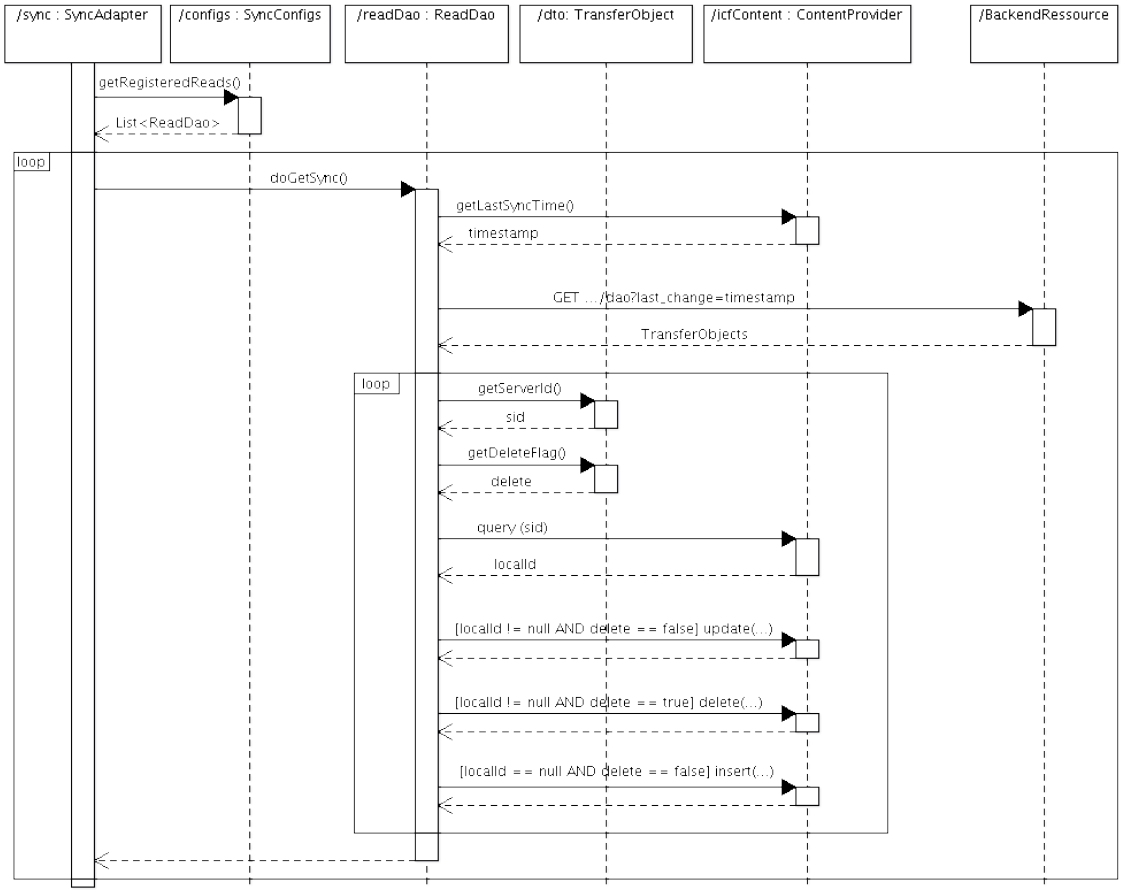

Abbildung 5.6: Sequenzdiagramm zur periodischen Abfrage serverseitiger Aktualisierungen.

werden die Daten durch client- und serverseitige Authentifizierungsmechanismen vor dem Zugriff unbefugter Dritter geschützt. Clientseitig muss sich der Benutzer in regelmäßigen Abständen mit Benutzername und Passwort authentifizieren. Serverseitig erfolgt die Authentifizierung mittels eines begrenzt gültigen AuthTokens über die HTTP Basic Authentication.

6. Zusammenfassung

Vor dem Hintergrund des derzeit stark expandierenden mHealth-Marktes beschäftigte sich diese Arbeit mit der Entwicklung einer mobilen Anwendung zur Dokumentation des Gesundheitszustandes von Patienten. Konkret wurde die Internationale Klassifikation der Funktionsfähigkeit, Behinderung und Gesundheit (ICF) der WHO umgesetzt. Dabei wurde ein innovativer Weg bestritten, da der praktische Einsatz von ICF im Gesundheitsbereich bisher keinen breiten Einzug gehalten hat. Aus diesem Grund war die Entwicklung dieser mobilen Geschäftsanwendung von zahlreichen Unsicherheitsfaktoren begleitet.

Zu Beginn des Entwicklungsprozesses standen wir vor der Entscheidung, welche mobile Plattform unterstützt werden sollte. Die Entscheidung für eine bestimmte Technologie sollte bei der Einführung eines kommerziellen Produktes neben technologischen auch marktpolitische Trends innerhalb einer Branche berücksichtigen, um eine hohe Akzeptanz bei den Anwendern dieser Branche zu erreichen. Nach einer Analyse des mHealth-Marktes kamen wir zu der Ansicht, dass iOS zwar aktuell noch die populärste Plattform für mobile Gesundheitsanwendungen ist, Android für die Zukunft aber das größte Potential zugesprochen wird. Basierend auf der Entscheidung für eine Android-Applikation führten wir die Designentwicklung durch. Nachdem keine konkreten Daten über die Arbeitsvorgänge und Anforderungen der potentiellen Nutzergruppen unserer ICF-Anwendung bekannt waren, war das Risiko eine Anwendung zu entwickeln, die an den tatsächlichen Nutzeranforderungen vorbeigeht, extrem hoch. Aus diesem Grund entschieden wir uns bereits in dieser frühen Phase des Softwareentwicklungprozesses potentielle Anwender einzubeziehen. Wir führten daher mit diplomierten Krankenpfleger/-innen und Physiotherapeut/-innen Usability-Tests basierend auf Papier-Prototypen durch. Somit war es uns trotz begrenzter Ressourcen möglich bereits zu einem frühen Zeitpunkt eine optimierte

Benutzerfreundlichkeit unserer Anwendung zu erreichen.

Auf die Erkenntnisse der Usability-Tests baute die erste softwaretechnische Implementierung der Benutzeroberfläche auf. Die Implementierung der Backend-Anbindung stellte den nächsten Schritt im Entwicklungsprozess dar. Mobile Geschäftsanwendungen werden hier von zahlreichen Qualitäts- und Sicherheitsanforderungen flankiert. Nach einer Analyse SOAP- und REST-basierter Webservices entschieden wir uns, entgegen der SOAP-basierten Architektur zahlreicher Geschäftsanwendungen, für eine REST-basierte Anbindung. Mit dieser Architektur konnten wir eine unnötige Komplexität vermeiden, ohne auf entsprechende Sicherheitsmechanismen verzichten zu müssen. Die prototypische Implementierung der ICF-Anwendung bildete den Abschluss der vorliegenden Arbeit.

Es wurde gezeigt, dass die erfolgreiche Entwicklung mobiler Gesundheitsanwendungen für den professionellen Einsatz von zahlreichen Faktoren beeinflusst wird. Dies beginnt bei der Entscheidung über die zu unterstützenden Plattformen und setzt sich bei der Nutzerorientierung des Softwareentwicklungsprozesses fort, ohne die eine benutzerfreundliche Entwicklung schwer möglich wäre. Eine solide Implementierung der Geschäftslogik und Backend-Anbindung unter Beachtung von Qualitäts- und Sicherheitsanforderungen bildet das Fundament für den Erfolg einer Anwendung.

Der entwickelte Prototyp stellt eine grundlegende Basis für intensive Weiterentwicklungen durch das Unternehmen FERK Systems dar. Zukünftige Arbeiten sind daher in vielfältiger Weise möglich. Ein nächster Schritt wäre beispielsweise die Durchführung eines Usability-Tests basierend auf dem Software-Prototypen, im Sinne eines kontinuierlich nutzerorientierten Softwareentwicklungsprozesses. Ein wesentlicher Erfolgsfaktor für die Anwendung wird das ausführliche Testen im realen Berufsalltag von Physiotherapeuten/-innen und Pflegepersonal sein. Weiters ist ein Mechanismus für Nutzerberechtigungen einzuführen. Es ist ein Berechtigungssystem zu entwickeln, das einzelnen Nutzern bzw. Nutzerrollen (Mediziner, Physiotherapeut, Pflegepersonal) Rechte an spezifischen Daten zuordnet. Weitere Überlegungen könnten auch in Richtung Datenbankabgleich angestellt werden. Kann eine Datenpartitionierung bei einem größeren Funktionsumfang der Anwendung aus organisatorischer Sicht nicht mehr gewährleistet werden, ist über den Einsatz einer vollständigen Mergereplikation nachzudenken.

Abbildungsverzeichnis

1.1 Pflegeprozess der WHO. 18

1.2 Beispiel einer mobilen IT-Unterstützung im Pflegeprozess. 19

1.3 Komponenten der ICF. 20

4.1 Testumgebung der Usability-Tests. 65

4.2 Erfolgsraten bei den einzelnen Testaufgaben. 69

4.3 Patientenblatt. 71

4.4 Erfassung einer ICD-Diagnose. 72

4.5 Verweis auf die Erfassung eines ICF-Krankheitsbildes. 72

4.6 ICF-Erfassungsdialog. 73

4.7 Information zu den Beurteilungswerten des Codes ”d450 Gehen“. . . . 73

4.8 Liste ICF-Trendübersicht. 76

4.9 Menü der ICF-Auswertung 76

4.10 Detailansicht ICF-Code ”d450 Gehen“. 77

4.11 Farbkarte Übersicht. 79

4.12 Farbkarte Details. 79

4.13 Aktuelle ICF-Beurteilung. 81

5.1 Use-Case-Diagramm zur Authentifizierung und Synchronisation. . . . 84

5.2 Use-Case-Diagramm zur ICD- und ICF-Datenmanipulation. 85

5.3 Use-Case-Diagramm zur ICF-Auswertung. 86

5.4 Software-Architektur. 89

5.5 Sequenzdiagramm zur Übermittlung neuer lokaler Daten an das Backend. 101
5.6 Sequenzdiagramm zur periodischen Abfrage serverseitiger Aktualisierungen. 102

Tabellenverzeichnis

1.1 Beispiel einer ICD-Codierung. 21

3.1 Gegenüberstellung der Client-Plattformen und -Technologien. 58

4.1 Anzahl der Testpersonen pro Nutzergruppe. 64
4.2 Ergebnisse nach Berufsgruppen. 69
4.3 Ergebnisse nach Smartphone-Nutzung. 70

5.1 Lokales Insert während POST-Anfrage. 95
5.2 Lokales Insert nach POST-Anfrage. 95
5.3 Lokales Update während UPDATE-Anfrage. 96
5.4 Lokales Update nach UPDATE-Anfrage. 96
5.5 Lokales Delete während DELETE-Anfrage. 96
5.6 Lokales Insert vor POST-Anfrage. 97
5.7 POST-Anfrage schlug fehl. 97
5.8 POST-Anfrage erfolgreich. 97
5.9 Server-Eintrag der Ausgangssituation. 98
5.10 Eintrag in Client 2 nach lokalem Update und UPDATE-Anfrage am Server. 98
5.11 Eintrag in Client 1 nach lokalem Update und UPDATE-Anfrage am Server. 98
5.12 Server-Eintrag nach Synchronisation. 98

Abkürzungsverzeichnis

ACID	Atomicity, Consistency, Isolation, Durability
AES	Advanced Encryption Standard
API	Application Programming Interface
BYOD	Bring Your Own Device
BYOD	Bring your own Device
HTML5	Hypertext Markup Language 5
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICD	International Classification of Diseases
ICF	International Classification of Functioning, Disability and Health
JSON	Java Simple Object Notation
MDM	Mobile Device Management
ME	Mobile Edition
ORM	Object Relational Mapping
POE	Post Once Exactly
REST	Representational State Transfer
RPC	Remote Procedure Call

SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WHO	World Health Organization
WS	Webservice
WSDL	Webservice Description Language

Wissenschaftliche Referenzen

[1] Allamaraju, S.: *RESTful Web Services Cookbook*. Yahoo!, Inc., 2010

[2] Andry, F. ; Wan, L. ; Nicholson, D.: A Mobile Application Accessing Patients' Health Records through a REST API - How Rest-style Architecture can Help Speed up the Development of Mobile Health Care Applications. In: *HEALTHINF'11*, 2011, S. 27–32

[3] Ashworth, Dechanoz B.: *People's needs for nursing care: a European study*. World Health Organisation, Regional Office for Europe, 1987

[4] Bastien, J.M.C.: Usability testing: a review of some methodological and technical aspects of the method. In: *International Journal of Medical Informatics* 79 (2010), Nr. 4, S. e18–e23. – ISSN 1386–5056

[5] Bender, D.: *Voraussetzungen für die nachhaltige Anwendung der Internationalen Klassifikation der Funktionsfähigkeit, Behinderung und Gesundheit (ICF) in der Rehabilitationspraxis Ergebnisse einer Analyse im Spannungsfeld von globaler Konzeption und lokaler Umsetzung*. 4. Tectum Verlag, 2010 (Wissenschaftliche Beiträge aus dem Tectum Verlag: Pädagogik 21). – ISBN 9783828824867

[6] Bertram, J. ; Kleiner, C.: Mobile Apps in Enterprise-Anwendungen unter Berücksichtigung von Sicherheitsaspekten. In: *Smart Mobile Apps*. Springer Berlin Heidelberg, 2012 (Xpert.press). – ISBN 9783642222580, S. 253–267

[7] Bloice, M. ; Simonic, K. ; Kreuzthaler, M. ; Holzinger, A.: Development of an interactive application for learning medical procedures and clinical decision making. In: *Proceedings of the 7th conference on Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer So-*

ciety: information Quality in e-Health. Berlin, Heidelberg : Springer-Verlag, 2011 (USAB'11). – ISBN 9783642253638, S. 211–224

[8] Boivie, I. ; Åborg, C. ; Persson, J. ; Löfberg, M.: Why usability gets lost or usability in in-house software development. In: *Interacting with Computers* 15 (2003), Nr. 4, S. 623–639. – ISSN 0953–5438

[9] Buchner, R. ; Mirnig, N. ; Weiss, A. ; Tscheligi, M.: Evaluating in real life robotic environment: Bringing together research and practice. In: *RO-MAN, 2012 IEEE*, 2012. – ISSN 1944–9445, S. 602–607

[10] Butz, A. ; Kruger, A.: User-Centered Development of a Pervasive Healthcare Application. In: *Pervasive Health Conference and Workshops, 2006*, 2006, S. 1–8

[11] Butz, A. ; Kruger, A.: User-Centered Development of a Pervasive Healthcare Application. In: *Pervasive Health Conference and Workshops, 2006*, 2006, S. 1–8

[12] Cooke, L. ; Mings, S.: Connecting usability education and research with industry needs and practices. In: *Professional Communication, IEEE Transactions on* 48 (2005), Nr. 3, S. 296–312. – ISSN 0361–1434

[13] Coulouris, G. ; Dollimore, J. ; Kindberg, T.: *Distributed Systems*. 4. Pearson Education Limited, 2005

[14] Fielding, R.T.: *Architectural styles and the design of network-based software architectures*, University of California, Irvine, Diss., 2000

[15] Fiotakis, G. ; Raptis, D. ; Avouris, N.: Considering Cost in Usability Evaluation of Mobile Applications: Who, Where and When. In: *Human-Computer Interaction – INTERACT 2009* Bd. 5726. Springer Berlin Heidelberg, 2009. – ISBN 9783642036545, S. 231–234

[16] Hein, M. ; Zeller, H.: *Java Web Services*. Addison-Wesley Verlag, 2003

[17] Höll, B. ; Spat, S. ; Plank, J. ; Schaupp, L. ; Neubauer, K. ; Beck, P. ; Chiarugi, F. ; Kontogiannis, V. ; Pieber, T. ; Holzinger, A.: Design of

a Mobile, Safety-Critical in-Patient Glucose Management System. In: *Studies in health technology and informatics* (2011), S. 950–954. – ISSN 0926–9630

[18] Holzinger, A.: Usability engineering methods for software developers. In: *Commun. ACM* 48 (2005), Nr. 1, S. 71–74. – ISSN 0001–0782

[19] Holzinger, A. ; Errath, M.: Mobile computer Web-application design in medicine: some research based guidelines. In: *Univers. Access Inf. Soc.* 6 (2007), Nr. 1, S. 31–41. – ISSN 1615–5289

[20] Holzinger, A. ; Kosec, P. ; Schwantzer, G. ; Debevc, M. ; Hofmann-Wellenhof, W. ; Frühauf, J.: Design and development of a mobile computer application to reengineer workflows in the hospital and the methodology to evaluate its effectiveness. In: *Journal of Biomedical Informatics* 44, Nr. 6, S. 968–977. – ISSN 1532–0464

[21] Holzinger, A. ; Treitler, P. ; Slany, W.: Making Apps Useable on Multiple Different Mobile Platforms: On Interoperability for Business Application Development on Smartphones. In: *Multidisciplinary Research and Practice for Information Systems* Bd. 7465. Springer Berlin Heidelberg, 2012. – ISBN 9783642324970, S. 176–189

[22] Horsky, J. ; McColgan, K. ; Pang, J.E. ; Melnikas, A.J. ; Linder, J.A. ; Schnipper, J.L. ; Middleton, B.: Complementary methods of system usability evaluation: Surveys and observations during software design and development cycles. In: *Journal of Biomedical Informatics* 43 (2010), Nr. 5, S. 782–790. – ISSN 1532–0464

[23] Höll, B. ; Spat, S. ; Plank, J.: Design einer mobilen Anwendung für das stationäre Glukosemanagement. In: *eHealth2011*, 2011, S. 51–57

[24] Inostroza, R. ; Rusu, C. ; Roncagliolo, S. ; Jimenez, C. ; Rusu, V.: Usability Heuristics for Touchscreen-based Mobile Devices. In: *Information Technology: New Generations (ITNG), 2012 Ninth International Conference on*, 2012, S. 662–667

[25] Kjeldskov, J. ; Stage, J.: New techniques for usability evaluation of mobile systems. In: *International Journal of Human-Computer Studies* 60 (2004), Nr. 5-6, S. 599–620. – ISSN 1071–5819

[26] Larusdottir, M.: Usability Evaluation in Software Development Practice. In: *Human-Computer Interaction – INTERACT 2011* Bd. 6949. Springer Berlin Heidelberg, 2011. – ISBN 9783642237676, S. 430–433

[27] Lee, K. B. ; Grice, R.A.: Developing a new usability testing method for mobile devices. In: *Professional Communication Conference, 2004. IPCC 2004. Proceedings. International*, 2004, S. 115–127

[28] Liu, C. ; Zhu, Q. ; Holroyd, K. ; Seng, E.: Status and trends of mobile-health applications for iOS devices: A developer's perspective. In: *Journal of Systems and Software* 84 (2011), Nr. 11, S. 2022–2033. – ISSN 0164–1212

[29] Melzer, I. ; Dostal, W. ; Jeckle, M. ; Zengler, B.: *Service-orientierte Architekturen mit Web Services*. 4. Spektrum Akademischer Verlag, 2010

[30] Mohamed, K.E. ; Wijesekera, D.: A Lightweight Framework for Web Services Implementations on Mobile Devices. In: *Mobile Services (MS), 2012 IEEE First International Conference on*, 2012, S. 64–71

[31] Nayebi, F. ; Desharnais, J.-M. ; Abran, A.: The state of the art of mobile application usability evaluation. In: *Electrical Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, 2012. – ISSN 0840–7789, S. 1–4

[32] Pardon, G. ; Pautasso, C.: Towards Distributed Atomic Transactions over RESTful Services. In: *REST: From Research to Practice*. Springer New York, 2011. – ISBN 9781441983022, S. 507–524

[33] Pautasso, C. ; Zimmermann, O. ; Leymann, F.: RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision. In: *WWW '08*, 2008, S. 805–814

[34] Peischl, B. ; Ziefle, M. ; Holzinger, A.: A Mobile Information System for Improved Navigation in Public Transport - User Centered Design, Development,

Evaluation and e-Business Scenarios of a Mobile Roadmap Application. In: *DCNET/ICE-B/OPTICS'12*, 2012, S. 217–221

[35] Pilgrim, M.: *HTML5: Up and Running.* O'Reilly Media, Inc., 2010. – ISBN 9780596806026

[36] Rensch, H. ; Bucher, P.: *ICF in der Rehabilitation: Die praktische Anwendung der internationalen Klassifikation der Funktionsfähigkeit, Behinderung und Gesundheit im Rehabilitationsalltag.* 4. 2006. – 289–291 S.

[37] Richardson, L. ; Ruby, S.: *RESTful Web Services.* O'Reilly Media, Inc., 2007

[38] Rubin, J. ; Chisnell, D.: *Handbook of Usability Testing.* 2. Wiley Publishing, Inc., 2008

[39] World Health Organization: *Internationale Klassifikation der Funktionsfähigkeit, Behinderung und Gesundheit.* DIMDI, 2005. – 23 S.

Web-Referenzen

[40] achartengine. URL: http://code.google.com/p/achartengine/. Stand: 30. Dezember 2012.

[41] Apple will update iOS to require user permission for apps to access contact data. URL: http://forums.appleinsider.com/t/143646/apple-will-update-ios-to-require-user-permission-for-apps-to-access-contact-data. Stand: 15. Juni 2012.

[42] Asynchronous Web Service Calls Using WS-Addressing. URL: https://developer.connectopensource.org/display/CONNECTWIKI/Asynchronous+Web+Service+Calls+Using+WS-Addressing. Stand: 20. Juli 2012.

[43] MediFox. URL: http://www.medifox.de/. Stand: 30. Dezember 2012.

[44] Notes on the implementation of encryption in Android 3.0. URL: http://source.android.com/tech/encryption/android_crypto_implementation.html. Stand: 2. Juli 2012.

[45] Storage Options. URL: http://developer.android.com/guide/topics/data/data-storage.html#filesInternal. Stand: 2. Juli 2012.

[46] Unified Modeling Language. URL: http://www.omg.org/spec/UML/2.2/. Stand: 30. Dezember 2012.

[47] Amazon. URL: http://www.amazon.com/. Stand: 30. Dezember 2012.

[48] Apple iPhone. URL: http://www.apple.com/de/iphone/. Stand: 30. Dezember 2012.

[49] assista Soziale Dienste GmbH. URL: http://www.assista.org/home/. Stand: 30. Dezember 2012.

[50] Bakowski, B. Use WS-Addressing within a WebSphere Application Server V6.1 application. URL: http://www.ibm.com/developerworks/library/ws-addressing/index.html. Stand: 30. Dezember 2012.

[51] DIMDI. URL: http://www.dimdi.de/static/de/klassi/icf/icf-projekte.html. Stand: 30. Dezember 2012.

[52] Dobjanschi, V. Developing android REST client applications. URL: http://www.google.com/events/io/2010/sessions/developing-RESTful-android-apps.html. Stand: 30. Dezember 2012.

[53] Dolan, P. Doctors cite ease of use in rapid adoption of tablet computers. URL: http://www.ama-assn.org/amednews/2011/04/18/bisc0418.htm. Stand: 30. Dezember 2012.

[54] Google. URL: http://www.google.com/intl/de/about/. Stand: 30. Dezember 2012.

[55] Google. Intents and Intent Filters. URL: http://developer.android.com/guide/components/intents-filters.html. Stand: 15. Juni 2012.

[56] Google Android. URL: http://www.android.com/. Stand: 30. Dezember 2012.

[57] gSOAP. http://www.cs.fsu.edu/~engelen/soap.html. Stand: 30. Dezember 2012.

[58] ICOsys. URL: http://www.icosys.at/icosys/de/philosophie/index.php. Stand: 30. Dezember 2012.

[59] ICOsys.mobil. URL: http://www.icosys.at/icosys/de/news/news_details.php?id=17. Stand: 30. Dezember 2012.

[60] Apple Inc. Distribute apps to your users. URL: http://www.apple.com/business/accelerator/deploy/app-distribution.html. Stand: 15. Juni 2012.

[61] Apple Inc. iOS Developer Enterprise Program. URL: https://developer.apple.com/support/ios/enterprise.html. Stand: 15. Juni 2012.

[62] Apple Inc. iOS Security. URL: http://images.apple.com/ipad/business/docs/iOS-Security_May12.pdf. Stand: 30. Mai 2012.

[63] International Data Corporation. URL: http://www.idc.com/home.jsp?t=1351618548017. Stand: 30. Dezember 2012.

[64] International Data Corporation. Android and iOS Surge to New Smartphone OS Record in Second Quarter, According to IDC. URL: http://www.idc.com/getdoc.jsp?containerId=prUS23638712. Stand: 30. Dezember 2012.

[65] iOS Developer Library. Implementing custom url schemes. URL: https://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/AdvancedAppTricks/AdvancedAppTricks.html#//apple_ref/doc/uid/TP40007072-CH7-SW50. Stand: 30. Dezember 2012.

[66] Jackson JSON processor. URL: http://jackson.codehaus.org/. Stand: 30. Dezember 2012.

[67] kSOAP 2. URL: http://ksoap2.sourceforge.net/. Stand: 30. Dezember 2012.

[68] ksoap2-android. URL: http://code.google.com/p/ksoap2-android/. Stand: 30. Dezember 2012.

[69] Lebenshilfe Kärnten. URL: http://www.lebenshilfe-kaernten.at/. Stand: 30. Dezember 2012.

[70] Manhattan Research. 75 Percent of U.S. Physicians Own some Form of Apple Device According to new Manhattan Research Study. URL: http://manhattanresearch.com/News-and-Events/Press-Releases/physician-iphone-ipad-adoption. Stand: 30. Dezember 2012.

[71] Microsoft. URL: http://www.microsoft.com/about/en/us/default.aspx. Stand: 30. Dezember 2012.

[72] Microsoft. Mergereplikation. URL: http://msdn.microsoft.com/de-de/library/ms152746.aspx. Stand: 30. Dezember 2012.

[73] J. Muchow. Using Keychain to Store Username and Password. UrL: http://mobiledevelopertips.com/core-services/using-keychain-to-store-username-and-password.html. Stand: 30. Mai 2012.

[74] OASIS. Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.2. URL: http://docs.oasis-open.org/ws-rx/wsrm/v1.2/wsrm.html. Stand: 30. Dezember 2012.

[75] Oracle. Oracle database mobile server 11g. URL: http://www.oracle.com/technetwork/products/database-mobile-server/overview/index.html. Stand: 30. Dezember 2012.

[76] Resting. URL: http://code.google.com/p/resting/. Stand: 30. Dezember 2012.

[77] Restlet. URL: http://www.restlet.org/about/introduction. Stand: 30. Dezember 2012.

[78] RestTemplate Module. URL: http://static.springsource.org/spring-android/docs/1.0.x/reference/html/rest-template.html. Stand: 30. Dezember 2012.

[79] RIM Blackberry. URL: http://de.blackberry.com/. Stand: 30. Dezember 2012.

[80] D. Rousset. Introduction to HTML5 Web Workers: The JavaScript Multithreading Approach. URL: http://msdn.microsoft.com/en-us/hh549259.aspx. Stand: 30. Dezember 2012.

[81] Spies, B. Web Services, Part 1: SOAP vs. REST. URL: http://www.ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest. Stand: 30. Dezember 2012.

[82] Tactio Software International. Global mHealth Developer Survey. URL: http://www.tactiosoft.com/files/GlobalmHealthDeveloperSurvey.pdf, 2010. Stand: 30. Dezember 2012.

[83] S. Tilkov. Using REST for SOA. URL: http://www.infoq.com/presentations/Using-REST-for-SOA. Stand: 30. Dezember 2012.

[84] Turisco, F. ; Garzone, M. Harnessing the value of mhealth for your organization. URL: http://www.csc.com/health_services/insights/69713-harnessing_the_value_of_mhealth_for_your_organization. Stand: 30. Dezember 2012.

[85] W3C. The Media Capture API. URL: http://www.w3.org/TR/media-capture-api/. Stand: 30. Mai 2012.

[86] W3C. Web Services Architecture. URL: http://www.w3.org/TR/ws-arch/#relwwwrest. Stand: 30. Dezember 2012.

[87] WHO. International Classification of Diseases (ICD). URL: http://www.who.int/classifications/icd/en/. Stand: 30. Dezember 2012.

[88] WHO. International classification of functioning, disability and health (icf). URL: http://www.who.int/classifications/icf/en/. Stand: 30. Dezember 2012.

[89] WHO. Towards a Common Language for Functioning, Disability and Health. URL: http://www.who.int/classifications/icf/icfapptraining/en/index.html. Stand: 30. Dezember 2012.

[90] P. Williams. Core Data: Apple's ORM Framework for iOS and the Mac. URL: http://www.dataversity.net/core-data-apples-orm-framework-for-ios-and-the-mac/. Stand: 15. Juni 2012.

[91] World Health Organization. URL: http://www.who.int/en/. Stand: 30. Dezember 2012.

[92] World Wide Web Consortium. URL: http://www.w3.org/. Stand: 30. Dezember 2012.