UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

HOCHSCHULE HEILBRONN
TECHNIK WIRTSCHAFT INFORMATIK

Universität Heidelberg, Hochschule Heilbronn

Studiengang Medizinische Informatik

Bachelor Thesis

Entwicklung einer mobilen Anwendung zur Verbesserung der Informationsgewinnung über (Notfall-) Patienten mithilfe eines NFC-Tags

Andreas Keil

Matr.-Nr.: 184651

18. Februar 2016, Heilbronn

Referent: Prof. Dr.-Ing. Wolfgang Heß

Korreferent: Prof. Dr. Jörg Winckler

Danksagung

Mein Dank richtet sich an Prof. Dr.-Ing. Wolfgang Heß für die Betreuung und Unterstützung bei der Erstellung dieser Arbeit.

Ebenfalls danke ich Wolfgang Zöller, der mir einen Einblick in die Sicht eines Notarztes gegeben sowie viele Fragen beantwortet hat. Weiterhin danke ich Johanna Schirmer für das Erstellen des App Icons und Lisa Böhler, die mir viele Tipps gegeben hat.

Ein weiterer Dank richtet sich auch an alle Korrekturleser sowie an alle, die mich während der Erstellung dieser Arbeit unterstützt haben.

Zusammenfassung

Informationslücken sind in der Notfallmedizin besonders folgenschwer, da sie zu einer ineffizienten Versorgung der Patienten führen und lebensbedrohlich sein können. Sie entstehen, wenn Patienten sich nicht mehr adäquat artikulieren können oder nicht wissen, welche Vorerkrankungen sie haben und welche Medikamente sie einnehmen. So kann es beispielsweise zu schwerwiegenden Wechselwirkungen zwischen Medikamenten kommen oder zu allergischen Reaktionen auf ein Medikament.

Im Rahmen dieser Arbeit soll ein System entwickelt werden, das diese Informationslücken zwischen Patienten und Erst- und Zweithelfern schließen soll. Es besteht aus einer mobilen Anwendung für Android Smartphones und NFC-Tags. Dazu schreiben Patienten ihre medizinischen Notfalldaten mit der mobilen Anwendung auf einen NFC-Tag. Diesen NFC-Tag tragen sie in Form einer Halskette, eines Armbands, eines Schlüsselanhängers o. ä. bei sich. In einem Notfall können Erst- und Zweithelfer diesen NFC-Tag mit der mobilen Anwendung auslesen und bekommen so Informationen über den Patienten geliefert.

Die implementierte mobile Anwendung kann testweise für Evaluationen eingesetzt werden. Für einen produktiven Einsatz müssen allerdings organisatorische sowie rechtliche Pflichten wie der Datenschutz oder das Medizinproduktegesetz berücksichtigt werden.

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abb Abbildung

API Application Programming Interface

BÄK Bundesärztekammer

BLE Bluetooth Low Energy

eGK Elektronische Gesundheitskarte

FHIR Fast Healthcare Interoperability Resources

JSON JavaScript Object Notation

KIS Krankenhausinformationssystem

BMP Bundesmedikationsplan

NDEF NFC Data Exchange Format

NFC Near Field Communication

QR Quick Response

REST Representational State Transfer

SOA Service-orientierte Architektur

URI Uniform Resource Identifier

UX User Experience

XML eXtensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Listingsverzeichnis

1 Einführung

1.1 Gegenstand und Bedeutung

Oft wissen Patienten nicht, welche Krankheiten bei ihnen diagnostiziert wurden oder welche Medikamente sie regelmäßig einnehmen. Diese Informationen sind bei medizinischen Notfällen allerdings von großer Bedeutung [1]. Sie unterstützen Erst- und Zweithelfer beim Aufstellen einer Verdachtsdiagnose oder Einleiten einer Notfalltherapie wie beispielsweise der Gabe eines Medikamentes. Hierbei sind Informationen über Medikamentenunverträglichkeiten oder -allergien des Patienten von großer Wichtigkeit, um Wechselwirkungen zu vermeiden und so dessen Zustand nicht zu verschlechtern. Auch andere Daten wie Notfallkontakte oder Name und Alter des Patienten können sehr hilfreich sein.

Um dieses Problem zu lösen, wird eine mobile Anwendung (auch *App* genannt) für Android entwickelt, mit der Patienten ihre medizinischen Notfalldaten auf einen NFC-Tag schreiben können, den sie bei sich tragen. In einem Notfall können Erst- und Zweithelfer diesen NFC-Tag mit ihrem Android Smartphone und der App auslesen und so wertvolle Informationen über den Patienten und dessen Gesundheitszustand erhalten. Dadurch kann im Idealfall eine verbesserte Patientenversorgung sichergestellt und das Leben des Patienten gerettet werden.

1.2 Problematik

Oft ist die Patientenversorgung im Notfall nicht optimal. Patienten wissen nicht auswendig, welche Medikamente sie einnehmen oder welche Krankheiten sie haben; gerade bei einem Notfall sind Patienten nicht immer bei Bewusstsein oder können sich nicht adäquat artikulieren. Dadurch verstreichen wertvolle Sekunden und Minuten, die lebensrettend sein können. Auch ist nicht immer ein Notfallkontakt bekannt, der den Helfern weitere Informationen über den Patienten geben könnte. All diese Faktoren hindern Erst- und Zweithelfer daran, den Patienten optimal zu versorgen.

1.3 Motivation

Durch diese Arbeit soll eine Möglichkeit entstehen, die Patientenversorgung in einem medizinischen Notfall zu verbessern. Die notwendigen Informationen zur effizienten Patientenversorgung können Erst- und Zweithelfer mithilfe einer App von dem NFC-Tag des Patienten auslesen.

1.4 Problemstellung

(1) Es ist unklar, ob auf einem handelsüblichen NFC-Tag genug Speicherkapazität für alle Notfalldaten zur Verfügung steht.

(2) Die grafische Oberfläche der App muss so gestaltet sein, dass alle ausgelesenen Notfalldaten schnell und übersichtlich dargestellt werden. Außerdem muss der Benutzer die App intuitiv bedienen können.

(3) Die App muss auf den verbreitetsten mobilen Betriebssystemen lauffähig sein.

(4) Es muss entschieden werden, welche medizinischen Notfalldaten auf dem NFC-Tag gespeichert werden sollen.

1.5 Zielsetzung

Im Rahmen des Projekts NExT sollen die folgenden Ziele erreicht werden:

Zu Problem (1):

(1) Daten komprimiert speichern.

Hierzu wird geprüft, wie die Daten komprimiert abgespeichert werden können. Die Vordiagnosen und Medikationen werden mit einer ID oder Alias gesichert.

Aus Ziel (1) folgt, dass die Datenbasis (Vordiagnosen und Medikationen) aktualisierbar sein muss.

Zu Problem (2):

(2) Eine intuitive Benutzeroberfläche ist entwickelt.

Eine möglichst intuitive Benutzeroberfläche soll entwickelt werden.

(3) Die Benutzeroberfläche entspricht den Design Guides der Android Plattform.

Die App entspricht den Design Guides der Android Plattform und garantiert so eine Benutzeroberfläche und **User Experience!** (**User Experience!**), die der Benutzer auch aus anderen Applikationen kennt.

(4) Der Benutzer sieht, wie viel Speicherplatz seine Daten verbrauchen und wie viel Speicherplatz noch zur Verfügung steht.

Während der Benutzer seine Daten eingibt, sieht er, wie viel Speicherplatz noch auf dem NFC-Tag zur Verfügung steht und wie viel Speicherplatz er bereits verbraucht hat.

Zu Problem (3):

(5) Die App ist auf der Android Plattform lauffähig.

Die App ist auf der Android Plattform ab Version 4 lauffähig. Andere Plattformen sowie Android Versionen können aufgrund der Verbreitung ausgeschlossen werden.

Zu Problem (4):

(6) Medizinische Notfalldaten werden auf dem NFC-Tag gespeichert.

Die medizinischen Notfalldaten, die auf dem NFC-Tag gespeichert werden sollen, sind durch eine Umfrage unter Rettungsdienstlern und Notfallmedizinern ausgewählt worden.

1.6 Struktur

Die nachfolgende Liste gibt eine Übersicht über den Aufbau dieser Thesis.

- Kapitel 2 *Grundlagen* definiert einige wichtige Begriffe und Konzepte, die für das Verständnis der Arbeit nötig sind.
- Kapitel 3 *Stand der Technik* zeigt existierende Lösungen für das beschriebene Problem auf.
- Kapitel 4 *Technologieevaluation* stellt verschiedene Technologien vor, mit denen das beschriebene Problem realisiert werden kann.
- Kapitel 5 *Anforderungsanalyse* legt die Anforderungen an das System fest.
- Kapitel 6 *Auswahl der Notfalldaten* beschreibt die durchgeführte Umfrage und die anschließende Auswertung.
- Kapitel 7 *Entwurf* beschreibt den Entwurf des Systems.
- Kapitel 8 *Implementierung* stellt die Implementierung des Entwurfs vor.
- Kapitel 9 *Diskussion & Ausblick* diskutiert das Konzept, den Entwurf und die Implementierung und gibt einen Ausblick auf weitergehende Funktionen sowie mögliche Entwicklungen und Anwendungen.

2 Grundlagen

2.1 Begriffserklärungen

Nachfolgend werden Begriffe definiert, die zentral für das Verständnis dieser Arbeit sind.

2.1.1 Webservice

Ein Webservice ist eine Software, die Maschinen-zu-Maschinen Kommunikation über das Hyper Text Transfer Protokoll erlaubt. Sie ist eine Schnittstelle zu Anwendungsfunktionen einer Software [2, S. 1].

2.1.2 Risikofaktor

Ein Risikofaktor ist eine ungünstige Einflussgröße, die die Auftrittswahrscheinlichkeit einer bestimmen Erkrankung deutlich erhöht [3, S. 45].

2.1.3 Medizinische Notfalldaten

Als medizinische Notfalldaten werden die Daten definiert, die medizinische Informationen über den Patienten darstellen wie beispielsweise Vorerkrankungen oder Medikamente, die der Patient einnimmt.

2.1.4 Medizinischer Notfall

Ein medizinischer Notfall ist eine unvorhergesehene Situation, in der eine Person „körperliche oder psychische Veränderungen im Gesundheitszustand aufweis[t], für welche der Patient selbst oder eine Drittperson unverzügliche medizinische[...] Betreuung als notwendig erachtet“ [4]. Aus dieser Definition folgt, dass medizinische Notfälle zeitkritisch sind.

2.1.5 ICD-10-GM

ICD (engl. *International Statistical Classification of Diseases and Related Health Problems*) ist „die amtliche Klassifikation zur Verschlüsselung von Diagnosen in der ambulanten und stationären Versorgung in Deutschland“ [5]. Die Weltgesundheitsorganisation (WHO) ist der Herausgeber der ICD-Klassifikation (ICD-10-WHO). In Deutschland gilt die Adaption ICD-10-GM. GM steht für *German Modification* und *10* bezeichnet die 10. Revision [5].

In der ICD-10-GM gibt es Kapitel, die verschiedene Krankheiten zusammenfassen [5] wie bspw. *Krankheiten des Nervensystems*, *Kreislaufsystems*, etc. [1] In den Kapiteln sind die Diagnosen in Gruppen organisiert [5].[2]

2.1.6 NExT

NExT steht für **N**fc **E**mergency **T**ag und bezeichnet dieses Projekt und infolgedessen das ganze System. Die NExT-App ist entsprechend die mobile Anwendung, die hier entwickelt wird. Ein NExT-Tag ist ein NFC-Tag, auf den Daten mit der NExT-App geschrieben werden.

2.1.7 Hybride Applikation

Eine hybride Applikation ist eine mobile Anwendung, die mit Webtechniken realisiert ist. Sie wird in einem Web View angezeigt und ist infolgedessen eine mobile Website. Da sie nur in einem Web View angezeigt wird, ist sie auf allen Betriebssystemen lauffähig, die eine Website anzeigen können.

2.2 Near Field Communication (NFC)

NFC ist eine standardisierte, kontaktlose, Zwei-Wege-Technologie zur sicheren Datenübertragung [6]. Mittels elektromagnetischer Induktion können Daten ausgetauscht werden. Dabei werden ein aktives und ein passives Element benötigt. Das passive Element ist oft ein sogenannter *NFC-Tag*.

Die maximale Übertragungsdistanz beträgt zehn Zentimeter [7, S. 87].

2.2.1 NFC-Tag-Technologie

Das NFC-Forum[3] ist eine Vereinigung von mehreren Firmen, um NFC weiter zu standardisieren und zu entwickeln [8]. Es legt vier verschiedene Tagtypen fest: Type 1, Type 2, Type 3 und Type 4. Die verschiedenen Typen orientieren sich dabei an bereits bestehenden RFID Chips verschiedener Hersteller [7, S. 109]. Die Speicherkapazität der Tag-Typen reicht von 48 Byte bis 32kb [9]. Die im Handel erhältlichen NFC-Tags haben oftmals eine Speicherkapazität von weniger als 1 Megabyte.

[1] `https://www.dimdi.de/static/de/klassi/icd-10-gm/kodesuche/onlinefassungen/htmlgm2016/index.htm`. Letzter Zugriff: 17.02.2016

[2] `https://www.dimdi.de/static/de/klassi/icd-10-gm/kodesuche/onlinefassungen/htmlgm2016/index.htm#VI`. Letzter Zugriff: 17.02.2016

[3] `http://nfc-forum.org/`. Letzter Zugriff: 17.02.2016

2.2.2 NFC Data Exchange Format (NDEF)

Das NFC Data Exchange Format ist ein binäres Datenformat. Es kapselt Anwendungsdaten mit Metainformationen, die den Aufbau der Daten und Informationen zur Interpretation angeben [7, S. 120].

2.2.3 NdefRecord

In einem NdefRecord werden die Anwendungsdaten gekapselt. Er besteht aus „einem Header und Datenteil Payload“ [7, S. 121]. In dem Payload sind die eigentlichen Nutzdaten binär enthalten.

2.2.4 NdefMessage

Eine NdefMessage fasst einen oder mehrere NdefRecords zusammen [7, S. 122].

2.3 Service-orientierte Architektur (SOA)

Eine Architektur, bei der Services lose gekoppelt nebeneinander existieren, heißt Service-orientierte Architektur. Die Nutzung eines Services ist unabhängig von dessen Implementierung. Ein Service besitzt eine öffentliche, wohldefinierte Schnittstelle [10].

2.4 Representational State Transfer (REST)

REST wurde erstmals im Jahr 2000 von Roy Thomas Fielding beschrieben. Es ist ein Architekturstil, der beschreibt, wie das World Wide Web aufgebaut ist [11].

In REST gibt 3 Prinzipien:

- Resources

  Eine Ressource ist eine Information wie zum Beispiel ein Bild, Link oder ein Service. Auf sie kann über einen Uniform Resource Identifier (URI) zugegriffen werden [11].

- Representations

  Eine *Representation* bezeichnet die Darstellung der Übertragung der Daten (wie sie übertragen werden) [11].

- Stateless

  Die Kommunikation zwischen Client und Server ist zustandslos, d.h. es werden keine kontextbezogenen Daten auf dem Server gespeichert. Bei jeder Anfrage müssen alle notwendigen Informationen mitgesendet werden [11].

Die Kommunikation geschieht über das Hyper Text Transfer Protokoll.

3 Stand der Technik

Im diesem Kapitel werden bereits existierende Lösungen für das in Kapitel 1 dargestellte Problem beschrieben.

3.1 SOS Notfallkapseln

SOS Notfallkapseln (Abb. 3.1) sind kleine Kapseln, die meist als Anhänger um den Hals getragen werden. Sie enthalten einen kleinen Zettel, auf den die medizinischen Notfalldaten geschrieben werden. Oft reicht der Platz in der Kapsel auch für ein Notfallmedikament aus.

Abbildung 3.1: Eine SOS Notfallkapsel

3.2 Notfallarmband

Auf dem Markt gibt es auch sogenannte Notfallarmbänder wie von Safesport ID[1] oder secur-id[2] (Abb. 3.2). Auf das Armband ist i.d.R. eine Metallplatte eingearbeitet, in welche die Notfalldaten eingraviert werden. Je nach Größe dieser Metallplatte steht unterschiedlich viel Platz für die Daten zur Verfügung.

Einige Notfallarmbänder enthalten dazu noch einen USB-Stick[3]. Über eine Software auf dem USB-Stick kann der Benutzer seine Notfalldaten eingeben und auf dem USB-Stick speichern. Beim Einstecken des Sticks in einen Computer öffnet sich die Software automatisch und die Notfalldaten werden angezeigt.

Abbildung 3.2: Ein Notfallarmband

3.3 Andere Notfall-Apps

Andere Notfall-Apps wie der Notfallpass der *Health App*[4] der Firma Apple Inc. oder *ICE: Notfalldaten*[5] von Appventive speichern die Notfalldaten direkt auf dem Smartphone des Benutzers. Dadurch kann auch noch ein Bild des Benutzers sowie beliebig viele andere Gesundheitsdaten gespeichert werden.

Oft muss das Smartphone entsperrt werden, um die Daten lesen zu können.

---

[1] `http://safesportid.de`. Letzter Zugriff: 17.02.2016
[2] `https://secur-id.net`. Letzter Zugriff: 17.02.2016
[3] bspw. der *UTAG Notfall-USB-Stick DOGATG/CreditCard/Sports*
[4] `https://www.apple.com/de/ios/health/`. Letzter Zugriff am 17.02.2016
[5] `https://play.google.com/store/apps/details?id=com.appventive.ice`. Letzter Zugriff am 17.02.2016

3.4 Elektronische Gesundheitskarte (eGK)

Die Elektronische Gesundheitskarte (eGK) ist eine Smart Card, die der Nachfolger der Versichertenkarte ist. Jeder gesetzlich Krankenversicherte in Deutschland hat eine eGK. Die Karte hat verschiedene Anwendungen wie zum Beispiel die Übermittlung der Patientenstammdaten, Speicherung von Arztbriefen oder das Speichern des elektronischen Arztbriefes [12].

Auf der eGK können auch Notfalldaten gespeichert werden. Dies geschieht auf freiwilliger Basis. Der Patient kann die Daten nicht selbst abändern. In einem medizinischen Notfall können die Daten ohne Zustimmung des Patienten vom Notarzt ausgelesen werden [12]. Diese Anwendung der eGK ist momentan noch nicht verfügbar[6]. Erst ab 2018 sollen die Notfalldaten auf der eGK gespeichert werden [13].

[6]Stand: 06.01.2016

4 Technologieevaluation

Im vorigen Kapitel wurden bereits auf dem Markt existierende Lösungen und deren Funktion beschrieben. In diesem Kapitel werden verschiedene Technologien vorgestellt, mit denen ein Patientennotfallinformationssystem realisiert werden kann. Anschließend werden die Vor- und Nachteile diskutiert.

4.1 iBeacon

iBeacon ist ein Protokoll, das von der Firma Apple Inc. ursprünglich zur Navigation in geschlossenen Räumen entwickelt wurde [14]. Implementierungen des Protokolls nutzen Bluetooth Low Energy (BLE) und werden *Beacons* genannt. Ein Beacon sendet die gleichen Daten in kurzen Abständen hintereinander und braucht daher eine Stromversorgung. Die gesendeten Daten sind nur Informationen über den Standort des Beacons (*„Hier bin ich“*). Sobald ein Gerät das Signal von mindestens zwei Beacons empfängt, kann die aktuelle Position berechnet werden.

Die Reichweite von BLE beträgt je nach Sendeleistung bis zu 50 Meter [15], daher kann theoretisch jedes Gerät in diesem Umkreis die Daten empfangen. Jedoch ist das aus Sicht des Datenschutzes für diese Anwendung nicht akzeptabel.

Aus diesem Grund eignen sich Beacons mit iBeacon als Protokoll nicht zur Implementierung eines Patientennotfallinformationssystems.

4.2 QR-Code

Ein QR-Code ist eine „matrix symbology. The symbols consist of an array of nominally square modules arranged in an overall square pattern [...]“ [16]. QR-Codes beschreiben Informationen auf maschinenverständliche Weise. Sie können auf eine Oberfläche aufgetragen und mithilfe einer Kamera ausgelesen werden. Abbildung 4.1 zeigt einen QR-Code, in den Notfalldaten codiert sind.

Die aktuelle Version 40 des QR-Codes[1] kann bis zu 4296 alphanumerische Zeichen speichern [17]. Dies entspricht 4 Kilobyte an Speicherkapazität.

Die Kosten für die Nutzung eines QR-Codes sind gering, da er nur generiert und zum Beispiel auf ein Blatt Papier gedruckt werden muss, die Generierung eines QR-Codes hingegen ist nicht mit Kosten verbunden. Jedoch ist ein Blatt Papier nicht robust gegen Druck oder Wettereinflüsse und anfällig für Beschädigungen, beispielsweise durch Knicke. Dadurch kann es passieren, dass der QR-Code nicht mehr lesbar ist. Der Druck eines QR-Codes auf robustere Materialien ist mit Aufwand und Kosten verbunden, da er mit Spezialgeräten durchgeführt werden muss, die i.d.R. nur in Unternehmen verfügbar sind.

[1] Stand: 27.01.2016

Aus den dargelegten Gründen ist ein QR-Code nicht geeignet, um darauf Notfalldaten für ein Patientennotfallinformationssystem zu speichern.

Abbildung 4.1: Ein QR-Code mit Notfalldaten

4.3 NFC

Die grundlegende Funktionsweise von NFC wurde bereits in Abschnitt 2.2 erläutert.

Zum Lesen oder Beschreiben eines NFC-Tags muss das Smartphone einen NFC-Sensor besitzen. Auch ein NFC-Tag muss für die Nutzung erworben werden. Dieser bietet bis zu 8 Kilobyte Speicherplatz und kostet rund 1 - 3 Euro.

Die Lesedistanz beträgt theoretisch maximal zehn Zentimeter [7, S. 87], in der Praxis sind es meist unter fünf Zentimeter. Das ist besonders für den Datenschutz relevant.

NFC-Tags gibt es in verschiedenen Ausführungen: als Aufkleber, Karten, Armband oder Anhänger. Ein aus Silikon gefertigtes Armband oder ein Anhänger aus Hartplastik ist sehr robust gegen äußere Einflüsse.

Tabelle 4.1: Vergleich der einzelnen Technologien

	iBeacon	**QR-Code**	**NFC-Tag**
Format	Beacon	Papier	viele, z.B. Anhänger
Kosten	ca. 40 Euro	ca. 1 Euro	ca. 1 - 3 Euro
Speicherkapazität	k.A.	bis zu 3kb	bis zu 8kb
Technik	BLE	binäre Darstellung mit schwarzen und weißen Quadraten	Induktion, Magnetfeld
Lesedistanz	bis zu 50 Meter	abhängig von Scanner	<10 Zentimeter
Smartphone Funktion	Bluetooth	Kamera	NFC

Tabelle 4.1 vergleicht die in diesem Kapitel beschriebenen Technologien. Sie alle bieten die nötigen Grundlagen zur Implementierung eines Patientennotfallinformationssystems. Da NFC-Tags im Gegensatz zu iBeacons und gedruckten QR-Codes robust gegen äußere Einflüsse wie das Wetter oder Druck sind und je nach Tag-Typ viel Speicherplatz bieten, eigenen sie sich zur Implementierung am besten. Außerdem lässt sich ein NFC-Tag nur auf weniger als fünf Zentimeter Entfernung auslesen. Das ist vor allem für den Datenschutz wichtig. Auch die Kosten sind zudem gering.

Zur Entwicklung wird der NFC-Tag *MIFARE DESFire EV1 4k* genutzt. Er bietet eine Speicherkapazität von 4096 Bytes an und ist ein Type 4 Tag sowie NDEF formatiert und kann daher von vielen Android Geräten genutzt werden.

Dieser NFC-Tag wird aufgrund der genannten Eigenschaften zur Benutzung empfohlen.

5 Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an die App beschrieben. Dabei wird auf die Ergebnisse aus Kapitel 4 *Technologieevaluation* zurückgegriffen. Zuerst werden der Workflow und die Nutzung beschrieben, danach die Zielgruppen definiert sowie die funktionalen und nichtfunktionalen Anforderungen aufgelistet.

5.1 Nutzung und Workflow

Die App hat zwei wesentliche Anwendungsfälle:

- Lesende Anwendung

  Dieser Anwendungsfall beschreibt den lesenden Zugriff auf die Notfalldaten, der hauptsächlich in einem medizinischen Notfall erfolgt.

  Wie der Workflow *Tag lesen* in Abbildung 5.1 zeigt, gibt es zwei wesentliche Benutzerinteraktionen: 1. den NFC-Tag an das Smartphone zu halten und 2. ihn wieder wegzunehmen. Dabei hat der Workflow die Voraussetzung, dass die App bereits auf dem Smartphone des Benutzers installiert und nicht geöffnet ist. Ist die App nicht installiert, öffnet sich der Google Play Store mit der App als Startbildschirm. Von dort kann die App installiert werden. Danach startet der Workflow erneut.

- Schreibende Anwendung

  Das Schreiben der Daten ist nicht zeitkritisch, dennoch sollte es so einfach wie möglich sein. Abbildung 5.2 zeigt, wie der Workflow dazu aussehen kann.

  Der Workflow *Tag schreiben* ist anfangs identisch mit dem Workflow *Tag lesen* in Abbildung 5.1. Allerdings hat er hat die Voraussetzung, dass die App zum Start des Workflows geöffnet ist. Nachdem der Workflow *Tag lesen* durchgeführt ist, kann der Benutzer seine Notfalldaten ändern. Zum Speichern auf den NFC-Tag muss er nur auf den Button *Schreiben* tippen und den NFC-Tag an sein Smartphone halten.

5.2 Zielgruppen der Anwendungsfälle

Es gibt zwei Zielgruppen. Die erste umfasst alle Personen, die in einem Notfall Erste oder Zweite Hilfe leisten. Das schließt die Personen ein, die im Rettungsdienst arbeiten oder in Hilfsorganisationen tätig sind. Die zweite Zielgruppe umfasst alle anderen Personen.

Für die zuerst genannte Gruppe ist besonders die lesende Anwendung interessant. Durch diese Anwendung bekommen sie die Notfalldaten des Patienten geliefert. Sie schreiben i.d.R. nicht auf den NFC-Tag.

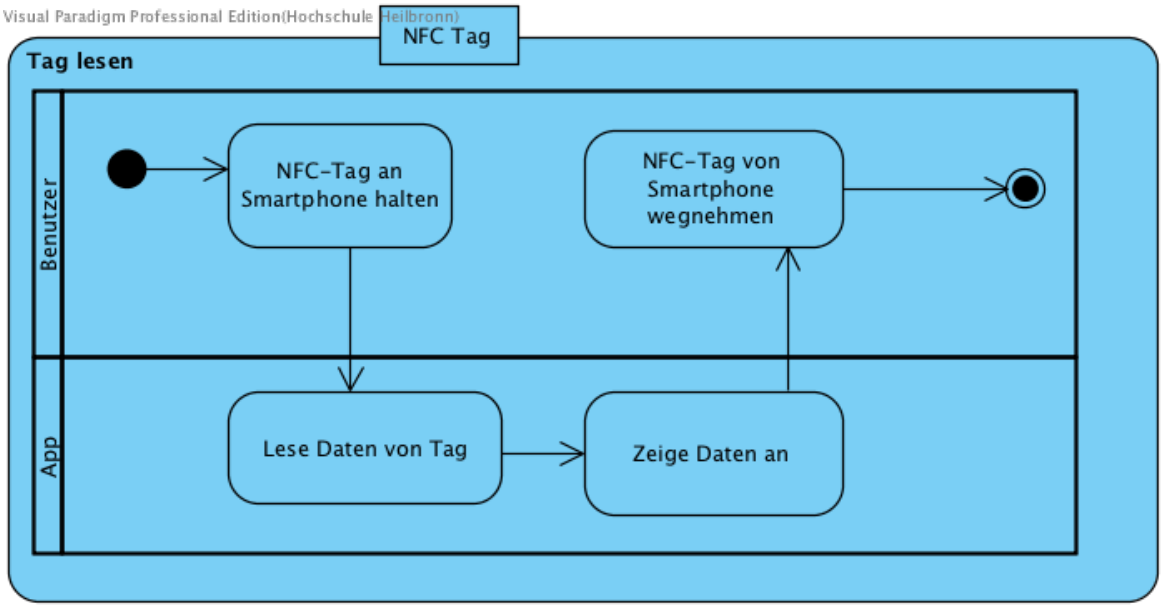


Abbildung 5.1: Workflow *Tag lesen* (Voraussetzung: App ist installiert)

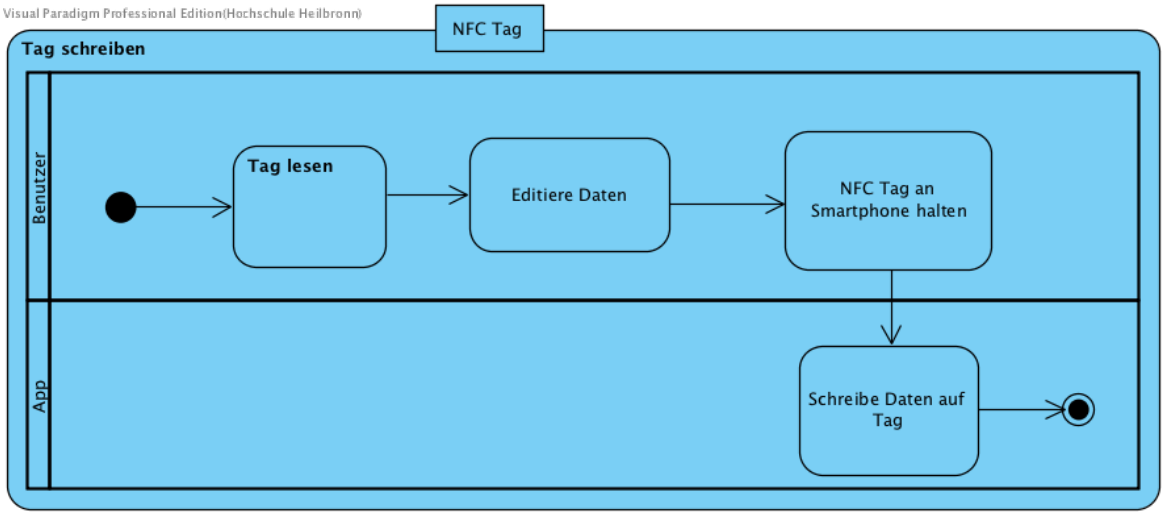


Abbildung 5.2: Workflow *Tag schreiben*

Die zweit genannte Gruppe umfasst alle Personen, die ein Interesse daran haben, ihre Versorgungsqualität in einem medizinischen Notfall zu erhöhen. Das können beispielsweise Personen sein, die schwerwiegende Vorerkrankungen haben oder viele Medikamente einnehmen müssen.

5.3 Funktionale Anforderungen

In diesem Abschnitt werden die funktionalen Anforderungen aufgelistet.

- Der Benutzer soll seine Notfalldaten eingeben können. Diese sind in Kapitel 6 *Auswahl der Notfalldaten* ausgewählt.
- Bei der Eingabe der Daten *Vordiagnosen* und *Medikationen* soll die App dem Benutzer während der Eingabe durch Vorschläge assistieren.
- Die vordefinierten Daten sollen in der App aktualisierbar sein.
- Die Medikationen sollen durch den Datamatrix-Code aus dem Bundesmedikationsplan einlesbar sein.
- Die Daten sollen auf einem NFC-Tag gespeichert werden. Die dabei verwendete Datenstruktur soll keine unnötigen Zeichen enthalten.
- Die Daten sollen verschlüsselt auf dem NFC-Tag gespeichert werden. Der Klartext soll nur mit der NExT App gelesen werden können.
- Zum Auslesen der Daten soll der Benutzer den NFC-Tag an sein Smartphone halten müssen[1].
- Die ausgelesenen Daten sollen übersichtlich auf einer Seite dargestellt werden.
- Ist NFC auf dem Smartphone ausgeschaltet, soll die App darauf hinweisen und den Nutzer auf die entsprechende Einstellungsseite leiten.
- Die App soll anzeigen, wie viel Speicherplatz auf dem NFC-Tag noch frei bzw. verbraucht ist, wenn die aktuell eingegebenen Daten auf den NFC-Tag geschrieben werden.
- Ist die mobile Anwendung nicht installiert und der NFC-Tag wird an das Smartphone gehalten, soll sich der Google PlayStore öffnen, der die App anzeigt.

5.4 Nichtfunktionale Anforderungen

- Die vordefinierten und vom NFC-Tag ausgelesenen Daten sollen jederzeit verfügbar sein.
- Beim Eingeben, Speichern und Auslesen der Daten soll es keine notwendige Abhängigkeit zum Internet geben.
- Es sollen nur NDEF-formatierte NFC-Tags unterstützt werden.
- Es sollen alle Android Versionen ab Android API Version 19 unterstützt werden.

[1] Unter der Annahme, dass die App auf seinem Smartphone installiert ist.

6 Auswahl der Notfalldaten

Ein NFC-Tag bietet nur wenig Speicherplatz, daher können nicht alle Patientendaten darauf gespeichert werden. Um die medizinischen Notfalldaten zu ermitteln, die auf dem NFC-Tag gespeichert werden sollen, wurde eine Umfrage durchgeführt. Außerdem sollte die Umfrage die Priorisierung der Notfalldaten ermitteln,

Die Umfrage wird im Folgenden beschrieben.

Sie bestand aus drei Fragen:

1. Welche Qualifikation haben Sie? Welchen Beruf üben Sie aus?

   Diese Frage ist wichtig, um die Qualität der Antworten festzustellen. Da bspw. ein Notarzt deutlich mehr Erfahrung als ein Sanitäter hat, können die Antworten der Teilnehmer anhand ihrer Qualifikation bewertet werden.

2. Bitte wählen Sie alle Notfalldaten aus, die für Sie in einem medizinischen Notfall am wichtigsten sind.

   Diese Frage stellt die eigentliche Auswahl der Notfalldaten dar.

3. Bitte bringen Sie die zuvor ausgewählten Notfalldaten in eine Reihenfolge (wichtigste zuerst).

   Das Ziel dieser Frage ist es, eine Priorisierung der Notfalldaten zu erhalten zur Feststellung, welche Notfalldaten bevorzugt gespeichert werden sollen. In der Auswertung der Umfrage wird diese Frage ausführlicher behandelt.

Die Umfrage war anonym und wurde hochschulintern auf einem Server des Studiengangs Medizinische Informatik gehostet.

6.1 Relevante Zielgruppe

Relevante Zielgruppen für die Umfrage waren Mitarbeiter im Rettungsdienst:

- Notfallsanitäter
- Rettungsassistenten
- Rettungssanitäter
- Rettungshelfer
- Notärzte

Diese Zielgruppen wurden ausgewählt, da sie entweder ein medizinisches Studium, eine Ausbildung oder Weiterbildung abgeschlossen haben und oft täglich mit Notfallpatienten zu tun haben.

Außerdem waren Personen aus Hilfsorganisationen mit folgenden Qualifikationen relevant:

- Einsatzsanitäter und Sanitäter
- Sanitätshelfer

Die Gruppen Einsatzsanitäter, Sanitäter und Sanitätshelfer wurden zur Gruppe *Sanitäter / Sanitätshelfer* zusammengefasst, da zu deren Qualifikation nur mehrtägige Lehrgänge nötig sind. Außerdem fahren sie in der Regel nicht im Regelrettungsdienst, sondern sind in Hilfsorganisationen meist nur in Sanitätsdiensten tätig und haben so nicht täglich mit Notfällen zu tun.

6.2 Initiale Vorauswahl der medizinischen Notfalldaten

Der Befragte konnte aus verschiedenen Notfalldaten wählen:

- Vorname des Patienten
- Nachname des Patienten
- Geburtsdatum
- Rhesusfaktor
- Blutgruppe
- Arzneimittelallergien
- Dauermedikationen
- Neue Medikationen (einmalige Einnahme, in den letzten Tagen)
- Aktuell bestehende Erkrankungen
- Bestehende Schwangerschaft
- Vordiagnosen (bspw. Herzinfarkt 2013 oder Lungenembolie 2015)
- Risikofaktoren (Alkohol, Raucher, Bewegungsmangel, Drogenkonsum)
- Notfallkontakte (Name und Telefonnummer)
- Name der Krankenkasse
- Versichertennummer des Patienten
- Operationen (was, wann)

Die Notfalldaten wurden teilweise aus dem SAMPLER Schema übernommen. Dieses Schema ist aus der Notfallmedizin zur schnellen Erhebung der Notfallanamnese bekannt [18]. Das Akronym steht für:

S Symptome
A Allergien
M Medikationen
P Medizinische Vorgeschichte
L Letzte Nahrungsaufnahme
E Dem Vorfall vorangegangene Ereignisse
R Risikofaktoren

Die Daten *Symptome*, *letzte Nahrungsaufnahme* und *dem Vorfall vorangegangene Ereignisse* wurden nicht berücksichtigt, da dies meist Ereignisse sind, die nur eine kurze Zeit zurückliegen und daher vom Patient voraussichtlich nicht auf dem NFC-Tag erfasst werden. Wie bereits in Abschnitt 5.1 beschrieben, ist der NFC-Tag nicht dazu gedacht, täglich aktualisiert zu werden.

In dem Datum *Vordiagnosen* wurde die *Medizinische Vorgeschichte* erfasst. Die *Operationen* wurden in der Umfrage explizit aufgeführt.

Die Daten *Medikationen*, *Allergien* und *Risikofaktoren* wurden unverändert übernommen.

Zusätzlich zu den Daten des SAMPLER Schemas wurden die folgenden Daten aufgenommen:

Der *Vorname* und *Nachname* des Patienten wurden aufgenommen, damit Helfer den Patienten persönlich ansprechen können. Auch zur Identifizierung des Patienten kann der Name sinnvoll sein.

Das *Geburtsdatum* gibt Informationen über Alter des Patienten. Es kann für bestimmte Krankheitsbilder wie den Herzinfarkt Hinweise geben.

Rhesusfaktor und *Blutgruppe* wurden mit der Idee aufgenommen, Kommunikationswege zu verkürzen. Das kann zum Beispiel die Anforderung einer bestimmten Blutkonserve an die Zielklinik sein, um im Schockraum der Notaufnahme keine Zeit zu verlieren.

Das Datum *Bestehende Schwangerschaft* ist wichtig, weil Schwangeren nicht alle Medikamente verabreicht werden dürfen und bei der Wahl und Durchführung der Notfalltherapie auch das Wohl des Kindes einbezogen werden muss.

Name der Krankenkasse und *Versichertennummer* stehen zur Auswahl, weil sie nach der Versorgung in der Notaufnahme zu Abrechnungszwecken verwendet werden könnten.

6.3 Durchführung

Die Umfrage wurde online mithilfe des Programms LimeSurvey durchgeführt.

Bei der Erstellung der Umfrage wurde auf eine verständliche Bedienbarkeit der Benutzeroberfläche geachtet, damit auch technisch unversierte Personen ohne Probleme an der Umfrage teilnehmen können.

6.3.1 Limitierungen

Während der Erstellung der Umfrage musste ein Kompromiss bei der Priorisierung der Notfalldaten eingegangen werden. Durch einen Bug in LimeSurvey konnte das Ranking nicht als Pflichtfeld eingestellt werden. Daher war es für Benutzer möglich die Frage zu überspringen, ohne ein Ranking erstellt zu haben.

6.3.2 Verteilung der Umfrage

Die Umfrage wurde einerseits an Personen verteilt, die beruflich im Rettungsdienst arbeiten, andererseits an Helfer in Hilfsorganisationen.

Zu den beruflichen Rettungsdienstlern gehörten: mehrere Notärzte, Rettungsassistenten sowie Rettungssanitäter. Zusätzlich wurde ein Link zu der Umfrage im Intranet der Johanniter Unfallhilfe Mainz platziert, an den Mailverteiler aller Rettungswachen des Malteser Hilfsdienstes in der Diözese Mainz und einen Ansprechpartner an der Pressestelle des Deutschen Roten Kreuzes zur weiteren Verteilung geschickt.

Außerdem wurde die Umfrage an mehrere Ortsgruppen des Deutschen Roten Kreuzes und des Malteser Hilfsdienstes verteilt.

Auch über Facebook wurde die Umfrage in mehrere Gruppen gestellt, die sich mit dem Thema Rettungsdienst beschäftigen.

6.4 Auswertung

6.4.1 Teilnehmer

Insgesamt haben 51 Personen an der Umfrage teilgenommen, allerdings konnten nach einer Vorverarbeitung nur die Daten von insgesamt 34 Teilnehmern ausgewertet werden. Die Vorverarbeitung schloss Teilnehmer aus, die nur eine Qualifikation angegeben, aber sonst keine weitere Frage beantwortet haben.

Tabelle 6.1 und Abbildung 6.1 zeigen die Verteilung nach Qualifikation der auswertbaren Teilnehmer.

Die Gruppe Sonstiges teilt sich auf in 1 Schüler, 1 First Responder und 1 ohne Angabe.

Tabelle 6.1: Aufschlüsselung der Teilnehmer nach Qualifikation

Qualifikation	**Absoluter Anteil**	**Relativer Anteil in Prozent(%)**
Sanitäter/Sanitätshelfer	8	23
Rettungshelfer	1	2
Rettungssanitäter	11	32
Rettungsassistent	3	8
Notfallsanitäter	1	2
Notarzt	7	20
Sonstiges	3	8

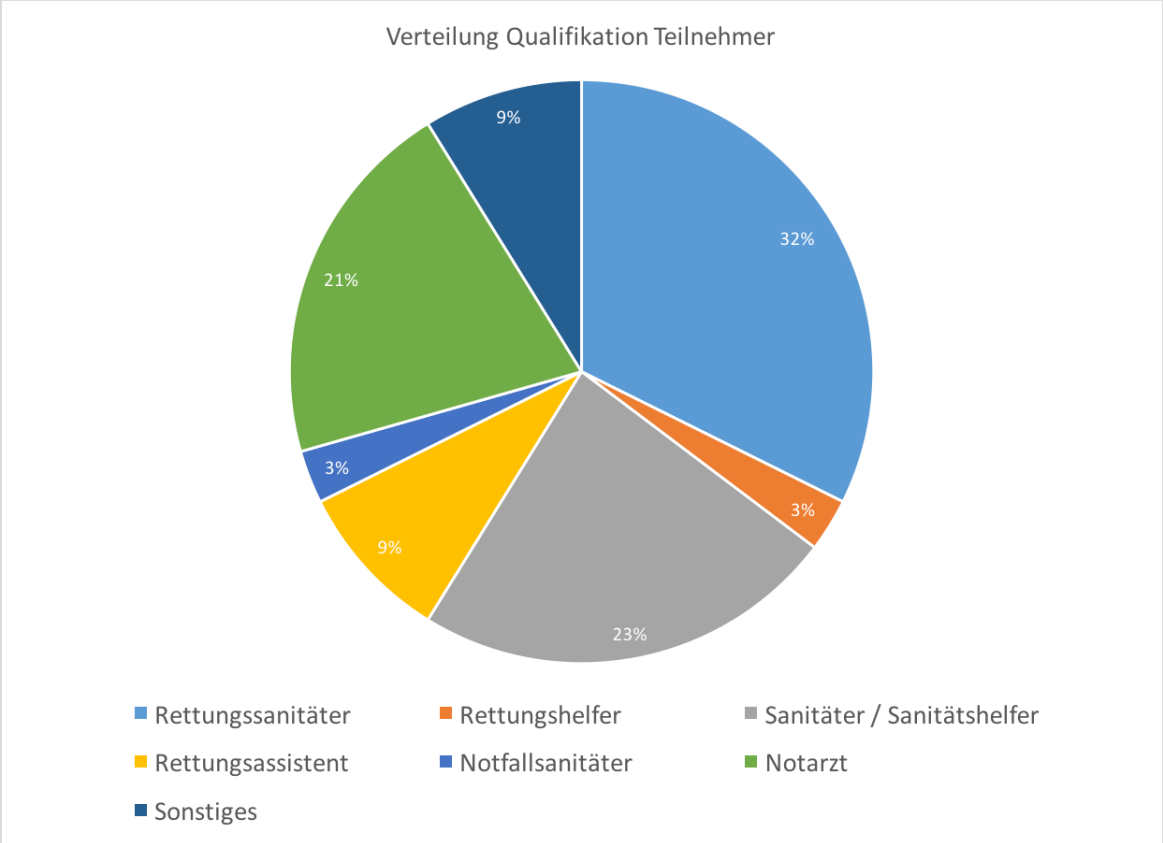


Abbildung 6.1: Verteilung der Qualifikationen der Teilnehmer

6.4.2 Notfalldaten

Tabelle 6.2 zeigt die Verteilung der ausgewählten Notfalldaten.

Tabelle 6.2: Verteilung der ausgewählten Notfalldaten

Datum	**Anzahl *Ja* Stimmen**	**Anzahl *Nein* Stimmen**
Vorname des Patienten	14	17
Nachname des Patienten	17	14
Geburtsdatum des Patienten	14	17
Arzneimittelallergien	25	6
Blutgruppe des Patienten	9	21
Rhesusfaktor des Patienten	5	25
Dauermedikationen	27	4
Neue Medikationen (einmalig, in den letzten Tagen)	21	10
Aktuell bestehende Erkrankungen	21	10
Bestehende Schwangerschaft	21	10
Vordiagnosen (bspw. Herzinfarkt 2013 oder Lungenembolie 2015)	27	4
Risikofaktoren (Alkohol, Raucher, Bewegungsmangel, Drogenkonsum)	10	20
Notfallkontakte (Name und Telefonnummer)	15	15
Name der Krankenkasse	3	27
Versichertennummer des Patienten	4	26
Operationen (was, wann)	17	14

Die Anzahl der *Ja* und *Nein* Stimmen legt die Auswahl der folgenden Daten nahe, um diese auf dem NFC-Tag zu speichern:

- Dauermedikationen
- Arzneimittelallergien
- Vordiagnosen
- Vorname des Patienten
- Nachname des Patienten
- Bestehende Schwangerschaft

Die Notfallkontakte haben eine gleiche Anzahl an *Ja* und *Nein* Stimmen. Sie wurden aber trotzdem hinzugenommen, denn sie bieten eine sehr gute Informationsquelle für Helfer, und sei es nur, um Angehörige über einen Unfall zu benachrichtigen.

Folgende Daten wurden trotz hoher Anzahl an *Ja* Stimmen weggelassen:

- Operationen (was, wann)

  Operationen können in den Vordiagnosen kodiert werden.

- Aktuell bestehende Erkrankungen

  Es ist nicht vorgesehen, dass der NFC-Tag in kurzen Zeitintervallen aktualisiert wird.

- Neue Medikationen (einmalig, in den letzten Tagen)

  Auch hier ist eine tagesaktuelle Abbildung nicht anzunehmen.

Die Risikofaktoren sollen laut Verteilung nicht aufgenommen werden, können aber in den Vordiagnosen gespeichert werden.

6.4.3 Priorisierung der Notfalldaten

Die Umfrage hat auch nach der Priorisierung der Notfalldaten gefragt. Insgesamt haben 41% der Teilnehmer, davon 41% Notärzte, kein Ranking erstellt. Diesen Stimmen fehlen aufgrund des in Abschnitt 6.3.1 angesprochenen Bugs in LimeSurvey.

Tabelle 6.3 zeigt die TOP 10 unter allen Qualifikationen.

Tabelle 6.3: TOP 10 unter allen Qualifikationen

Platz	**Datum**	**Stimmen**
1	Aktuell bestehende Erkrankungen	6
2	Neue Medikationen (einmalig, in den letzten Tagen)	5
3	Dauermedikationen	5
4	Arzneimittelallergien	4
5	Bestehende Schwangerschaft	4
6	Notfallkontakte (Name und Telefonnummer)	2
7	Risikofaktoren	1
8	Nachname des Patienten	1
9	Vorname des Patienten	1
10	Geburtsdatum	1

Die Priorisierung kann als Grundlage für einen Eingabeassistenten dienen. Der Eingabeassistent soll aufgrund des zur Verfügung stehenden Speicherplatzes auf dem NFC-Tag die Notfalldaten auswählen, die darauf gespeichert werden sollen. Auf Grundlage der Reihenfolgen aus Tabelle 6.3 könnte der Eingabeassistent seine Entscheidungen treffen. Allerdings ergeben sich die folgenden Probleme bei einer solchen Implementierung:

1) Ein Patient könnte nicht alle Daten angeben wollen

2) In einem Notfall könnten Daten gebraucht werden, die aufgrund fehlender Speicherkapazität nicht auf dem NFC-Tag gespeichert sind

Problem 1) liegt in der informationellen Selbstbestimmung des Benutzers begründet. Jeder Mensch darf selbst entscheiden, welche personenbezogenen Daten er preisgeben will und wie sie verwendet werden sollen.

Hier kann die App ihn mit Hilfetexten auf die Wichtigkeit der einzelnen Notfalldaten hinweisen.

Problem 2) ist nicht lösbar, denn es ist nicht vorhersehbar, welcher Notfall eintreten wird. Die Entscheidung, welche Daten gespeichert werden, muss daher beim Benutzer liegen. Er kennt zukünftige Notfälle zwar auch nicht, kann aber seine eigenen Daten priorisieren. Zur Abhilfe dieses Problems kann der Benutzer einen NFC-Tag mit großer Speicherkapazität benutzen. Gleichzeitig speichert die App die Daten durch die in Kapitel 8.3 beschriebenen Verfahren platzsparend ab.

Aus den genannten Problemen kann die Priorisierung nicht, wie vorerst angenommen, als Grundlage für einen Eingabeassistenten dienen.

Daher wird hier auf eine weitere Auswertung der Priorisierung nach Qualifikation verzichtet.

7 Entwurf

In diesem Kapitel wird der Entwurf der mobilen Anwendung, des serverseitigen Backends und der Gesamtarchitektur beschrieben. Dabei werden die Anforderungen aus Kapitel 5 *Anforderungsanalyse* einbezogen.

7.1 Systemarchitektur

Das System besteht aus einem NFC-Tag, der im Rahmen dieser Arbeit entwickelten Android App und dem serverseitigen Backend (Abb. 7.1).

Die App schreibt die eingegebenen Notfalldaten auf den NFC-Tag, liest sie aus und zeigt sie an.

Das serverseitige Backend besteht aus einer REST API und einer Datenbank. Es stellt Datenaktualisierungen für die Vordiagnosen und Medikationen bereit. Ferner kann es als Fallback Lösung dienen, falls auf dem NFC-Tag gespeicherte Daten nicht auf dem auslesenden Smartphone verfügbar sind. Dies ist jedoch nur ein Fallback und nicht die reguläre Verwendung.

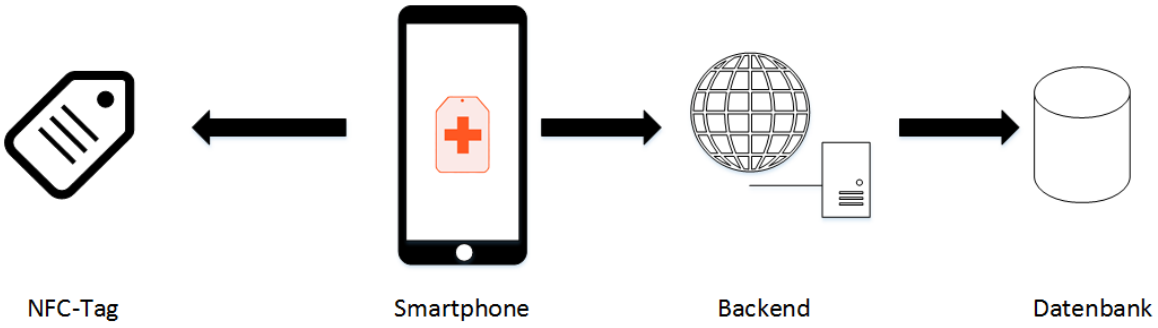


Abbildung 7.1: Die Verteilung von NFC-Tag, Smartphone und Backend Server.

7.2 Datenmodell

Das Datenmodell bildet die in Kapitel 6 *Auswahl der Notfalldaten* ausgewählten Notfalldaten ab.

Eine Medikation wird durch `Medication`, eine Vordiagnose durch `Diagnose` und ein Notfallkontakt durch `EmergencyContact` beschrieben (s. Abb. 7.2). `Metadata` stellt die Patientendaten mit Vorname, Nachname und Geburtsdatum dar.

Eine `Diagnose` hat die Felder `allergy` und `intolerance`, mit denen Allergien und Intoleranzen abgebildet werden können.

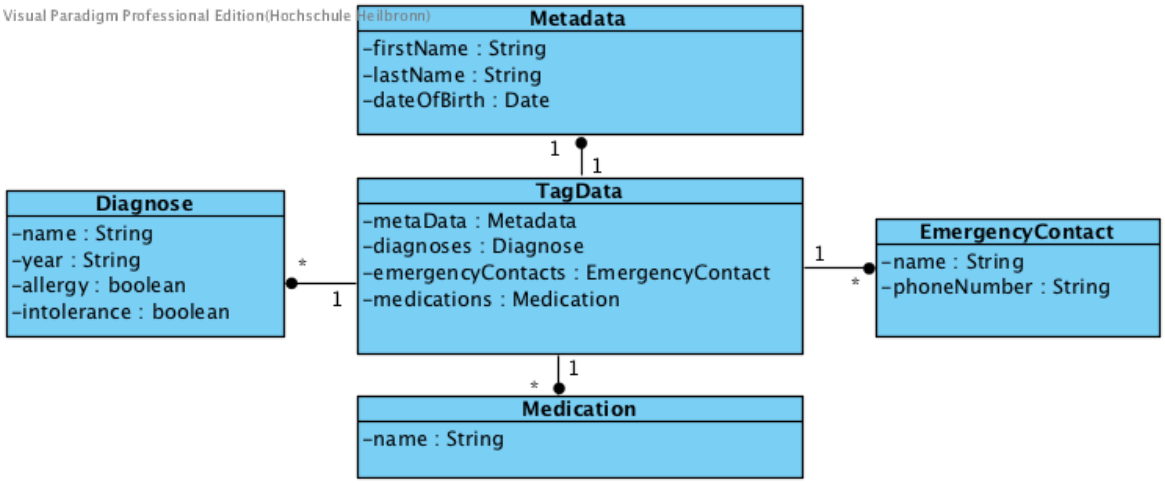


Abbildung 7.2: Datenmodell der Notfalldaten

7.3 API

Die API stellt die Interfaces für die Domäneklassen und den Datenbankservices sowie Exceptions, eine NFC-Contract Klasse und eine abstrakte Klasse zum Parsen der Daten bereit. Außerdem gibt es eine Klasse `SimpleTextFormat`, die ein `TagData`-Objekt in eine textuelle Repräsentation bringt.

Das in Abschnitt 7.2 beschriebene Datenmodell wird durch die Interfaces `Medication`, `Diagnose` und `EmergencyContact` abgebildet. Zusätzlich gibt es das Interface `TagData`, das Methoden zum Zugriff auf die anderen Domäneklassen bereitstellt (s. Abb. 7.2).

Die Datenbankkomponente ist als SOA entworfen. Dabei definiert das Interface `DbService` alle Methoden, um `Medications` und `Diagnoses` valide zu instanziieren, persistieren und aus der Datenbank zu lesen. Dabei müssen alle persistierbaren Objekte von der abstrakten Klasse `AbstractPersistedObject` erben.

Das Interface `NFCContract` definiert Konstanten, die für das Lesen und Schreiben der Daten auf den NFC-Tag gebraucht werden, wie zum Beispiel in welcher Reihenfolge die Daten gespeichert sind oder durch welche Zeichen sie separiert werden.

Die Daten auf dem NFC-Tag sind in einem eigens dafür entwickelten Speicherformat gespeichert. Das Format erlaubt es, die Daten platzsparend zu speichern. Um die Daten auslesen zu können, muss die Position der `NdefRecords`, die die Daten enthalten, auf dem NFC-Tag berechnet werden. Die abstrakte Klasse `DataParser` stellt die dafür notwendigen (Hilfs-) Methoden zur Verfügung. Die Spezifikation des Datenformats wird in Kapitel 8.3 *Effiziente Datenspeicherung* im Detail beschrieben.

7.4 App

Die App implementiert u.a. das beschriebene Datenmodell und viele Interfaces der API wie zum Beispiel die Interfaces `DbService` oder `DataParser`. Außerdem besteht die App noch aus weiteren Teilen wie den Android Activities und der NFC Zugriffsschicht. Alle diese Teile werden im Folgenden beschrieben.

7.4.1 Android Activities

Tabelle 7.1 gibt eine Übersicht über die einzelnen Activities der App und ihre Funktion.

Tabelle 7.1: Die Activity-Klassen der mobilen Anwendung

Klasse	**Funktion**
EditDiagnoseActivity	Hinzufügen/Ändern einer Vordiagnose
EditEmergencyContactActivity	Hinzufügen/Ändern eines Notfallkontaktes
EditMedicationActivity	Hinzufügen/Ändern einer Medikation
EditMetadataActivity	Hinzufügen/Ändern der Patientendaten
MainActivity	Der Hauptbildschirm
QRActivity	Anzeige des QR-Codes
ReadActivity	Die Leseansicht
SettingsActivity	Die Einstellungen
UpdateActivity	Zuständig für das Steuern der Updates
WizardActivity	Der Eingabeassistent

7.4.2 Datenbank

Der Datenbankservice implementiert das Interface `DbService` aus der API wie in Abschnitt 7.3 beschrieben. Mit der Factoryklasse `DbServiceFactory` kann eine Instanz des Datenbankservices erzeugt werden. Die Factoryklasse implementiert dabei ein Singleton für den Datenbankservice.

Die Klasse `NextStorageContract` beinhaltet Konstanten zur Angabe der Datenbankversion und des Datenbanknamens. Ferner enthält sie noch die inneren Klassen `Diagnose` und `Medication`, die Konstanten zur Angabe des Tabellenschemas und der einzelnen Spaltennamen definieren.

7.4.3 Parser

Der Datenparser erfüllt zwei Aufgaben: Zum einen konvertiert er ein `TagData` Objekt in eine Liste von `NdefRecord`s. Diese `NdefRecord`s können dann auf einen NFC-Tag geschrieben werden. Zum anderen bietet er Methoden, um aus einer Liste von `NdefRecord`s ein TagData Objekt zu erstellen. Das ist das eigentliche Parsen der Notfalldaten vom NFC-Tag.

Die abstrakte Klasse `AbstractDataParser` implementiert die Klasse `DataParser` aus der API. Sie definiert zwei abstrakte Methoden `parseConcrete` und `parseInverseConcrete`, die statt zweidimensionale Byte Arrays ein Array von `NdefRecord`s benutzen.

Der Parser implementiert das Filter Pattern. Es gibt einen Parser `DataParser`, dem weitere Parser hinzugefügt werden können. Diese werden in der Reihenfolge wie sie hinzugefügt werden durchlaufen. So gibt es die Parser `MedicationParser` für die Medikationen und `DiagnoseParser` für die Vordiagnosen. Alle Parser erben von der Klasse `AbstractDataParser`.

7.4.4 NFC

Die Lese- und Schreibefunktion ist in die Interfaces `NfcReader` und `NfcWriter` gekapselt. Die Interfaces werden durch die Klassen `NfcReaderImpl` zum Auslesen des NFC-Tags und `NfcWriterImpl` zum Beschreiben eines NFC-Tags implementiert.

Um Instanzen der beiden Klassen zu erzeugen, gibt es die Factory Klasse `NfcFactory` mit den beiden Methoden `newNfcReader()` und `newNfcWriter()`.

Zusätzlich gibt es die Klassen `ReadNfcTask`, `WriteNfcTask` und `GetDataSizeTask`, die von der Android Klasse `AsyncTask`s erben. Sie stellen einzelne Aufgaben dar und laufen in einem eigenen Thread, um die UI nicht zu blockieren. `ReadNfcTask` und `WriteNfcTask` sind für das Lesen und Schreiben von und auf den NFC-Tag zuständig. Die Klasse `GetDataSizeTask` berechnet die Speichergröße der aktuell eingegebenen Daten, wenn sie so auf den NFC-Tag geschrieben würden.

7.4.5 In App Kommunikation

Die Kommunikation innerhalb der App zum Aktualisieren geänderter Daten ist mit Broadcasts realisiert.

Broadcasts funktionieren ähnlich wie das bekannte Observer Design Pattern der Gang of Four (GoF), mit der Ausnahme, dass es kein konkretes `Observable` gibt. Objekte können mithilfe von `BroadcastReceiver` Systemevents oder Events anderer Applikationen abonnieren. Wenn ein anderes Objekt Daten zu diesem Event verschickt, werden diese den `BroadcastReceivern` zugestellt.

Abbildung 7.3 zeigt die Kommunikationswege schematisch am Beispiel des Hinzufügens einer Vordiagnose.

Das `DiagnoseFragment` zeigt die `EditDiagnoseActivity` an und erhält ein erzeugtes `Diagnose` Objekt. Dieses wird in ein `Intent` gekapselt und über den `LocalBroadcastManager` verschickt. Der `ModifiedTagDataBroadcastReceiver` in der `MainActivity` empfängt das Broadcast und fügt dem dortigen `TagData` Objekt das `Diagnose` Objekt hinzu.

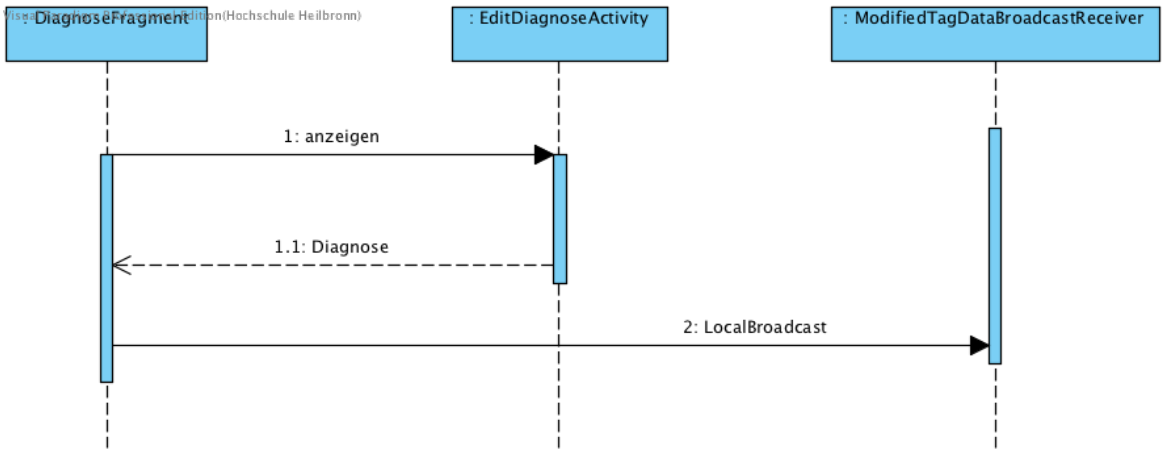


Abbildung 7.3: Schematische Darstellung der In App Kommunikation

Die LocalBroadcasts werden abonniert, wenn die Activity oder das Fragment erstellt wird. Wenn eine Activity oder ein Fragment zerstört wird, werden die LocalBroadcasts unregistriert.

7.4.6 QR-Code

Außerdem kann die App zur Herstellung von Interoperabilität einen QR-Code anzeigen, der die Notfalldaten enthält. Das Datenformat ist mit dem beschriebenen Datenformat aus Abschnitt 8.3 identisch. Da in einem QR-Code keine `NdefRecords` zur Verfügung stehen, werden die einzelnen Datenfelder mit dem Pipe-Symbol getrennt.

7.5 Backend - DbRuntime

Die DbRuntime ergibt sich aus der Anforderung, dass die vordefinierten Medikationen und Vordiagnosen aktualisierbar sein sollen.

Die DbRuntime bildet die Datenzugriffsschicht des Backends. Sie ist dafür zuständig, die Daten aus der Datenquelle an den Webservice zu liefern. Für den Datenbankzugriff wird die Java Persistence API verwendet.

Das Backend implementiert die in Abschnitt 7.3 beschriebenen Interfaces der Domäneklassen. Das Interface `DbService` aus der API wird nicht implementiert, da dort die Version, zu der die Datensätze gehören, nicht berücksichtigt wird. In der App muss sie nicht berücksichtigt werden, denn es werden immer alle Daten angezeigt. Im Backend spielt die Versionsnummer eine zentrale Rolle zur Verteilung der Daten bei einer Aktualisierung. Aus diesen beiden Gründen wird ein neues Interfaces `BackendDbService` implementiert, das die Versionsnummer der Daten berücksichtigt.

Das Backend ist als SOA entworfen mit der Klassen `DbServiceImpl` als Implementierung des Services.

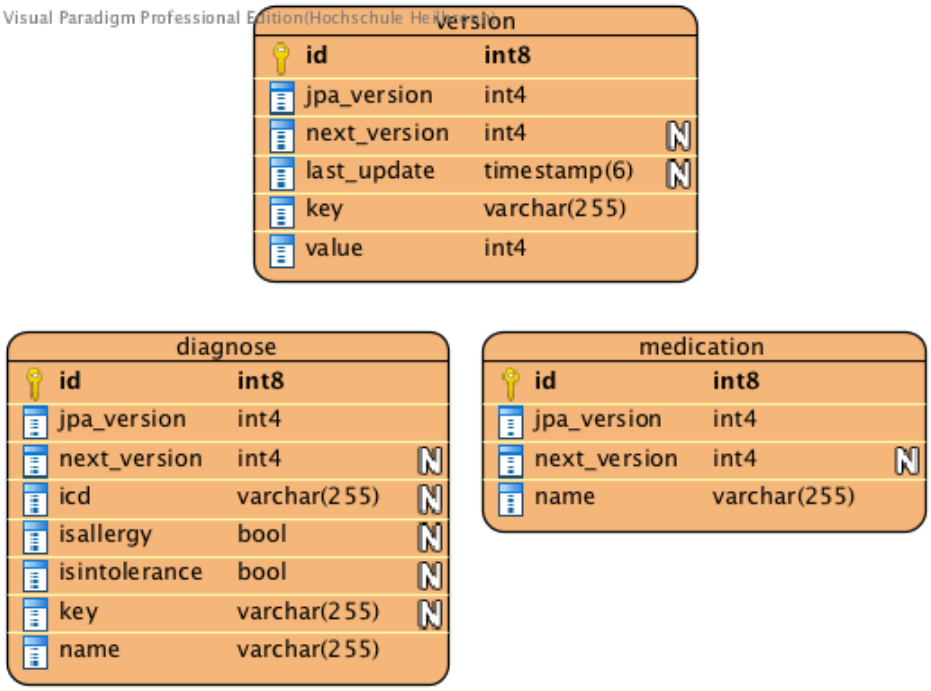


Abbildung 7.4: Datenbankschema des Backends

Abbildung 7.4 zeigt das Datenbankschema des Backends. In der Tabelle *Diagnose* sind die Vordiagnosen und in der Tabelle *Medication* die Medikationen gespeichert. Zusätzlich gibt es eine Tabelle *Version*, in der die aktuelle Versionen der Vordiagnosen und Medikationen gespeichert sind. Jede Tabelle hat eine Spalte *next_ version*, die die Version der einzelnen Tupel angibt.

7.6 Backend - Webservice

Die Webservice ergibt sich ebenfalls aus der Anforderung, dass die vordefinierten Medikationen und Vordiagnosen aktualisierbar sein sollen. Darüber sind die Daten für die

Updatefunktion und die Fallback Option erreichbar. Die angefragten Daten werden im JSON Format ausgegeben. Er ist als Restful API entworfen.

Der Webservice gliedert sich in XTO Klassen, die Webservice API und deren Implementierung.

Die Implementierungen haben Endpunkte, welche die aktuelle Version (*/version*), alle Daten einer Version (*/get*) oder ein bestimmtes Datum (*/getById*) anhand der ID des Datums zurückgeben. Tabelle 7.2 zeigt die einzelnen Endpunkte der REST API. Der GET-Parameter *:id* entspricht der ID des Datums, das zurückgegeben werden soll. Der GET-Parameter *:version* gibt die Version an, zu der alle Daten ausgegeben werden sollen. Der Versionierung der API wird über einen Parameter /v1 in der URI genügt.

Tabelle 7.2: Endpunkte der REST API

HTTP	**URI**
GET	/next-ws/service/v1/diagnose/get?id=:version
GET	/next-ws/service/v1/diagnose/getById?id=:id
GET	/next-ws/service/v1/diagnose/version
GET	/next-ws/service/v1/medication/get?id=:version
GET	/next-ws/service/v1/medication/getById?id=:id
GET	/next-ws/service/v1/medication/version

XTO steht für Cross Transfer Object (auch bekannt als *DTO, Data Transfer Object*). In ihnen werden die Daten, die an andere Anwendungen geschickt werden, gekapselt. Medikationen, Diagnosen und Versionen werden im Backend durch die JPA Entities `MedicationImpl`, `DiagnoseImpl` und `VersionImpl` repräsentiert. Für diese Entities gibt es entsprechende XTO Klassen. Die XTO Klassen werden dann als JSON Objekte ausgeliefert.

7.6.1 Fallback Lösung

Wie bereits in Abschnitt Systemarchitektur dieses Kapitels angesprochen, dient der Webservice auch als Fallback Lösung für das Parsen der Daten. Sie kommt zum Tragen, wenn ein Benutzer einen nicht aktuellen Datenstand in seiner App hat und die Daten, die auf den NFC-Tag geschrieben wurden, nur in einer neueren Version verfügbar sind. Die App kann dann den Webservice nach der textuellen Repräsentation fragen. Dazu können die Ressourcen `medication/getById?id=:id` und `diagnose/getById?id=:id` genutzt werden.

7.7 Clientseitige Updateverarbeitung

Es gibt verschiedene Ansätze, um das Aktualisierungsverhalten in der App zu realisieren:

(1) Alte Daten löschen, neue Daten hinzufügen

(2) Daten hinzufügen, keine Daten löschen

(3) Alte Daten vorläufig löschen, neue Daten hinzufügen

> Daten werden vorläufig gelöscht und können nicht mehr ausgewählt werden. Neue Daten werden hinzugefügt.

Das Aktualisieren der vordefinierten Medikationen und Vordiagnosen ist problematisch. Es dürfen keine Daten aus der App gelöscht werden, da sonst vordefinierte Notfalldaten, die auf dem NFC-Tag gespeichert sind, nicht angezeigt werden können.

Daher wird die beschriebene Lösung *(2) Daten hinzufügen, keine Daten löschen* zur Implementierung gewählt.

8 Implementierung

Nachdem die Anforderungen in Kapitel 5 und der Entwurf in Kapitel 7 festgelegt worden sind, wird in diesem Kapitel die Implementierung der App und des Backends beschrieben.

Die Grafik rechts ist das App Icon. Es wurde von Johanna Schirmer gestaltet.

8.1 Entwicklungswerkzeuge und Bibliotheken

Tabelle 8.1 listet alle verwendeten Entwicklungswerkzeuge und Bibliotheken der App inklusive ihrer Versionsnummer auf. Zur Entwicklung der App und API wurde das Build Tool Gradle benutzt. Es wird von Google offiziell als Build Tool für Android vorgeschlagen [19].

Das Backend wurde mithilfe des Build Tools Maven entwickelt. Dazu wurde eine Maven Vorlage des PITA Praktikums verwendet. Da diese Vorlage zur Verfügung stand und beispielsweise das Bereitstellen des Applikationsserver stark vereinfachte, wurde für das Backend Maven verwendet.

8.2 Auswahl Betriebssystem

Die Entscheidung, eine Android App zu entwickeln, wurde aufgrund der Marktverteilung von Android getroffen. So hat Android in Deutschland im Oktober 2015 einen Marktanteil von über 70 Prozent. Windows Phone liegt bei 7.2 Prozent und iOS bei 19.8 Prozent [20].

Außerdem soll die App nur mit bis zur Android Version 4.4 abwärtskompatibel sein, da Android 4.4 im Februar 2015 mit über 35% die größte Marktverteilung unter den Android Versionen hat [21].

8.3 Effiziente Datenspeicherung

Nachdem in Kapitel 6 die Notfalldaten durch eine Umfrage ausgewählt wurden, werden hier verschiedene Möglichkeiten der Speicherung diskutiert sowie die Spezifikation gezeigt.

8.3.1 Möglichkeiten

Datenformat

Datenformate erster Wahl zur Herstellung von Interoperabilität sind XML und JSON. Da beide Formate viele Zeichen enthalten, die nicht zur Semantik des gespeicherten Wertes beitragen und somit mehr Speicherplatz als nötig verbrauchen, wurde ein eigenes Datenformat entwickelt. Dieses wird im Folgenden beschrieben.

Tabelle 8.1: Genutzte Entwicklungswerkzeuge und Bibliotheken

Komponente	**Version**
IntelliJ IDEA	15.0.3
Gradle	2.2.1
Java	1.7
Android Build Tools	23.0.1
Maven	3
Android Design Support Library[1]	23.1.1
Android Async Http Client[2]	3
Maven	3
Iconify[3]	2.1.0
AppIntro[4]	3.4.0
SQLiteAsset Helper[5]	2.0.1
QRGen[6]	2.0.1
WizarDroid [7]	1.3.1
Android ViewPagerIndicator[8]	2.4.1
TagInfo by NXP Semiconductors[9]	4.11.59
TomEE Plus	1.3.2
slf4j	1.7.10
log4j	1.2.17
Hibernate	4.3.11.Final
Hibernate Entitymanager	4.3.11.Final
Hibernate C3PO	4.3.11.Final
PostgreSQL	9.4-1201-jdbc41
C3PO	0.9.5.1
Apache Commons Lang	2.6

[1] `https://developer.android.com/tools/support-library/index.html`. Letzter Zugriff: 17. Februar 2016

[2] `https://github.com/AsyncHttpClient/async-http-client`. Letzter Zugriff: 17. Februar 2016

[3] `https://github.com/JoanZapata/android-iconify`. Letzter Zugriff: 17. Februar 2016

[4] `https://github.com/PaoloRotolo/AppIntro`. Letzter Zugriff: 17. Februar 2016

[5] `https://github.com/jgilfelt/android-sqlite-asset-helper`. Letzter Zugriff: 17. Februar 2016

[6] `https://github.com/kenglxn/QRGen`. Letzter Zugriff: 17. Februar 2016

[7] `https://github.com/Nimrodda/WizarDroid`. Letzter Zugriff: 17. Februar 2016

[8] `https://github.com/JakeWharton/ViewPagerIndicator`. Letzter Zugriff: 17. Februar 2016

[9] `https://play.google.com/store/apps/details?id=com.nxp.taginfolite`. Letzter Zugriff: 17. Februar 2016

Daten komprimiert speichern

Das Datenformat nutzt aus, dass in der App *Vordiagnosen* und *Medikationen* in einer Datenbank gespeichert sind. Der Benutzer kann aus diesen Daten seine Vordiagnosen und Medikationen wählen oder sie selbst eintragen. Der Surrogate (ID), der den Primärschlüssel in der Datenbank darstellt und vom Datentyp Long ist, hat in der Regel eine kleinere Zeichenzahl als die ausgeschriebene Repräsentation des Datums. Wenn nur der Surrogate auf dem NFC-Tag gespeichert wird, kann so Speicherplatz eingespart werden. Auf diese Weise kann mit allen Daten verfahren werden, die vordefiniert in der App zur Auswahl stehen.

Diese Art der Speicherung impliziert neue Herausforderungen:

(1) Die vordefinierten Daten müssen vollständig sein.

Damit viele vom Benutzer eingegebene Medikationen und Vordiagnosen wie beschrieben abgespeichert werden können, müssen sie vollständig sein, damit er aus ihnen wählen kann.

(2) Die vordefinierten Daten müssen aktualisierbar sein.

Es können neue Krankheiten entdeckt werden. Daher müssen die Daten aktualisierbar sein, um alte Diagnosen ersetzen oder neue Diagnosen hinzuzufügen zu können.

(3) Alle App-Installationen müssen den gleichen Datenstand haben.

Damit jede App-Installation jeden NFC-Tag auslesen kann, müssen alle Installationen die gleichen Vordiagnosen und Medikationen in ihrer Datenbank gespeichert haben.

Problem (1): Vollständigkeit der vordefinierten Daten

Dieses Problem lässt sich nicht vollständig lösen. Allerdings kann sich daran angenähert werden, indem bei den Medikationen alle Medikamente der Roten Liste, Grünen Liste und Gelben Liste zur Auswahl angeboten werden. Für die Entwicklung wurden Testdaten der Roten Liste verwendet. Die Rote Liste, Grüne Liste und Gelbe Liste sind Arzneimittelverzeichnisse in Deutschland. Allerdings sind die Pharmakonzerne nicht verpflichtet, ihre Medikamente dort anzumelden.

Bei den Vordiagnosen ist es schwieriger. Es können zwar die Beschreibungen des ICD-10-GM Code gespeichert werden, diese enthalten allerdings medizinische Termina und sind daher für medizinische Laien oft nicht verständlich. Für dieses Teilproblem ist noch keine Lösung gefunden.

Problem (2): Aktualisierbarkeit der vordefinierten Daten

Das Problem lässt sich durch eine Update Funktion in der mobilen Anwendung beheben.

Problem (3): Gleiche Datenstände in allen App-Installationen

Dieses Problem lässt sich nicht lösen. Der Benutzer kann auf bereitstehende Updates aufmerksam gemacht, aber nicht zum Update gezwungen werden. Durch den Updatezwang würde die App unbenutzbar werden. Da der Benutzer aber mit der App jederzeit einen NFC-Tag auslesen können muss, muss die App jederzeit benutzbar sein. Daher kann der Benutzer nicht zu einem Update gezwungen werden.

Eine mögliche Abhilfe besteht darin, dem Benutzer eine Warnung anzuzeigen, wenn er einen NFC-Tag liest, auf dem bei ihm nicht hinterlegte Daten gespeichert sind. Dadurch wird allerdings die Information des gespeicherten Datums nicht wiederhergestellt. Eine andere Möglichkeit ist, dass bei Internetverbindung die textuelle Repräsentation der gespeicherten ID beim Backend angefragt wird. Ist keine Internetverbindung vorhanden, ist die Information des gespeicherten Datums verloren.

8.3.2 Storage Byte

Das Storage Byte zeigt an, welche Daten auf dem NFC-Tag gespeichert wurden. Es ist auf dem zweiten NdefRecord auf dem NFC-Tag gespeichert. Tabelle 8.2 zeigt den Aufbau des Storage Bytes.

Tabelle 8.2: Aufbau des Storage Bytes

Bitstelle	**Beschreibung**
8	Nachname
7	Vorname
6	Geburtsdatum
5	Vordiagnosen
4	Medikationen
3	Notfallkontakte
2	nicht besetzt
1	nicht besetzt

Die Bits 2 und 1 sind nicht besetzt.

Ist ein Bit auf 1 gesetzt, ist dieses Datum auf dem Tag gespeichert.
Ist ein Bit auf 0 gesetzt, ist dieses Datum auf dem Tag *nicht* gespeichert.

In Listing 8.1 wird gezeigt, wie duch Bit Shifting getestet werden kann, ob ein Datum gespeichert ist oder nicht. Eine Variable mit Wert *true* zeigt, dass das entsprechende Datum gespeichert ist.

Listing 8.1: Bit Shifting auf dem Storage Byte, um zu testen, welche Daten gespeichert sind

```
1 boolean lastName = ((1 << NfcContract.STORAGE_POSITION_LAST_NAME)
      & storageByte) != 0;
2 boolean firstName = ((1 << NfcContract.STORAGE_POSITION_FIRST_NAME
      ) & storageByte) != 0;
3 boolean dateOfBirth = ((1 << NfcContract.
      STORAGE_POSITION_DATE_OF_BIRTH) & storageByte) != 0;
4 boolean drugs = ((1 << NfcContract.STORAGE_POSITION_MEDICATIONS) &
       storageByte) != 0;
5 boolean diseases = ((1 << NfcContract.STORAGE_POSITION_DIAGNOSES)
      & storageByte) != 0;
6 boolean emergencycontacts = ((1 << NfcContract.
      STORAGE_POSITION_EMERGENCY_CONTACTS) & storageByte) != 0;
```

8.3.3 Datenfelder

Tabelle 8.3 stellt dar, welche Datenfelder in welcher Konfiguration enthalten sein dürfen.

Tabelle 8.3: Aufbau und Kardinalitäten der speicherbaren Daten

Kardinalität	Datum	Datentyp	Format
0..1	Nachname	String	
0..1	Vorname	String	
0..1	Geburtsdatum	String	ddMMyy
0..*	Vordiagnosen[1, 2, 3]	String oder Long	
0..*	Medikationen[1, 2]	String oder Long	
0..*	Notfallkontakte[1, 4]	String	Name mit vorangestelltem #

[1] Enumeration. Endet mit einem NdefRecord, der nur ein *;* beinhaltet.

[2] Die Benutzereingabe oder ein Long, der in der Datenbank oder im Webservice nachgeschlagen werden kann.

[3] Ist die Vordiagnose benutzerdefiniert, folgen zwei weitere NdefRecords mit einer 1 oder 0 als Inhalt. Der erste NdefRecord gibt an, ob die Vordiagnose eine Medikamentenallergie ist. Der zweite NdefRecord, ob die Vordiagnose eine Medikamentenunverträglichkeit ist. Nach diesen beiden Records kann ein weiterer optionaler Record mit einer Jahreszahl als String folgen.

[4] Nur die Telefonnummer ist ein Pflichtfeld. Der Name ist optional und hat ein # als Präfix.

Beispieldatensatz

Tabelle 8.4 zeigt einen Beispieldatensatz auf dem NFC-Tag, der nach der beschriebenen Spezifikation des Speicherformats aufgebaut ist.

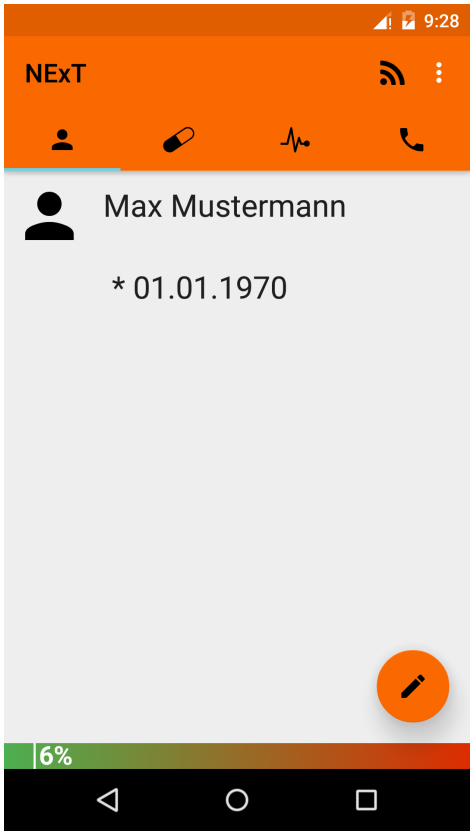
Tabelle 8.4: Aufbau eines Beispieldatensatzes ohne Verschlüsselung

Index	Wert	Was
1	de.akeil.next	Android Application Record
2	252	Storage Byte
3	Max	Vorname
4	Mustermann	Nachname
5	010170	Geburtsdatum
6	294	Medikament
7	;	Separator
8	Herzinfarkt	Vordiagnose
9	0	Vordiagnose ist Allergie
10	0	Vordiagnose ist Intoleranz
11	2010	Jahr Vordiagnose
12	;	Separator
13	#Erika Mustermann	Notfallkontakt Name
14	0123456789	Notfallkontakt Telefonnummer
15	;	Separator

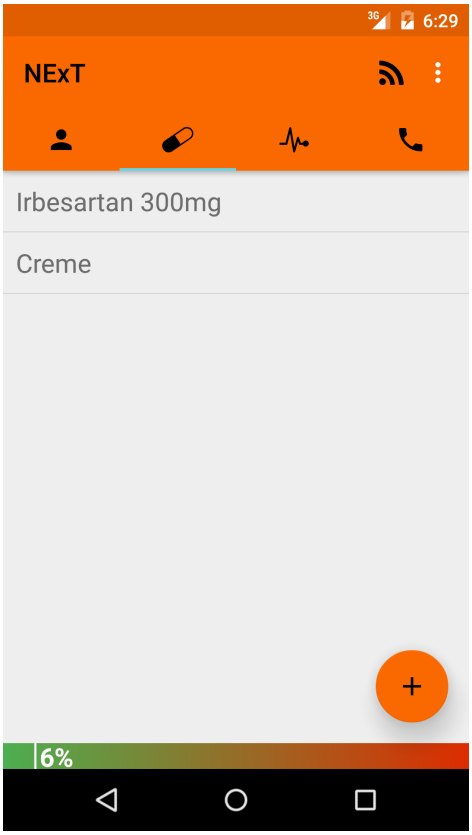
8.4 Schreibansicht

Abbildung 8.1 zeigt das Widget FuelStateView. Es wurde im Rahmen dieser Arbeit entwickelt und zeigt dem Benutzer den Füllgrad der momentan angezeigten oder eingegebenen Daten in Prozent an, wenn ein NFC-Tag vorher gescannt wurde. Je weiter die weiße Markierung im roten Bereich des Farbverlaufs ist, desto mehr Speicherplatz ist verbraucht.

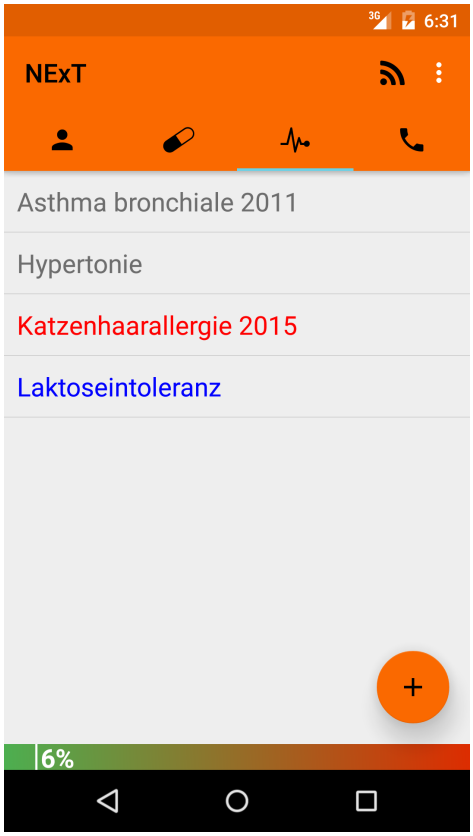
Abbildung 8.1: Widget FuelStateView



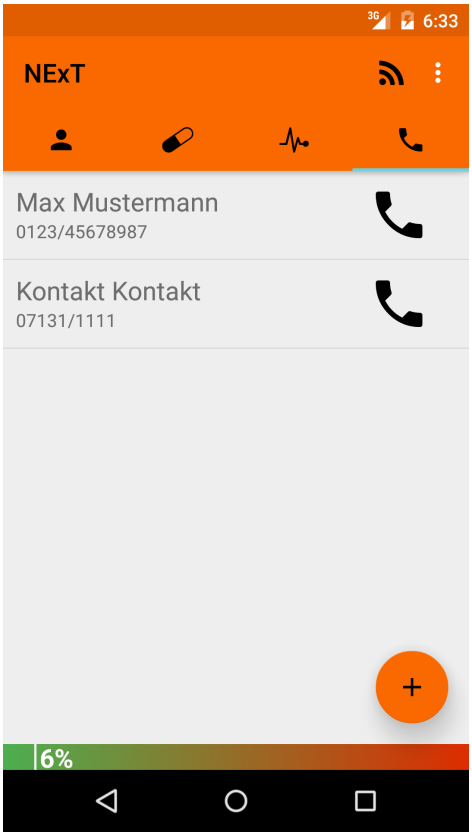
(a) Ansicht der Patientendaten



(b) Ansicht der Medikationen



(c) Ansicht der Vordiagnosen



(d) Ansicht der Notfallkontakte

Abbildung 8.2: Die verschiedenen Schreibansichten

Das FuelStateView wird aktualisiert, wenn der Benutzer seine Notfalldaten eingibt. Zu den nötigen Eingabemasken, welche das Ändern der Patientendaten oder Hinzufügen von Listeneinträgen ermöglichen, gelangt der Benutzer über den FloatingActionButton unten rechts auf den einzelnen Ansichten. Je nach ausgewähltem Tab gelangt er in die Ansicht in Abbildung 8.3a, 8.3b, 8.3c oder 8.3d.

Abbildung 8.2 zeigt die verschiedenen Schreibansichten. Über das Icon links neben dem Overflow Icon/Menü in der ActionBar wird ein Intent aktiviert, wodurch die Daten auf einen NFC-Tag geschrieben werden können. Das Optionenmenü wird durch das Overflow Icon rechts in der ActionBar erreicht. Von hier aus können die Einstellungen erreicht, der Eingabeassistent gestartet und der Lesemodus aktiviert werden.

Die einzelnen Ansichten sind in einem TabView verbunden. Von links nach rechts sind das die Tabs: Patientendaten, Medikationen, Vordiagnosen und Notfallkontakte.

Die Patientendatenansicht zeigt Vor- und Nachname sowie das Geburtsdatum des Patienten an. Die einzelnen Daten werden dabei mit einem Icon links daneben symbolisiert (Abb. 8.3a).

Die Vordiagnosen werden als Liste angezeigt (Abb. 8.2b). Als Allergie markierte Vordiagnosen werden rot, Unverträglichkeiten blau hervorgehoben. Wählt der Benutzer eine Vordiagnose aus, gelangt er zum Bildschirm aus Abbildung 8.3c. Hat der Benutzer ein Jahr angegeben, sieht die Darstellung in der Liste (Abb. 8.2c) wie folgt aus: *<Name der Vordiagnose> <Jahr>*. Ohne die Angabe eines Jahres wird in der Liste nur der Diagnosename angezeigt.

Die Medikationen (Abb. 8.2b) werden ebenfalls als Liste angezeigt.

Notfallkontakte werden mit Name und Telefonnummer ebenfalls in einer Liste angezeigt (Abb. 8.2d). Hat der Benutzer keinen Namen angegeben, wird nur die Telefonnummer angezeigt. Über das Telefonsymbol rechts neben dem Kontakt kann die Nummer des Kontaktes schnell gewählt werden.

Abbildung 8.3 zeigt die Eingabemasken zum Ändern der Patientendaten, einer Medikation, Vordiagnose und eines Notfallkontaktes. Es können nur die Patientendaten Vorname, Nachname sowie das Geburtsdatum angegeben werden (Abb. 8.3a). Letzteres kann über einen DatePicker ausgewählt werden.

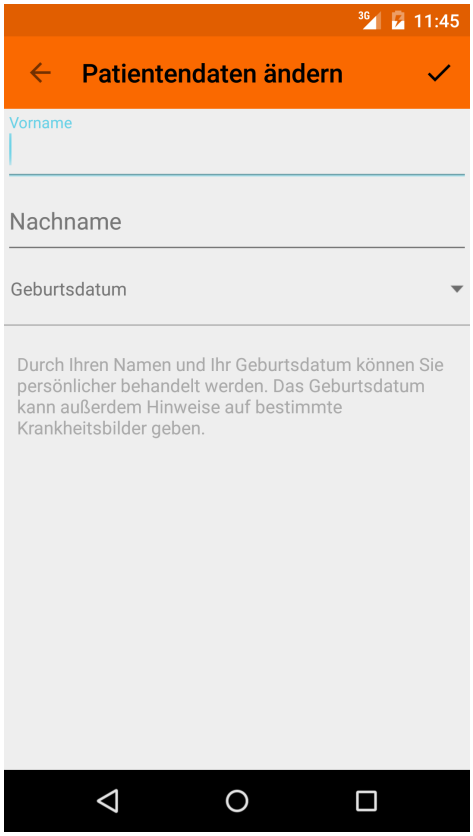
Eine Medikation gibt der Benutzer in das Textfeld ein (Abb. 8.3b). Dort werden ihm per Autovervollständigung Vorschläge gemacht, die er auswählen kann.

Vordiagnosen können mit Namen und Jahr hinzugefügt werden (Abb. 8.3c). Zusätzlich kann die Vordiagnose als Allergie oder Arzneimittelunverträglichkeit markiert werden. Auch hier werden bei der Eingabe des Diagnosenamens Vorschläge gemacht. Wenn ein Vorschlag ausgewählt wird, werden automatisch die Kontrollkästchen *Diagnose ist*

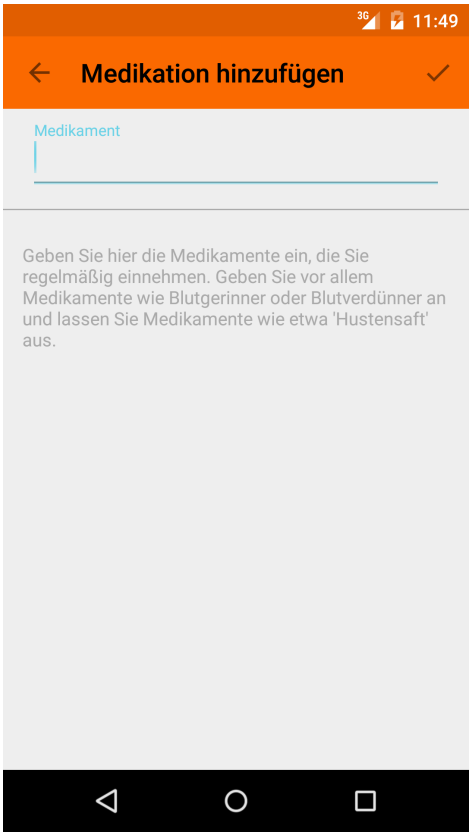
Allergie und *Diagnose ist Arzneimittelunverträglichkeit* deaktiviert, sowie abhängig von der ausgewählte Diagnose gesetzt.

Notfallkontakte können vom Benutzer mit Name und Telefonnummer selbst eingegeben oder aus dem Telefonbuch gewählt werden (Abb. 8.3d). Name und Telefonnummer werden dabei übernommen. Das Feld Telefonnummer ist ein Pflichtfeld, der Name ist optional.

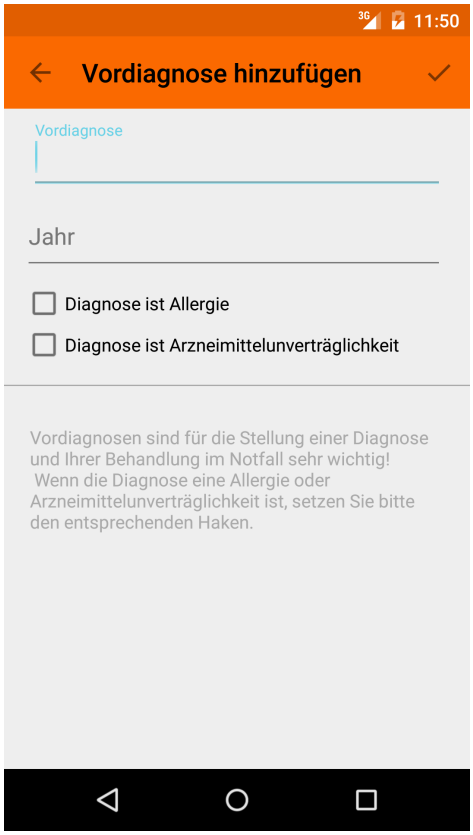
Auf allen Bildschirmen kann der Benutzer mit dem Pfeil links in der ActionBar auf den vorherigen Bildschirm zurückkehren ohne die Eingaben zu speichern und mit dem Haken rechts in der ActionBar die Eingaben speichern. Dann gelangt er zurück zum vorherigen Bildschirm.



(a) Hinzufügen und Editieren der Patientendaten



(b) Hinzufügen oder Editieren einer Medikation



(c) Hinzufügen oder Editieren einer Vordiagnose

(d) Hinzufügen oder Editieren eines Notfallkontaktes

Abbildung 8.3: Die verschiedenen Eingabemasken

8.5 Leseansicht

In der Leseansicht (Abb. 8.4) werden alle Notfalldaten des Patienten angezeigt. Dabei werden die Daten in Medikationen, Vordiagnosen, Allergien & Diagnosen und Notfallkontakte eingeteilt. Hinter dem Geburtsdatum wird in Klammern das Alter des Patienten angezeigt.

Notfallkontakte können über einen Klick auf das Telefonsymbol neben dem Namen und der Telefonnummer des Kontaktes angerufen werden.

Über den Button oben rechts in der ActionBar kann in die Schreibansicht umgeschaltet werden. Dabei werden alle Daten übernommen.

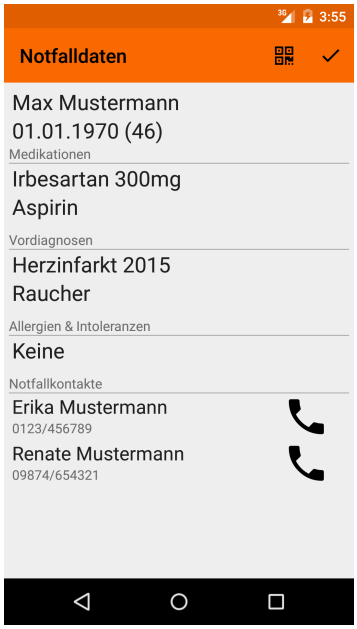


Abbildung 8.4: Leseansicht

Die Leseansicht wird immer aufgerufen, wenn ein NFC-Tag gescannt wird und die App dabei nicht im Vordergrund geöffnet ist.

8.6 AppIntro

Das AppIntro ist mit der Bibliothek AppIntro[1] realisiert. Es öffnet sich, wenn der Benutzer zum ersten Mal nach der Installation die App startet.

Das AppIntro zeigt dem Benutzer die Funktionsweise der App und die einzelnen Funktionen der App anhand von Screenshots.

8.7 Eingabeassistent

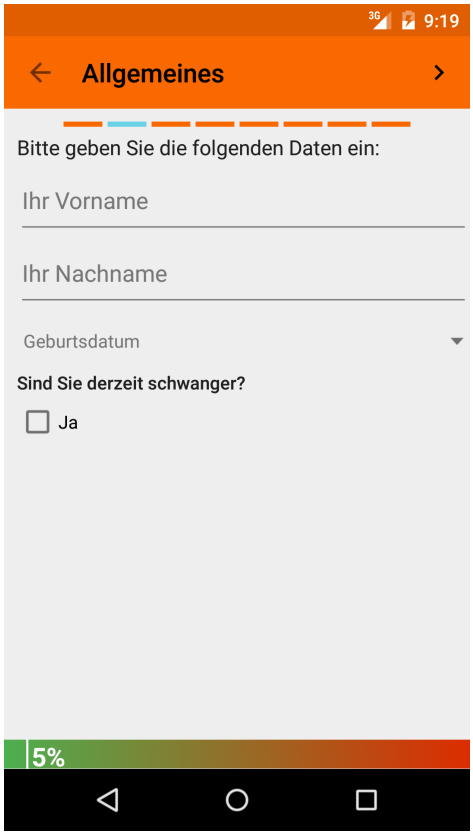
Der Eingabeassistent fragt den Benutzer nach seinen Patientendaten wie Name und Geburtsdatum (Abb. 8.5a) sowie nach Risikofaktoren (Abb. 8.5b) wie einer bestehenden Schwangerschaft oder bereits gestellten Vordiagnosen (Abb. 8.5c). Außerdem fragt er auch Notfallkontakte (Abb. 8.5f), Allergien (Abb. 8.5d) und Medikationen (Abb. 8.5e) ab. Zusätzlich zeigt der Assistent kurze Hinweistexte an, wieso bestimmte Notfalldaten wichtig sind.

Über die Fortschrittsanzeige unter der ActionBar sieht der Benutzer auf welcher Seite er sich befindet und wie viele noch folgen. Der Pfeil links in der ActionBar führt den Benutzer zu der vorherigen Seite zurück, der Pfeil rechts führt ihn zur nächsten Seite. Neue Listeneinträge können mit dem FloatingActionButton rechts unten auf dem Bildschirm hinzugefügt werden. Er erscheint immer, wenn auf der aktuellen Seite eine Liste enthalten ist.

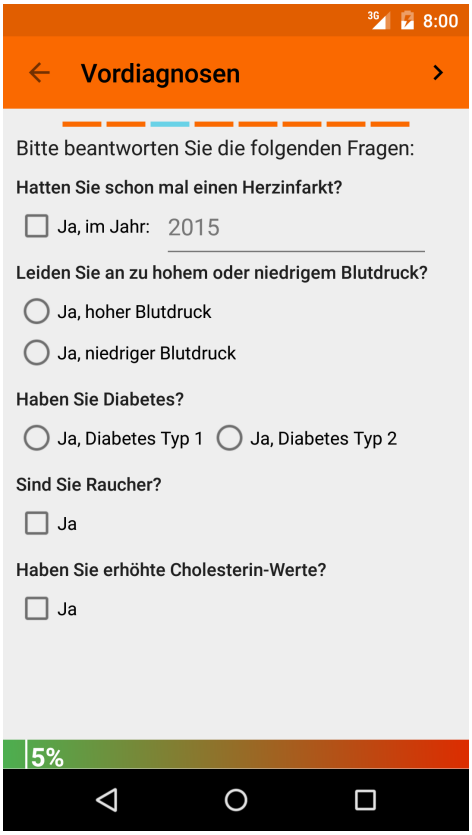
Unten auf jeder Seite befindet sich das FuelStateView, dessen Funktion bereits in Abschnitt 8.4 erklärt wurde.

Der Eingabeassistent ist mit der freien Bibliothek WizarDroid[2] erstellt. Jede einzelne Seite entspricht einem Fragment. Untereinander werden die eingegeben Daten des Benutzers durch einen Mechanismus der Bibliothek weitergegeben. Zum Hinzufügen von Medikationen und Vordiagnosen werden die in Tabelle 7.1 beschriebenen Activities genutzt.

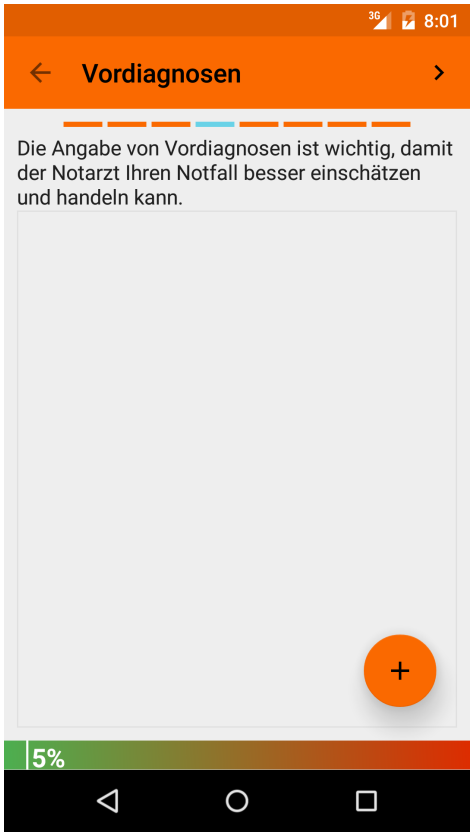
[1] `https://github.com/PaoloRotolo/AppIntro`. Letzter Zugriff: 16.02.2016
[2] `https://github.com/Nimrodda/WizarDroid`. Letzter Zugriff: 16.02.2016



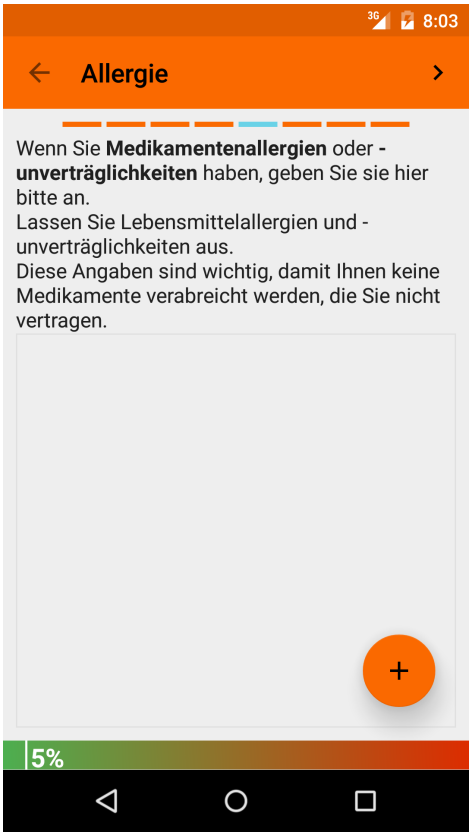
(a) Eingabe der Patientendaten



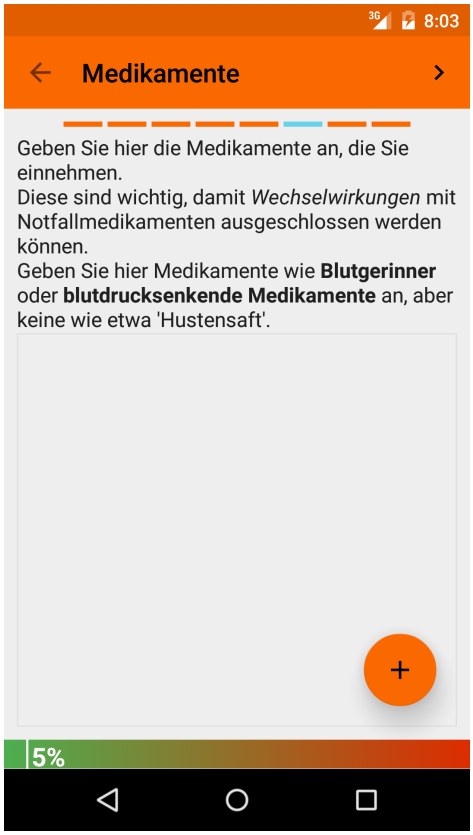
(b) Abfrage einiger Risikofaktoren



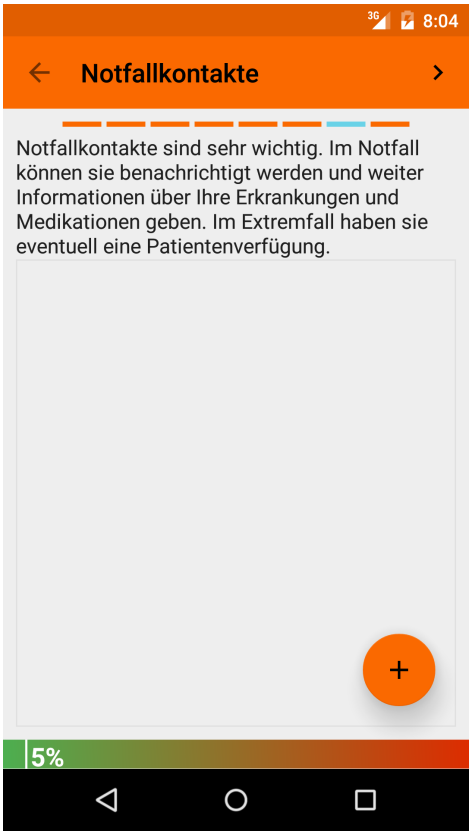
(c) Eingabe von Vordiagnosen



(d) Eingabe von Medikamentenallergien und -unverträglichkeiten



(e) Ansicht der Vordiagnosen



(f) Ansicht der Notfallkontakte

Abbildung 8.5: Die Seiten des Eingabeassistenten

8.8 Berechtigungen

Berechtigungen sind unter Android ein Diskussionsthema, nicht zuletzt wegen des Datenschutzes [22]. Daher sollte eine App nur die Berechtigungen einfordern, die sie notwendigerweise zur eigenen Funktion benötigt. Dieser Abschnitt beschreibt und begründet die Berechtigungen.

In Listing 8.2 werden die Berechtigungen, die die NExT App fordert, gezeigt.

Listing 8.2: Die Berechtigungen aus der AndroidManifest.xml

```
1 <uses-permission android:name="android.permission.READ_CONTACTS"/>
2 <uses-permission android:name="android.permission.NFC"/>
3 <uses-permission android:name="android.permission.VIBRATE"/>
4 <uses-permission android:name="android.permission.INTERNET"/>
```

Diese begründen sich wie folgt:

Berechtigung: android.permission.READ_CONTACTS

Die Berechtigung android.permission.READ_CONTACTS gibt an, dass die App auf die Kontaktliste zugreifen darf. Das ist nötig, damit der Benutzer seine Notfallkontakte direkt aus seiner Kontaktliste wählen kann.

Berechtigung: android.permission.NFC

Die Berechtigung android.permission.NFC besagt, dass die App auf den NFC-Sensor des Smartphones zugreifen darf. Die Begründung ergibt sich aus dem Namen der Berechtigung.

Berechtigung: android.permission.VIBRATE

Die Berechtigung android.permission.VIBRATE gibt an, dass die App eine Vibration auslösen darf. Beim Lesen und Schreiben der Daten auf einen Tag wird eine kurze Vibration ausgelöst, um dem Benutzer eine kleine Bestätigung der Vorgangs zu geben.

Berechtigung: android.permission.INTERNET

Die Berechtigung android.permission.INTERNET besagt, dass die App Verbindungen ins Internet herstellen darf. Dies wird für das Aktualisieren der Daten benötigt.

8.9 Verschlüsselung

Listing 8.3 zeigt, wie die Daten mithilfe der Java Cryptography Architecture verschlüsselt werden. Dabei wird der Stromverschlüsselungsalgorithmus RC4 genutzt.

Listing 8.4 zeigt, wie die RC4 verschlüsselten Daten ebenfalls mit der JCA entschlüsselt werden können.

Es wird eine Stromchiffre benutzt, da aufgrund des begrenzten Speicherplatzes auf dem NFC-Tag die Chiffre nicht mehr Speicherplatz einnehmen darf wie der Klartext. Die Vor- und Nachteile der Verschlüsselung werden in Kapitel Diskussion & Ausblick diskutiert.

Listing 8.3: Verschlüsselung der Daten mit RC4 mittels der Java Cryptography Architecture (JCA)

```
1 final byte[] key = new byte[]{14, 21, -67, -91, -35, -28, -12,
      -30, 0, 58, 2, -10, 109, -37, 21, -47};
2 final Cipher cipher;
3 SecretKey secretKey;
4 cipher = Cipher.getInstance("RC4");
5 secretKey = new SecretKeySpec(key, "RC4");
6 cipher.init(Cipher.ENCRYPT_MODE, secretKey);
```

```
byte[] secret = cipher.update(msg.getBytes(Charset.forName("UTF-8")));
```

Listing 8.4: Entschlüsselung der Daten mit RC4 mittels der Java Cryptography Architecture (JCA)

```
final byte[] key = new byte[]{14, 21, -67, -91, -35, -28, -12, -30, 0, 58, 2, -10, 109, -37, 21, -47};
final Cipher cipher;
SecretKey secretKey;
cipher = Cipher.getInstance("RC4");
secretKey = new SecretKeySpec(key, "RC4");
cipher.init(Cipher.DECRYPT_MODE, secretKey);
String message = new String(cipher.update(msg), Charset.forName("UTF-8"));
```

8.10 NFC

In diesem Abschnitt wird das Lesen und Schreiben von und auf NFC-Tags beschrieben.

8.10.1 Lesen

Die Activity `MainActivity` abonniert per Intent Filter das Event *NDEF_DISCOVERED*. Es wird ausgelöst, wenn ein NDEF formatierter NFC-Tag von Android erkannt wurde. Die NdefRecords vom NFC-Tag werden dann mit dem Android *Tag Dispatch System*[3] an die MainActivity gesendet. Listing 8.5 zeigt, wie der Intent Filter der MainActivity hinzufügt wird.

Listing 8.5: Intent Filter für die MainActivity zur Aktivierung beim NDEF_DISCOVERED Ereignis

```
<activity
        android:name=".activities.MainActivity"
        android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
    <intent-filter>
        <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:scheme="vnd.android.nfc"/>
    </intent-filter>
</activity>
```

[3] https://developer.android.com/guide/topics/connectivity/nfc/nfc.html#dispatching

In der `MainActivity` werden die NdefRecords ausgelesen und dem `ReadNfcTask` übergeben (Listing 8.6). Darin werden die Daten in einem separaten Thread geparst. Die Variable `readData` beinhaltet anschließend alle auf dem NFC-Tag verfügbaren Daten.

Listing 8.6: Lesen und Parsen der Daten von einem NFC-Tag

```
1 Parcelable[] msgs = getIntent().getParcelableArrayExtra(NfcAdapter
      .EXTRA_NDEF_MESSAGES);
2 ReadNfcTask task = new ReadNfcTask(dataParser, this);
3 try {
4     TagData readData = task.execute(msgs).get();
5 } catch (InterruptedException e) {
6     // hier ausgelassen
7 } catch (ExecutionException e) {
8     // hier ausgelassen
9 }
```

8.10.2 Schreiben

Über den beschriebenen Intent Filter wird auch beim Event *ACTION_TAG_DISCOVERED* die `MainActivity` aufgerufen. Listing 8.7 zeigt, wie dann die Daten auf einen NDEF formatierten NFC-Tag geschrieben werden können. Vorher werden die NdefRecords, die mit dem `DataParser` aus dem `TagData` Objekt erstellt werden, in einer `NdefMessage` gekapselt. Diese `NdefMessage` wird dann auf den NFC-Tag geschrieben.

Listing 8.7: Schreiben der Daten auf einen NFC-Tag

```
1 List<NdefRecord> ndefRecords = ...
2 NdefMessage msgMetadata = new NdefMessage(ndefRecords.toArray(new
      NdefRecord[ndefRecords.size()]));
3 Ndef ndef = Ndef.get(tag);
4
5 if (ndef != null) { // may be null if nfc tag does not support
      ndef
6     if (ndef.getMaxSize() < getSize(msgMetadata)) {
7         Log.e(TAG, "MemoryExceededException");
8         throw new MemoryExceededException(ndef.getMaxSize(),
              getSize(msgMetadata));
9     }
10    try {
11        ndef.connect();
12        ndef.writeNdefMessage(msgMetadata);
13    } catch (Exception e) {
14        throw new NfcWriteException(e);
15    } finally {
16        try {
17            ndef.close();
```

```
18          } catch (IOException e) {
19              e.printStackTrace();
20          }
21      }
22  }
```

8.11 Updatefunktion

Die Updatefunktion ist durch die `UpdateActivity` und die AsynTasks `CheckVersionTask`, `UpdateDiagnoses` und `UpdateMedications` realisiert. Alle Tasks Klassen erben von der Klasse `AbstractUpdateTask`.

Die `UpdateActivity` ruft zuerst die Klasse `CheckVersionTask` auf. Sie prüft, ob es eine neuere Version der Vordiagnosen oder Medikationen gibt. Die Klassen `UpdateDiagnoses` und `UpdateMedications` sind für das Holen und Verarbeiten der neuen Daten zuständig.

Die App sucht in einem festgelegten Intervall nach Aktualisierungen. Es kann vom Benutzer im Optionenmenü eingestellt werden.

Das in Abschnitt 7.7 Clientseitige Updateverarbeitung beschrieben Updateverhalten *(3) Alte Daten vorläufig löschen, neue Daten hinzufügen* wird ebenfalls durch die einzelnen Task Klassen implementiert.

8.12 QR-Code

Die Notfalldaten können auch in einem QR-Code codiert angezeigt werden (Abb. 8.6). Die Activity `QRCodeActivity` zeigt den QR-Code an und ist über das Optionenmenü erreichbar.

Abbildung 8.6: QR-Code in App

9 Diskussion & Ausblick

9.1 Diskussion

9.1.1 Umfrage

Die im Rahmen dieser Arbeit durchgeführte Umfrage hat aufgrund der geringen Teilnehmeranzahl nur wenig Aussagekraft. Dennoch decken sich die Ergebnisse mit einer Umfrage, die die Bundesärztekammer (BÄK) 2013 durchgeführt hat. Ziel dieser Umfrage war es, die Notfalldaten zu finden, die auf der eGK gespeichert werden sollen. Die Vorauswahl der Notfalldaten für die 2013 durchgeführte Umfrage wurde von der BÄK vorgenommen. Insgesamt wurden 37 Notärzte in der Notaufnahme, Notärzte (präklinisch) und Rettungsassistenten befragt. Als Ergebnis wurden *Diagnosen*, *Medikamente* und *Allergien und Unverträglichkeiten* als besonders wichtig eingeschätzt [23].

9.1.2 Updateserver

Das Backend wurde mit JavaEE umgesetzt und ist in einen TomEE Plus deployed. In der Produktivphase sollte der Updateserver hochverfügbar sein. Daher sollte das Backend in einem Cluster laufen, das einen vorgeschalteten Load Balancer hat. Die Datenbank läuft auch in einem Cluster und hat auch wiederum einen vorgeschalteten Load Balancer.

Da das Backend nicht nur zur Bereitstellung der Aktualisierungen, sondern auch als Fallback Funktion genutzt werden kann, muss eine Hochverfügbarkeit gegeben sein.

Für die Fallback Funktion sollten die Daten komprimiert über einen verschlüsselten Kanal übertragen werden.

9.1.3 Weitere Entwurfsmöglichkeiten

Die in Abschnitt 7.1 beschriebene Systemarchitektur hätte auch noch auf andere Weisen realisiert werden können. Einerseits ist die Realisierung als WebApp denkbar, bei der die Notfalldaten nur im Internet gespeichert werden. Andererseits kann die Systemarchitektur auch mit einem NFC-Tag und einer WebApp realisiert werden, wobei die Daten auf dem NFC-Tag und in der WebApp gespeichert werden.

WebApp

Der Benutzer meldet sich auf einer Webseite an und gibt dort seine medizinischen Notfalldaten ein. Diese werden in einer Datenbank verschlüsselt gespeichert. Es wird ein eindeutiger QR-Code generiert, der einen Zugangsschlüssel enthält. Der Benutzer kann den QR-Code ausdrucken und mit einer speziellen App auslesen. Danach zeigt die App die Notfalldaten bei einer Internetverbindung an.

Zusätzlich zur Verschlüsselung werden die Daten anonymisiert. Das ist jedoch nur in dem Maße möglich, wie der Benutzer sie eingibt. Gibt er seinen vollen Namen an, sind die Daten eindeutig zuordenbar.

Der generierte QR-Code beinhaltet einen Schlüssel, über den die Daten abgerufen werden können.

Bei dieser Realisierung fallen nur die Kosten für den gedruckten QR-Code an. Ferner muss auf dem auslesenden Smartphone die App installiert sein. Außerdem benötigt es eine Internetverbindung zum Laden der Daten.

NFC-Tag und WebApp

Grundsätzlich ist dieser Entwurf der Architektur aus Abschnitt 7.1 ähnlich. Die Daten werden aber zusätzlich noch im Internet gespeichert. Auf dem NFC-Tag werden die wichtigsten Notfalldaten sowie alle personenbezogenen Daten gespeichert, parallel dazu werden mit der WebApp alle restlichen Notfalldaten gesichert. Durch diese Trennung sind die Daten in der WebApp anonymisiert. Alle gespeicherten Daten werden verschlüsselt. Als Authentifizierungsschlüssel kann die ID des NFC-Tags dienen. Sie ist pro NFC-Tag eindeutig und 4 - 7 Byte lang.

Internet und Datenschutz

Beide Varianten wurden nicht für die Implementierung gewählt. Es kann nicht gewährleistet werden, dass an allen Orten, an denen eine Person einen medizinischen Notfall haben könnte, eine Internetverbindung besteht. Daher können im schlechtesten Falle nicht alle oder sogar keine Daten abgerufen werden. Der Informationsgewinn wäre gering oder nicht vorhanden.

Nicht zu vernachlässigen ist auch, dass die Speicherung der Gesundheitsdaten im Internet für viele Benutzer hinsichtlich der Akzeptanz ein Hindernis darstellt und die entwickelte Lösung daher mit Skepsis betrachtet wird.

Aus diesen Gründen sind alle Varianten, die die Abhängigkeit *Internet* besitzen, nicht für eine Implementierung zu empfehlen.

9.1.4 Datenverschlüsselung

Die Notfalldaten eines Patienten sind personenbezogen. Daher werden sie verschlüsselt abgespeichert, damit der Inhalt des NFC-Tags nicht mit anderen mobilen Anwendungen ausgelesen werden kann. Aufgrund des Anwendungsszenarios (s. Kapitel 1) müssen die Notfalldatendaten auf dem NFC-Tag auch ohne Mitwirken dessen Inhabers verfügbar sein. Deswegen müssen die Daten zwangsweise mit dem gleichen Schlüssel verschlüsselt werden.

Wenn ein NFC-Tag gescannt wird, der mit der NExT-App beschrieben wurde, und die NExT-App nicht installiert ist, öffnet sich der Google PlayStore. Nach der Installation der App kann der NFC-Tag ausgelesen werden.

Aus diesen beiden Gründen bietet die Verschlüsselung zwar eine erste Hürde, um den Zugriff unberechtigter Personen auf die Daten zu verhindern, ist aber nicht sehr effektiv.

9.1.5 Informationsqualität der Daten auf dem NFC-Tag

Da jeder Mensch das Recht auf informationelle Selbstbestimmung hat[1], soll der Benutzer selbst entscheiden können, welche der in Abschnitt 6.4.2 ausgewählten Notfalldaten er auf dem NFC-Tag speichern will. Das kann jedoch die Qualität der abgespeicherten Daten negativ beeinflussen, wenn Daten gespeichert werden, die nicht für eine Erstversorgung relevant sind.

Ein weiterer Aspekt ist die Aktualität der abgespeicherten Notfalldaten. Der Informationsgehalt beispielsweise einer eingegebenen Medikation könnte sinken, wenn nicht ersichtlich ist, wann der Eintrag vorgenommen wurde. Ebenso verhält es sich mit den Vordiagnosen. Aus dem genannten Grund sollte mit mehreren Notärzten evaluiert werden, inwiefern das Alter dieser Informationen bei der Notfallversorgung relevant ist.

9.1.6 Evaluation - User Acceptance Test

Bevor NExT produktiv eingesetzt wird, sollte noch eine Evaluation der Benutzung seitens der Rettungskräfte und Patienten stattfinden. Dabei sollte die Akzeptanz des Systems, die Benutzeroberfläche, Bedienbarkeit und Relevanz der abgespeicherten Daten für einen medizinischen Notfall evaluiert werden.

9.1.7 Medizinproduktegesetz

Mobile Anwendungen fallen unter das Medizinproduktegesetz, wenn sie

(a) *„der Erkennung, Verhütung, Überwachung, Behandlung oder Linderung von Krankheiten,*

(b) *der Erkennung, Überwachung, Behandlung, Linderung oder Kompensierung von Verletzungen oder Behinderungen,*

(c) *der Untersuchung, der Ersetzung oder der Veränderung des anatomischen Aufbaus oder eines physiologischen Vorgangs oder*

[1] BVerfG, Urteil vom 15.12.1983 - 1 BvR 209/83; 1 BvR 269/83; 1 BvR 362/83; 1 BvR 420/83; 1 BvR 440/83; 1 BvR 484/83

(d) *der Empfängnisregelung“ (§1 Nr.1 MPG)*

dienen. Da laut Bundesamt für Arzneimittel und Medizinprodukte eine „[r]eine Datenspeicherung [...] nicht zu einer Einstufung als Medizinprodukt“[24] führt und die App nur das macht, ist sie voraussichtlich kein Medizinprodukt. Dies sollte allerdings durch einen sachkundigen Juristen überprüft werden.

9.1.8 Datenkompression

Wenn die Daten komprimiert abgespeichert werden, verbrauchen sie weniger Speicherplatz auf dem NFC-Tag und es können mehr Daten gespeichert werden. Durch die Speicherung der vordefinierten Daten mit deren Surrogaten anstelle von Zeichenketten wird bereits eine gewisse Kompression erreicht. Die vom Benutzer eingegebenen Zeichenketten werden bisher nicht komprimiert. Mit dem empfohlenen NFC-Tag *MIFARE DESFire EV1 4k* steht mit einer Speicherkapazität von 4096 Bytes auch genug Speicherplatz zur Verfügung.

9.2 Ausblick

9.2.1 Updateverhalten

Im Laufe der Entwicklung der App hat sich herausgestellt, dass das implementierte Updateverhalten *„Daten hinzufügen, keine Daten löschen“* nicht sehr effektiv ist, da nur Daten hinzugefügt, aber nicht gelöscht werden können. Dies könnte jedoch notwendig sein, wenn bspw. Medikamente vom Markt genommen werden oder Namen von Vordiagnosen geändert werden müssen. Daher ist die in Abschnitt 7.7 beschriebene Strategie *„Alte Daten vorläufig löschen, neue Daten hinzufügen“* besser geeignet und sollte in zukünftigen Versionen der App implementiert werden.

9.2.2 Fallback

Die in Abschnitt 7.6.1 beschriebene Fallback Lösung wurde aus Zeitgründen nicht implementiert. Da aber wichtige Informationen zur Behandlung des Patienten liefern könnte, sollte diese Funktion in Zukunft umgesetzt werden.

Da sie aber Kosten verursachen kann, sollte sie im Optionenmenü deaktivierbar sein.

9.2.3 Automatisiertes Testen

Aus Zeitgründen wurde auf ein automatisiertes Testen der Anwendung verzichtet. Stattdessen wurde sie mithilfe von Testprotokollen getestet. Um die Codequalität ohne den Aufwand des manuellen Testens sicherzustellen, sollten automatische Unit Tests genutzt werden.

9.2.4 Interoperabilität

Nicht alle Anforderungen konnten im Rahmen dieser Thesis umgesetzt werden. Darunter fällt auch teilweise die Anforderung nach Interoperabilität. Mögliche Lösungen werden hier beschrieben.

Integration in ein Krankenhausinformationssystem (KIS)

Die Notfalldaten des Patienten sind bisher nur auf dem NFC-Tag gespeichert und mit der App auslesbar. Da die Notfalldaten während einer Notfallbehandlung erst in ein Notarzteinsatzprotokoll und in der Notaufnahme in das KIS des Krankenhauses übernommen werden, bietet sich die Möglichkeit an, die Daten direkt an das KIS zu senden.

Zum einen gibt es bereits elektronische Systeme zur Einsatzprotokollierung, die auch schon in ein KIS integriert sind wie bspw. das Projekt *MEER* des Fraunhofer-Institut für Experimentelles Software Engineering. Daher könnte es bereits ausreichend sein, die NExT-App in ein solches System zu integrieren. Die App bietet dazu einen QR-Code an, in den die Notfalldaten kodiert sind. Außerdem ist die Spezifikation der Speicherstruktur auf dem NFC-Tag öffentlich gemacht.

Zum anderen könnten die Notfalldaten auch direkt an das KIS gesendet werden. Dazu bietet sich HL7 Fast Healthcare Interoperability Resources (FHIR) an, sofern das KIS einen FHIR Server betreibt. FHIR ist ein neuer Standard, der den Datenaustausch zwischen Softwaresystemen im Gesundheitswesen vereinfachen will [25]. Er befindet sich derzeit allerdings erst in der Entwicklungsphase.

Die App könnte direkt auf den FHIR Server die Notfalldaten senden. Da der Benutzer seinen Namen und Geburtsdatum in die App eingibt, kann der Patient auch auf dem Server identifiziert werden. Es muss allerdings sichergestellt werden, dass nur berechtige Personen wie bspw. Notärzte die Notfalldaten an den FHIR Server schicken können.

Daten aus dem Bundesmedikationsplan einlesen

Der Bundesmedikationsplan (BMP) ist ein papierbasiertes Dokument, das eine Auflistung enthält, welche Medikamente ein Patient einnehmen muss. Spezifiziert wird der BMP von der Arzneimittelkommission der deutschen Ärzteschaft. Das Ziel des BMP ist es, die Arzneimitteltherapiesicherheit zu gewährleisten, wenn mehrere Ärzte einem Patienten verschiedene Medikamente verschreiben. Er wird i.d.R. von einem Arzt oder Apotheker erstellt. Auf dem BMP ist ein DataMatrix-Code aufgedruckt, der die Interoperabilität zu einer Software herstellen soll [26].

Dieser DataMatrix-Code könnte mit der Kamera der Smartphones gescannt werden. Dadurch könnten die Medikationen direkt in die App übertragen werden und der Benutzer müsste nichts mehr von Hand eintragen. Da auf dem BMP auch der Vorname, Nachname und das Geburtsdatum vermerkt sind, müsste der Benutzer nur noch Vordiagnosen und Notfallkontakte eingeben. Die so eingetragenen Medikationen hätten eine vergleichsweise höhere Informationsqualität, da der BMP von einem Arzt oder Apotheker erstellt wird.

Alle Personen, die drei oder mehr Arzneimittel regelmäßig einnehmen, haben „ab Oktober 2016 einen Anspruch auf einen Medikationsplan“ [13].

9.2.5 Vertrieb

Der Vertrieb des NFC-Tags und der App kann bspw. über den Hausarzt erfolgen. Dieser kennt in der Regel die Dauermedikationen des Patienten sowie dessen Vordiagnosen. Wenn der Hausarzt die Notfalldaten auswählt und auf den NFC-Tag schreibt, haben sie aus den genannten Gründen eine höhere Qualität (s. Abschnitt 9.1.5). Der Arzt kann auch leichter entscheiden, welche Informationen wichtig sind und welche nicht.

Eine andere Möglichkeit wäre es, den Vertrieb über die Krankenkassen zu regeln. Sie kennen i.d.R. ebenfalls die Medikationen und Vordiagnosen ihrer Versicherten.

Beide Varianten haben den Vorteil, dass ältere Personen, die nicht in der Lage sind, einen NFC-Tag zu kaufen und auch kein Smartphone haben, einen NExT-Tag besitzen können.

Denkbar ist auch die Kombination eines NExT-Tags mit einem Notfallarmband. Auf dem Notfallarmband sind vom Benutzer definierte Daten eingraviert. Das Armband beinhaltet einen NFC-Tag, wodurch weitere Daten gespeichert werden können. Diese Daten können wie in Kapitel 7 *Entwurf* beschrieben mit der NExT-App auf den NFC-Tag im Armband geschrieben und von dort gelesen werden. Falls kein Smartphone mit NFC-Sensor zur Verfügung steht, können Erst- und Zweithelfern dennoch einige Notfalldaten geliefert werden.

Auch sollte das Format des NExT-Tags betrachtet werden. Es ist möglich, einen NFC-Tag in vielen Formen herzustellen. Denkbar wäre eine Halskette, ein Armband oder ein Schlüsselanhänger. Der NExT-Tag sollte durch eine farbliche Hervorhebung gut erkennbar und leicht zu finden sein.

Weiterhin müssen Mitarbeiter des Rettungsdienstes und Notärzte geschult werden, damit sie den NExT-Tag erkennen und die NExT-App nutzen können.

Wie in Abschnitt 8.3.1 beschrieben, können die Rote Liste, Gelbe Liste und Grüne Liste als Quelle für die vordefinierten Medikationen genutzt werden. Diese müssen

noch beschafft und eingepflegt werden, da in dieser Arbeit nur Testdaten der Roten Liste genutzt wurden.

Die vordefinierten Vordiagnosen können wie beschrieben vom ICD-10-GM Code genommen werden (s. Abschnitt 8.3.1). Da diese jedoch medizinische Termina enthalten, die für medizinische Laien nicht verständlich sind, sollten diese in die Laiensprache übersetzt werden.

9.2.6 Erweiterung des Funktionsumfang

In einer nächsten Version könnten noch die folgenden Funktionen eingebaut werden. Zum einen ein Knopf, mit dem der Notruf direkt gewählt werden kann. Zum anderen eine Funktion, die es dem Benutzer erlaubt, seinen aktuellen Standort auf einer Karte anzusehen. Zusätzlich können der Ort und der Straßenname angezeigt werden. Damit kann beim Absetzen des Notrufs die Frage nach dem Standort präzise beantwortet werden und der Rettungsdienst kann den Einsatzort schneller finden. Das kommt vor allem dem Patienten zugute.

10 Literaturverzeichnis

[1] A. Stiell, A. J. Forster, I. G. Stiell, and C. Van Walraven, "Prevalence of information gaps in the emergency department and the effect on patient outcomes," *Cmaj*, vol. 169, no. 10, pp. 1023–1028, 2003.

[2] J. Snell, D. Tidwell, and P. Kulchenko, "Webservice-Programmierung mit SOAP," 2002.

[3] K. D. J. Renate Huch, *Mensch Körper Krankheit*. Urban & Fischer: Springer-Verlag, 2011. aufl ed., 2011.

[4] W. Behringer, U. Buergi, M. Christ, C. Dodt, and B. Hogan, "Fünf Thesen zur Weiterentwicklung der Notfallmedizin in Deutschland, Österreich und der Schweiz," *Notfall + Rettungsmedizin*, vol. 16, no. 8, pp. 625–626, 2013.

[5] Deutsches Institut für Medizinische Dokumentation und Information, "ICD-10-GM." `https://www.dimdi.de/static/de/klassi/icd-10-gm/`. – Letzter Zugriff: 06.02.2016.

[6] NFC Forum, "NFC and Contactless Technologies." `http://nfc-forum.org/what-is-nfc/about-the-technology/`. – Letzter Zugriff: 06.01.2016.

[7] J. Langer and M. Roland, *Anwendungen und Technik von Near Field Communication (NFC)* -. Berlin Heidelberg New York: Springer-Verlag, 2010. aufl ed., 2010.

[8] NFC Forum, "About us." `http://nfc-forum.org/about-us/`. – Letzter Zugriff: 27.01.2016.

[9] "NFC Forum Technical Specifications." `http://nfc-forum.org/our-work/specifications-and-application-documents/specifications/nfc-forum-technical-specifications/`. – Letzter Zugriff: 10.02.2016.

[10] I. Melzer, *Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis*, ch. Service-orientierte Architektur, pp. 9–31. Heidelberg: Spektrum Akademischer Verlag, 2010.

[11] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," *Building*, vol. 54, p. 162, 2000.

[12] Gematik, "Anwendungen der eGK." `https://www.gematik.de/cms/de/egk_2/anwendungen/anwendungen_1.jsp`. – Letzter Zugriff: 06.01.2016.

[13] Bundesministerium für Gesundheit, "Das E-Health-Gesetz." `http://www.bmg.bund.de/themen/krankenversicherung/e-health-gesetz/e-health.html`. – Letzter Zugriff: 06.01.2016.

[14] J. Yang, Z. Wang, and X. Zhang, "An ibeacon-based indoor positioning systems for hospitals," *International Journal of Smart Home*, vol. 9, no. 7, pp. 161–168, 2015.

[15] P. Martin, B.-J. Ho, N. Grupen, S. Muñoz, and M. Srivastava, "An iBeacon Primer for Indoor Localization: Demo Abstract," in *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14, (New York, NY, USA), pp. 190–191, ACM, 2014.

[16] ISO, "Information technology – Automatic identification and data capture techniques – QR Code bar code symbology specification," ISO ISO/IEC 18004:2015, International Organization for Standardization, Geneva, Switzerland, 2015.

[17] INCORPORATED, DENSO WAVE, "Information capacity and versions of the QR code." `http://www.qrcode.com/en/about/version.html`. – Letzter Zugriff: 27.01.2016.

[18] T. Schneider, B. Wolcke, and R. Böhmer, "Notfalldiagnostik," in *Taschenatlas Notfall & Rettungsmedizin*, pp. 52–69, Springer Berlin Heidelberg, 2010.

[19] Google, "Build System Overview." `https://developer.android.com/sdk/installing/studio-build.html`. – Letzter Zugriff: 02.02.2016.

[20] Kantar Worldpanel ComTech, "Smartphone OS sales market share." `http://www.kantarworldpanel.com/global/smartphone-os-market-share/`. – Letzter Zugriff: 10.02.2016.

[21] Google, "Platform Versions." `https://developer.android.com/about/dashboards/index.html#Platform`. – Letzter Zugriff: 10.02.2016.

[22] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *in Proceedings of the 18th ACM conference on Computer and communications security. ACM*, pp. 627–638.

[23] J. Schenkel, J. Albert, N. Butz, J. Born, and C. Juhra, "Notfalldatenmanagement: Was genau soll auf die Karte?," *Dtsch Arztebl International*, vol. 112, no. 19, pp. A–866–, 2015.

[24] Bundesinstitut für Arzneimittel und Medizinprodukte, "Orientierungshilfe Medical Apps." `http://www.bfarm.de/DE/Medizinprodukte/Abgrenzung/medical_apps/_node.html`. – Letzter Zugriff: 02.02.2016.

[25] HL7, "FHIR Overview - Clinicians." `http://hl7.org/fhir/overview-clinical.html`. – Letzter Zugriff: 13.02.2016.

[26] D. H. M. Dr. Farid Aly, Dr. Gunther Hellmann, "Spezifikation Medikationsplan," pp. 1–44, 2012.

[27] Gematik, "Viel mehr als eine neue Krankenversichertenkarte." `https://www.gematik.de/cms/de/egk_2/egk_3/egk_2.jsp`. – Letzter Zugriff: 06.01.2016.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

(Ort, Datum) (Unterschrift)