ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Umsetzung eines Fragebogens zur Bewertung von Risikofaktoren während Schwangerschaften mit modernen Webtechnologien

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Fabian Fischer
fabian-1.fischer@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Johannes Schobel

2017

Fassung 30. Januar 2017

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Viele Befragungen in der klinischen Psychologie oder dem allgemeinen Gesundheitsbereich finden noch immer durch Papierfragebögen statt. Der KINDEX - Mum Screen, der zur Erhebung und Bewertung von Risikofaktoren während der Schwangerschaft dient, ist ein Beispiel dafür. Die papierbasiert erhobenen Daten werden zur Auswertung in ein digitales Format übertragen. Dies geschieht oft von Hand und ist dadurch fehleranfällig und zeitintensiv. Eine elektronische Umsetzung der papierbasierten Fragebögen wäre wünschenswert. Dadurch könnten die Daten direkt in einer digitalen Form erhoben werden. Dies würde somit den Schritt der fehleranfälligen Digitalisierung eliminieren.

Da die Entwicklung mobiler Geräte wie Smartphones oder Tablets voranschreitet, gibt es eine Vielzahl unterschiedlicher Geräte mit unterschiedlichen Betriebssystemen auf dem Markt. Die native Umsetzung eines Fragebogens für alle Plattformen erfordert die Kenntnis verschiedener Programmiersprachen und nimmt viel Zeit in Anspruch. Durch moderne Webtechnologien wird eine mobile Applikation umgesetzt, die auf mehreren Plattformen lauffähig ist. Diese kann die Daten der papierbasierten Fragebögen in einer digitalen Form erheben, sodass sie zur Auswertung direkt in einem digitalen Format vorliegen. Dadurch wird der Prozess der Digitalisierung analoger Daten eliminiert, wodurch für die beteiligten Personen ein sehr aufwändiger Schritt entfällt und gleichzeitig die Qualität der Daten steigt.

Danksagung

An erster Stelle möchte ich mich beim Institut für Datenbanken und Informationssysteme der Universität Ulm sowie deren Mitarbeiter bedanken. Durch Prof. Dr. Manfred Reichert wurde mir die Bearbeitung der Bachelorarbeit mit einem Thema ermöglicht, zu dem ich auch eine Anwendung implementieren konnte. Mein größter Dank geht an meinen Betreuer Johannes Schobel. Durch seine immer freundliche und motivierende Art aber auch durch seine sachliche Kompetenz hat er mich bei dieser Arbeit tatkräftig unterstützt.

Weiterhin danke ich meiner Familie für die Unterstützung während dieser Arbeit und des gesamten Studiums. Besonderen Dank geht an meinen Vater Otto, durch den mir einige Rechtschreibfehler und grammatikalisch falsche Sätze in dieser Arbeit erspart blieben.

Abschließend bedanke ich mich bei all denjenigen, die mich in meinem Studium in irgendeiner Art und Weise unterstützt haben.

Herzlichen Dank!

Inhaltsverzeichnis

1

Einleitung

In der heutigen Zeit sind mobile Endgeräte (wie Smartphones oder Tablets) omnipräsent. In den jüngeren Generationen nutzt sogut wie jeder ein Smartphone oder ein Tablet. Auch in den älteren Generationen ist eine ständige Zunahme der Nutzung zu erkennen. Laut einer Umfrage setzt sich die Internetnutzungsdauer in Deutschland zu 51% aus mobilen Endgeräten zusammen [1]. Dies bedeutet, dass jetzt schon mehr Zeit mit Smartphone oder Tablet im Internet verbracht wird, als mit herkömmlichen PCs oder Notebooks. Das mobile Gerät ist somit für den Benutzer zum alltäglichen Begleiter geworden. Dabei kommen mobile Applikationen für die unterschiedlichsten Zwecke zum Einsatz, wie zum Beispiel für Videostreaming, Online-Banking, Kochrezepte oder zum Abrufen der nächsten Bahnverbindung. Dem Benutzer wird in jedem Fall ein gewisser Komfort geboten, da durch die Applikationen bisher komplexe und aufwändige Vorgänge vereinfacht werden.

In vielen Bereichen des Alltags wird in zunehmendem Maße die Möglichkeit genutzt, durch mobile Applikationen existierende Abläufe zu erleichtern. Werden allerdings Anforderungen an eine Unterstützung durch mobile Applikationen explizit definiert, kann oft das mögliche Potential aufgrund nicht vorhandener Lösungen auch nicht genutzt werden.

Ein Beispiel dafür ist unter anderem in der klinischen Psychologie zu finden. Dort finden heute noch viele Befragungen auf traditionelle Weise (mit Papier und Stift) statt. Dies ist beispielsweise auch beim KINDEX - Mum Screen, der zur Bewertung von Risikofaktoren während der Schwangerschaft dient, der Fall. Dabei haben papierbasierte Fragebögen im heutigen Zeitalter einen offensichtlichen Nachteil: Sie sind analog und eine effiziente Verarbeitung der Daten ist nur dann möglich, wenn diese digital vorliegen.

Für eine elektronische Auswertung müssen also die analog erhobenen Daten vorher in ein digitales Format (beispielsweise Excel-Tabellen) übertragen werden. Genau dieses Problem tritt auch beim KINDEX auf. Die Auswertung des Fragebogens erfolgt in einem sehr ineffizienten Prozess, dessen Dauer sich teilweise über mehrere Wochen erstrecken kann.

Eine Unterstützung durch eine mobile Applikation würde diesen Prozess effizienter gestalten. Die Realisierung einer mobilen Applikation zur elektronischen Umsetzung eines Fragebogens bringt jedoch auch Probleme mit sich, denn die Entwicklung immer leistungsstärkerer mobiler Geräte schreitet unaufhaltsam voran. Die technischen Spezifikationen der mobilen Geräte, die sich momentan auf dem Markt befinden, variieren daher sehr stark. Um allen Nutzern eine mobile Applikation zur Verfügung zu stellen, muss diese für jede Plattform verfügbar sein. Die native Entwicklung für eine Plattform erfordert allerdings Kenntnis der zugrunde liegenden Programmiersprache und nimmt durch die Umsetzung in verschiedenen Programmiersprachen viel Zeit in Anspruch.

Das Ziel der hier vorliegenden Arbeit ist es also, den KINDEX-Fragebogen mit Hilfe von modernen Webtechnologien in ein digitales Format zu übertragen. Dadurch soll ermöglicht werden, dass Auswertungsprozesse effizienter ablaufen können. Die Flexibilität des Papierfragebogens soll in Form einer mobilen Applikation erhalten bleiben und gleichzeitig soll die Erhebung der Daten digital stattfinden. Die mobile Applikation soll so aufgebaut werden, dass nicht nur der KINDEX, sondern auch beliebig andere Fragebögen unterstützt werden können und eventuelle Änderungen an den Fragebögen nicht über eine Änderung des Programmcodes vorgenommen werden müssen [2]. Deswegen sollen alle Daten, die einem Fragebogen zugeordnet sind, in externe Dateien ausgelagert werden.

1.1 Struktur der Arbeit

Der weitere Verlauf der Arbeit ist in 7 Kapitel gegliedert (siehe Abbildung 1.1). In Kapitel 2 werden Grundlagen geschaffen, die im Zusammenhang mit dem KINDEX und der Entwicklung einer hybriden mobilen Applikation stehen. Das 3. Kapitel diskutiert verwandte

Arbeiten. Im darauffolgenden Kapitel 4 wird das Anwendungsszenario anhand von Ist- und Soll-Zustand beschrieben. Im 5. Kapitel wird auf Basis der bisherigen Erkenntnisse die Anforderungsanalyse durchgeführt. Das 6. Kapitel beinhaltet die Konzeption und Architektur anhand der ermittelten Anforderungen. Im 7. Kapitel wird die Implementierung der mobilen Applikation beschrieben. Abschließend wird die Arbeit im 8. Kapitel zusammengefasst und zusätzlich ein Ausblick auf mögliche Erweiterungen der mobilen Applikation gegeben.

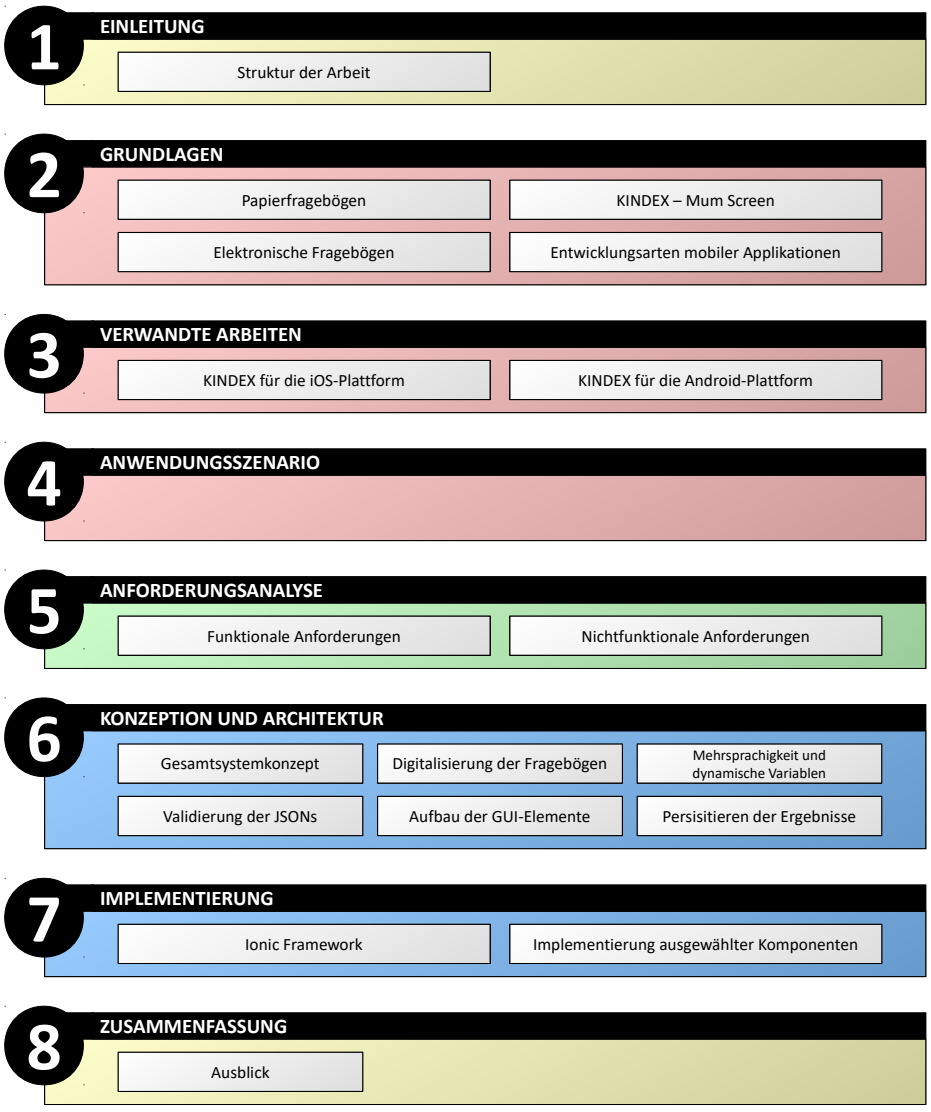


Abbildung 1.1: Struktur der Arbeit

2

Grundlagen

In diesem Kapitel werden die Grundlagen geschaffen, die für die Entwicklung einer mobilen Applikation am Beispiel des KINDEX - Mum Screen relevant sind. Zuerst wird in Abschnitt 2.1 aufgezeigt, was ein Papierfragebogen ist und welche Probleme er mit sich bringt. Anschließend wird in Abschnitt 2.2 auf den KINDEX - Mum Screen eingegangen. In Abschnitt 2.3 werden elektronische Fragebögen und deren Vorteile vorgestellt. Zum Schluss werden in Abschnitt 2.4 verschiedene Entwicklungsarten für mobile Applikationen veranschaulicht.

2.1 Papierfragebögen

Papierfragebögen sind Fragebögen, bei denen die Erhebung der Daten traditionell und auf analogem Wege abläuft. Hierbei werden die gestellten Fragen von Hand bearbeitet und in einem vorgesehenen Bereich auf einem vorgedruckten Fragebogen mit einem Stift niedergeschrieben. In vielen Befragungen kommt diese traditionelle Methode zur Datenerhebung noch immer zum Einsatz, da es unter anderem für alle Beteiligten die bekannteste und einfachste Art und Weise ist. Da dieses Verfahren regelmäßig angewendet wird, ist dessen Vorhergehensweise allgemein bekannt [3].

Meist werden die Daten eines ausgefüllten Fragebogens von Hand in ein digitales System übertragen, wodurch die Auswertung der gesammelten Daten vereinfacht wird. Je nach Umfang des Fragebogens und der Anzahl der Teilnehmer stellt dies für die Person, welche die Daten überträgt, einen erheblichen Zeitaufwand dar. Außerdem besteht die Gefahr, dass es bei der Übertragung der Daten zu Abweichungen kommt, beispielsweise wenn die Handschrift eines Teilnehmers bei einer Freitext-Frage schwer zu entziffern ist [4].

2.2 KINDEX - Mum Screen

Ein Beispiel für einen papierbasierten Fragebogen ist der KINDEX - Mum Screen, welcher zur Erhebung psychosozialer Belastungen während Schwangerschaften dient. Dabei werden einer Schwangeren von Gynäkologen, Hebammen oder Mitarbeitern einer Beratungsstelle, verschiedene Fragen zu ihrem Alltag während der Schwangerschaft gestellt. Dadurch werden mögliche Risiken ermitteln. Die Schwangere wird dabei entweder interviewt oder sie füllt den Fragebogen als Selbstbericht eigenständig aus. Zur Auswertung wird der ausgefüllte Fragebogen postalisch an Psychologen der Universität Konstanz geschickt und dort von Mitarbeitern analysiert. Hierzu werden die Antworten des Fragebogens in ein elektronisches System übertragen. Die Ergebnisse der Analyse werden, ebenfalls per Post, an den Absender zurückgeschickt, wodurch bis zu sechs Wochen vergehen können, bis die Schwangere eine Rückmeldung erhält. Erst dann könnten weitere Schritte durch Gynäkologen eingeleitet werden um die Schwangere entsprechend zu unterstützen. [5].

Am Beispiel des KINDEX-Fragebogen werden die Probleme herkömmlicher Papierfragebogen recht deutlich. Die Zeitspanne von der Bearbeitung des Fragebogens bis zur tatsächlichen Auswertung und Weiterleitung an die Schwangere ist sehr groß. Dies liegt unter anderem daran, dass die Universität Konstanz einen erheblichen Zeitaufwand bei der individuellen Erfassung und Auswertung der Fragebögen hat [5]. Wünschenswert wäre eine Lösung, die den Zeitaufwand für die Mitarbeiter auf ein Minimum reduziert und eine sofortige Rückmeldung an die Schwangere bietet. Dazu gibt es bereits Ansätze, welche unter anderem diese Problematik beseitigen. Diese werden später in Kapitel 3 genauer betrachtet.

2.3 Elektronische Fragebögen

Im Zeitalter von allgegenwärtigen mobilen Endgeräten (z.B. Smartphones und Tablets) nimmt die Anzahl an elektronisch umgesetzten Fragebögen im Bereich der Gesundheitsvorsorge und der klinischen Forschung stetig zu [6]. Dabei wird das klassische Verfahren

des Papierfragebogens auf ein elektronisches Gerät übertragen. In mehreren Studien wurden diese Fragebögen im Vergleich zu der papierbasierten Variante evaluiert. Dabei stellte sich heraus, dass sich in der Genauigkeit der Ergebnisse keine signifikanten Abweichungen der Resultate ergaben und die allgemeine Fehlerrate und Auslassungen von Fragen bei elektronischen Fragebögen sogar niedriger war [6, 7, 8]. 59 % der Studienteilnehmer bevorzugten gar die elektronische gegenüber der papierbasierten Variante (19 %), während 22 % keine Bevorzugung hatten [6].

Elektronische Fragebögen bieten viele Vorteile gegenüber Papierfragebögen. Diese können so programmiert werden, dass sie auf eine Frage nur begrenzte oder vordefinierte Antwortmöglichkeiten zulassen (zum Beispiel die Eingabe des Alters muss eine positive Ganzzahl sein). Bei der Eingabe der Antwort kann gleichzeitig ein Zeitstempel hinterlegt oder das Überspringen von Fragen limitiert werden. Fragen, die nur unter bestimmten Voraussetzungen beantwortet werden sollen, können automatisch ausgeblendet werden. Die Möglichkeit, Antworten zu bereits ausgefüllten Fragen im nachhinein noch einmal zu ändern, kann verhindert werden. Einer der größten Vorteile ist allerdings, dass die Daten bereits elektronisch erfasst werden und somit direkt ausgewertet werden können. Dies reduziert den Arbeitsaufwand bei der Verarbeitung der Daten erheblich und erhöht die Qualität der erhobenen Daten [6].

Da beim KINDEX das größte Problem der Arbeitsaufwand bei der Auswertung der Daten ist, erspart eine elektronische Umsetzung des Fragebogens den Mitarbeitern viel Arbeit. Des weiteren könnten die Schwangeren in einem Selbstbericht, unabhängig von ihren Gynäkologen, Hebammen oder Mitarbeitern von Beratungsstellen, den Fragebogen ausfüllen und dessen Daten elektronisch zur Auswertung zur Verfügung stellen.

2.4 Entwicklungsarten mobiler Applikationen

Entsprechend den Anforderungen einer mobilen Applikation, kann auf unterschiedliche Entwicklungsarten zurückgegriffen werden. IBM spezifiziert vier verschiedene Kategorien in die mobile Applikationen eingeordnet werden können (siehe Abbildung 2.1) [9].

Abbildung 2.1: Entwicklungsarten für mobile Applikationen (nach [9])

Web-Applikation Gewöhnlich wird eine Web-Applikation im Web-Browser des Geräts angezeigt. Dabei kommen die gängigen Entwicklungssprachen für Webseiten zum Einsatz, wie HTML5, CSS und JavaScript [10]. Einer der größten Vorteile ist, dass diese Applikationen auf allen Plattformen lauffähig sind und somit für alle Betriebssysteme nur einmal entwickelt werden müssen. Dies hat aber Einschränkungen in der Performance zur Folge. Ebenfalls ist die Nutzung von Gerätefunktionen nicht möglich [11].

Webbasierte hybride Applikation Webbasierte hybride Applikationen sind reinen Web-Applikationen sehr ähnlich. Implementiert werden diese ebenfalls mit den gängigen Entwicklungssprachen für Webseiten, werden dann aber, anstatt im Browser, in nativen Basisapplikationen ausgeführt. Diese stellt den mobilen Applikationen weiteren Zugriff auf Gerätefunktionen zur Verfügung [11]. Dabei bleibt der Quellcode der Applikationen weitestgehend plattformunabhängig.

Gemischte hybride Applikation Bei dieser Art der Entwicklung können Teile der Applikationen sowohl webbasiert, als auch nativ programmiert werden. Die webbasierten

Teile der Applikationen werden bei der Erstellung in nativen Code der entsprechenden Plattform umgewandelt [11]. Der native Teil der Applikationen ist plattformspezifisch und muss deshalb für unterschiedliche Plattformen angepasst werden. Dadurch steigt zum einen der Aufwand bei der Implementierung und zum anderen der Wartungsaufwand, wenn im nativen Teil der Applikation Änderungen vorgenommen werden. Ein Vorteil dieser Entwicklungsart ist, dass externe Bibliotheken in webbasiertem, als auch in nativem Code verwendet werden können.

Native Applikation Native Applikationen werden explizit für Plattformen oder sogar Geräte entwickelt. Hierfür kommen die entsprechenden Programmiersprachen der jeweiligen Plattform zum Einsatz (z.B. Java für Android, Swift für iOS, Visual C++ bzw. Visual C# für Windows Mobile) [10]. Dabei stehen Applikationen direkte Zugriffe auf die Geräte-APIs zur Verfügung, wodurch alle Gerätefunktionen mit der bestmöglichen Performance genutzt werden können. Der Aufwand bei der Entwicklung ist im Vergleich aber am höchsten [11], speziell dann, wenn eine Applikation für mehrere Plattformen gleichzeitig erstellt werden soll.

Zusammenfassend werden in Tabelle 2.1 die Unterschiede der verschiedenen Entwicklungsarten miteinander verglichen.

Tabelle 2.1: Einige Vor- und Nachteile unterschiedlicher Entwicklungsarten für mobile Applikationen (nach [11])

	Web-Anwendung	**Hybrid-Anwendung (Web)**	**Hybrid-Anwendung (Gemischt)**	**Native Anwendung**
Ausgeführt in	Web-Browser	Web-Ansicht in einer nativen Basisanwendung	native Umgebung in einer nativen Basisanwendung	native Laufzeitumgebung
Programmiersprache	webbasiert	webbasiert	nativ und webbasiert	nativ
Code Portabilität	hoch	hoch	mittelmäßig	keine
Gerätefeatures	keine	wenige	viele	alle
Grafiken/Animationen	mittelmäßig	mittelmäßig	gut	gut
externe Bibliotheken	JavaScript	JavaScript	JavaScript und native Bibliotheken	native Bibliotheken
User Experience	niedrig	niedrig	mittelmäßig	hoch
Performance	niedrig	niedrig	mittelmäßig	hoch

Mobile Applikationen können auf unterschiedlichste Arten entwickelt werden, welche verschiedene Vor- und Nachteile mit sich bringen. Schlussendlich entscheiden aber die

Anforderungen einer mobilen Applikation, welche Methode zur Entwicklung angewendet wird. Die Anforderungen, die der KINDEX-Fragebogen an mobile Applikationen stellt, waren ausschlaggebend für die Wahl der Entwicklungsart.

3

Verwandte Arbeiten

In diesem Kapitel werden zwei mobile Applikationen analysiert, die den KINDEX bereits als mobile Applikation umsetzen. In der Diplomarbeit [5] wurde der KINDEX-Fragebogen für das Apple iPad, in der Bachelorarbeit [12] hingegen für das Android-Betriebssystem konzipiert und realisiert. Im Folgenden wird der Funktionsumfang dieser mobilen Anwendungen untersucht und die Anforderungen an das eigene Projekt von den bereits existierenden mobilen Anwendungen abgeleitet, gegebenenfalls reduziert oder erweitert.

3.1 KINDEX für die iOS Plattform

Die Umsetzung des KINDEX-Fragebogens für das iPad war die erste mobilen Applikation, die zu diesem Thema realisiert wurde [5]. Der Autor stand in engem Kontakt mit Psychologen der Universität Konstanz, um die Anforderungen für die mobile Applikation detailliert zu erheben.

Realisiert wurde diese mobile Applikation mit der *Rapid Prototyping* Entwicklungsmethode, da zu Beginn der Implementierung noch nicht alle Anforderungen bekannt waren und viele Anforderungen erst im Laufe der Entwicklung bekannt wurden. Durch die gewählte Entwicklungsmethode war es möglich die Anforderungen während der Entwicklung zu präzisieren und das erhaltenes Feedback in einem iterativen Prozess in den finalen Prototypen einzubringen (siehe Abbildung 3.1).

In der finalen Version der mobilen Applikation kann der komplette Fragebogen auf dem iPad ausgefüllt werden. Pro Seite wird eine Frage angezeigt. Dabei ist es dem Benutzer nicht möglich, eine Frage zurückzuspringen. Des weiteren ist es ebenfalls nicht möglich,

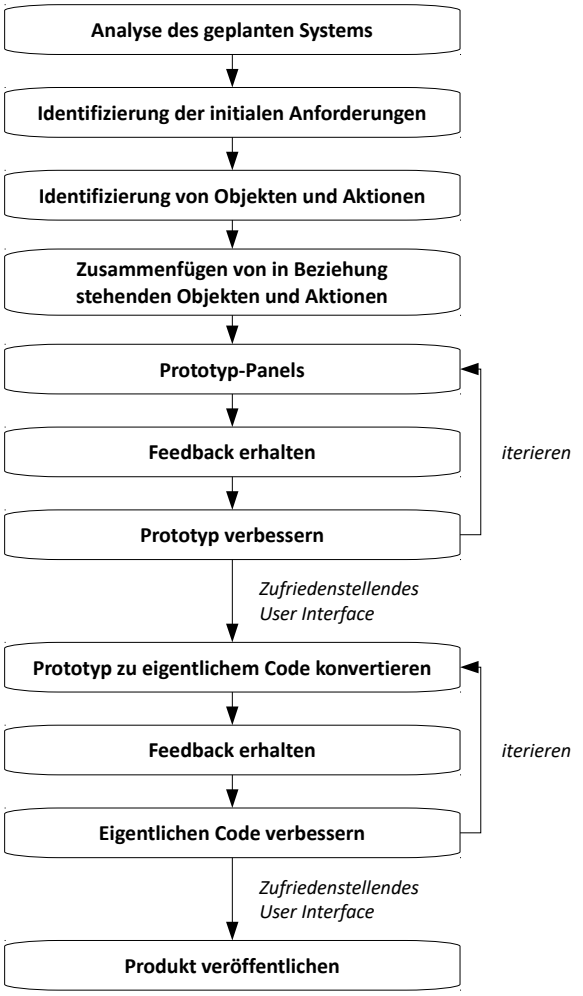


Abbildung 3.1: Rapid Prototyping (nach [13])

eine Frage zu überspringen oder nur teilweise zu beantworten. Ist dies der Fall, so wird der Benutzer durch eine Meldung darauf hingewiesen. Allerdings gibt es die Möglichkeit, den Fragebogen jederzeit zu beenden und somit die Befragung abzubrechen. Sobald eine Frage beantwortet wird, speichert die mobile Applikation zusätzlich die Zeit, die gebraucht wurde, um die Frage zu beantworten. Sind alle Fragen beantwortet, so kann der Arzt sofort eine Auswertung des Fragebogens einsehen, in der die berechneten Risikofaktoren aufgelistet sind. Ebenfalls werden entsprechende Gegenmaßnahmen dargestellt. Gleichzeitig wird der ausgefüllte Fragebogen anonymisiert auf dem Gerät

gespeichert. Administrative Angelegenheiten können in einem internen, durch ein Passwort geschützten Bereich, erledigt werden, auf den nur der Arzt Zugriff hat. Dort hat er unter anderem die Möglichkeit, den ausgefüllten Fragebogen erneut einzusehen. Ebenfalls kann er im geschützten Bereich personenbezogene Daten zur Arztpraxis oder seiner eigenen Person bearbeiten. Dort können mehrere Datensätze hinterlegt werden, sodass zum Beispiel im Falle einer Gemeinschaftspraxis der Arzt die auf ihn bezogenen Daten auswählen kann. Abhängig von dem gewählten Datensatz werden dem Arzt auch nur die Fragebögen angezeigt, die mit dem entsprechenden Datensatz verknüpft sind. Eine weitere Funktion, die sich im geschützten Bereich findet, ist der Export von ausgefüllten Fragebögen im Excel-Format. Alternativ zur deutschen Sprache, kann der Fragebogen in anderen Sprachen angezeigt werden. Zusätzlich soll am Anfang der Befragung eine Einverständniserklärung erscheinen, die den Benutzer über die Nutzung der erhobenen Daten aufklärt. Am Ende der Befragung soll ebenfalls ein Abschlusstext oder eine Danksagung erscheinen.

Die intuitive Bedienbarkeit der mobilen Applikation sollte ebenfalls gewährleistet sein, da das Alter der Schwangeren aber auch der Ärzte sehr stark variieren kann. So wurde ein sehr schlichtes und einfaches Design gewählt. Da die mobile Applikation für das iPad entwickelt wurde, kommt zur Übertragung der Daten von iPad zu Computer die Software iTunes zum Einsatz.

Die tatsächlich realisierte mobile Applikation setzt viele dieser Anforderungen um. Die Struktur des KINDEX-Fragebogen wurde fest in den Programmcode der mobilen Applikation eingebunden. Die Textelemente wurden in spezielle Dateien übertragen und so von der mobilen Applikation eingelesen. Alle Anforderungen zur Navigation und Zeiterfassung innerhalb des Fragebogens wurden umgesetzt. Dabei wird pro Seite des Fragebogens nur eine Frage angezeigt (siehe Abbildung 3.2a). Wurde der Fragebogen komplett ausgefüllt wird dieser ebenfalls als unverschlüsselte XML-Datei auf dem Gerät abgespeichert und kann mit iTunes auf einen Computer transferiert werden. Die Auswertung des Fragebogens erfolgt im Anschluss direkt auf dem iPad nach dem vorgegebenen Auswertungsalgorithmus. Die Risikofaktoren sowie deren Gegenmaßnahmen werden dem Arzt im geschützten Bereich angezeigt (siehe Abbildung 3.2b). Dieser ist durch ein Passwort geschützt und es gibt dort die Möglichkeit, personenbezogene Daten

der Arztpraxis einzugeben (siehe Abbildung 3.2c). Die Benutzerverwaltung, der Excel-Export, die Verschlüsselung der Daten als auch die Mehrsprachigkeit des Fragebogens wurden in diesem Prototypen nicht umgesetzt. Nichts desto trotz bleibt dadurch die Kernfunktionalität des Prototyps erhalten, die für die Tests und Evaluierung der mobilen Anwendung in den Arztpraxen ausreichend waren.

(a) KINDEX Frage 10 (b) Fragebogenauswertung (c) Dateneingabe

Abbildung 3.2: Screenshots des Prototypen der mobilen Applikation

In dieser Diplomarbeit wurden detailliert Anforderungen für die Umsetzung des KINDEX als mobile Applikation erhoben, die durch die Zusammenarbeit mit Ärzten, Hebammen und den Psychologen der Universität Konstanz nach und nach verfeinert wurden. Diese bieten eine Basis für die im Rahmen dieser Arbeit entwickelten mobilen Applikation, für die viele Anforderungen abgeleitet werden können. Durch Tests und Feedback der Benutzer wurde belegt, dass die Umsetzung des KINDEX-Fragebogen großteils akzeptiert wurde, aber auch, dass es an manchen Stellen noch Verbesserungen bedarf. Dies fließt ebenfalls in die eigene Anforderungsanalyse mit ein.

3.2 KINDEX für die Android Plattform

Die Autorin in [12] arbeitete bei ihrer Bachelorarbeit bei der Umsetzung des KINDEX-Fragebogens als mobile Applikation auf Basis des Android Betriebssystems. Dabei wurde die Applikation für Tablets, also Geräte mit großem Display, konzipiert und realisiert.

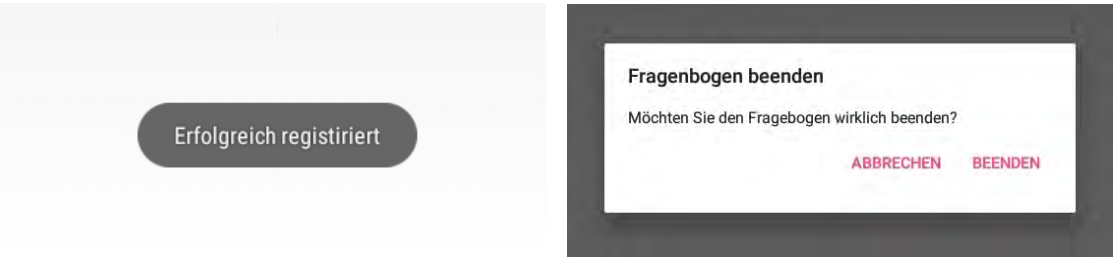
Die funktionalen Anforderungen, die an die finale Version der mobilen Applikation gestellt werden, wurden in 2 Bereiche geteilt. Den öffentlichen Bereich, zu dem jeder Benutzer freien Zugang hat, sowie den internen Bereich, der nur für die Ärzte zugängig ist.

Der öffentliche Bereich soll es ermöglichen, dass sich in erster Linie Ärzte, aber auch Patienten im System registrieren können. Dazu werden personenbezogene Daten abgefragt, die in einem Formular eingetragen werden. Dadurch wird den Ärzten der spätere Zugang zum internen Bereich ermöglicht. Die mobile Applikation startet mit der Fragebogenauswahl, in welcher Informationen gesammelt werden, die am Ende bei der Speicherung des Fragebogens dem Patienten sowie dem behandelnden Arzt zugeordnet werden. In einer Sprachauswahl wird der Fragebogen entweder in Deutsch oder Englisch geladen. Der digitale Fragebogen selbst beinhaltet alle Fragen des papierbasierten KINDEX-Fragebogen. Dabei wird dem Benutzer die Möglichkeit gegeben, Fragen zu überspringen oder nur zum Teil auszufüllen. Eine Warnmeldung soll versehentliches Überspringen einer Frage verhindern, denn das Zurückspringen zu einer bereits beantworteten Frage ist nicht möglich. Es ist aber möglich, die komplette Befragung abzubrechen. Sollte dieser Fall eintreten, werden keine Daten gespeichert. Andernfalls wird der ausgefüllte Fragebogen am Ende der Befragung nach einer Bestätigung des Benutzers auf dem Gerät abgespeichert, sodass nur noch der behandelnde Arzt Zugriff auf die Daten hat. Zusätzlich wird im öffentlichen Bereich ein Kontaktformular zur Verfügung gestellt, damit die Benutzer den Entwickler bei Fragen, Anregungen oder Empfehlungen kontaktieren können.

Im internen Bereich, zu dem nur registrierte Benutzer Zugang haben, erhalten diese, je nach Rolle, Zugang zu Patientendaten, deren ausgefüllte Fragebögen sowie persönlichen Daten zur Person oder zur behandelnden Praxis. Um die ärztliche Schweigepflicht zu wahren, werden einem Arzt in einer alphabetischen Liste alle ausgefüllten Fragebö-

gen der Patienten angezeigt, für die er der behandelnde Arzt ist. Die Auswertung erfolgt nach einem extern spezifizierten Algorithmus, der entsprechend umgesetzt wurde. Des weiteren besteht die Möglichkeit, dass ein Benutzer seine persönliche Daten ändern kann, sofern er sich im internen Bereich angemeldet hat.

Weiterhin wurden Anforderungen beschrieben, welche die Qualität des Systems und dessen Benutzbarkeit genauer spezifizieren. So werden die technischen Anforderungen an die mobile Applikation gestellt, die ein Tablet mit mindestens 7 Zoll Bildschirmdiagonale und mindestens Android Version 4.4 voraussetzen. Besonderen Wert wird auch auf die intuitive und dadurch einfache Bedienbarkeit der mobilen Applikation gelegt. Auf Grund dessen ist die Benutzeroberfläche schlicht gehalten. Es werden beispielsweise keine Grafiken verwendet und die Gestaltung der Benutzeroberfläche orientiert sich an gängigen Standards des Material Designs von Google [14]. Außerdem soll der Benutzer zu jeder Zeit wissen, in welchem Zustand sich das System befindet, welches durch sogenannte Toasts (siehe Abbildung 3.3a) oder Pop-Ups (siehe Abbildung 3.3b) vermittelt wird. Zuletzt spielt die Zuverlässigkeit der mobilen Applikation eine wichtige Rolle. So ist diese zum Beispiel auch ohne Internetverbindung in vollem Umfang nutzbar.

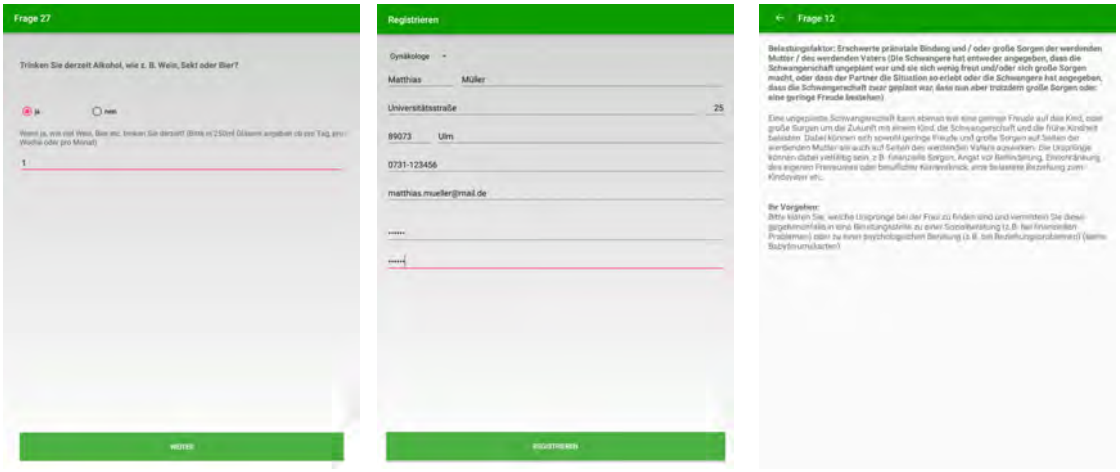


(a) Toast (b) Pop-Up

Abbildung 3.3: Vermittlung des aktuellen Systemzustands

In der letztendlich realisierten mobilen Applikation wurden alle erhobenen Anforderungen umgesetzt, die an die Applikation gestellt wurden. Die Datenhaltung wurde zum Großteil mit JSON-Dateien realisiert, die Infotexte, Beschriftungen aber auch den KINDEX-Fragebogen selbst, sowie dessen Antworten beinhalten. Die Navigation durch den Fragebogen, die Auswahl der Sprache sowie die Fragebogenauswahl entspricht den

Vorgaben der Anforderungen. Dabei wurde speziell darauf geachtet, dass nicht zu viele Elemente auf dem Bildschirm angezeigt werden, um die kognitive Belastung gering zu halten (siehe Abbildung 3.4a). Fragebögen werden zu Beginn der Befragung einem im System registrierten Arzt zugeordnet und nach Ende der Befragung auf dem Gerät abgelegt. Das System unterstützt dabei mehrere Benutzer, die sich auf dem Gerät anmelden können. Eventuelle Risikofaktoren und deren Gegenmaßnahmen kann ein Arzt für zugeordnete Fragebögen im internen Bereich anhand des gegebenen Auswertungsalgorithmus direkt auf dem Tablet betrachten (siehe Abbildung 3.4b). Des weiteren bietet die mobile Applikation auch die Möglichkeit, persönliche Informationen im internen Bereich zu ändern (siehe Abbildung 3.4c).



(a) KINDEX Frage 27 (b) persönliche Daten (c) Fragebogenauswertung

Abbildung 3.4: Screenshots der mobilen Applikation

Die Anforderungen, die diese Bachelorarbeit an die Umsetzung des KINDEX als mobile Applikation stellt, sowie die tatsächliche Realisierung dieser mobilen Applikation, unterscheiden sich zum Teil von den Anforderungen aus dem vorgehenden Kapitel 3.1. So bietet diese mobile Applikation beispielsweise keinerlei Möglichkeit, den ausgefüllten Fragebögen zu exportieren. Im Gegenzug findet sich hier aber beispielsweise ein Mehrbenutzersystem, sowie eine Sprachwahl. Es finden sich aber auch sehr viele Gemeinsamkeiten, was am Beispiel des Fragebogenablaufs deutlich wird. Dadurch werden

die Probleme und Schwierigkeiten bei der Entwicklung des KINDEX-Fragebogens als mobile Applikation zum Teil ersichtlich, was einen Einfluss auf die Anforderungsanalyse für die eigene Applikation hat.

4

Anwendungsszenario

Dieses Kapitel beschreibt, wie die Befragung zum KINDEX - Mum Screen momentan abläuft und wie der gewünschte Soll-Zustand in Zukunft aussehen muss, damit die Abläufe effizienter gestaltet und die Durchlaufzeiten reduziert werden. Dadurch erschließt sich, welche zusätzlichen Anforderung an eine Umsetzung als mobile Applikation gestellt werden, um den Arbeitsaufwand der beteiligten Personen zu reduzieren. Ziel ist es, dass Hebammen und Gynäkologen den KINDEX - Mum Screen in ihren Alltag integrieren. Anhand von Diagrammen werden dazu die Abläufe aufgezeigt. Dabei werden jeder Aktivität im Diagramm die beteiligten Personen zugeordnet (siehe Abbildung 4.1).

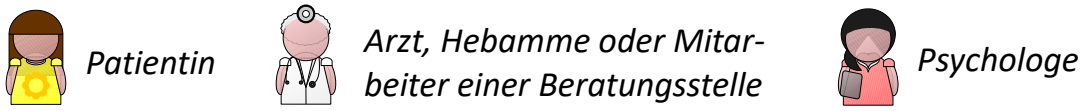


Abbildung 4.1: Legende für nachfolgende Diagramme

In Abbildung 4.2 wird der momentane Zustand in einem Aktivitätendiagramm beschrieben. Im ersten Schritt muss die Schwangere die Praxis ihres Gynäkologen oder eine Beratungsstelle besuchen. Möglicherweise wird sie auch von ihrer Hebamme besucht. Dies geschieht in der Regel im zuge einer routinemäßigen Vorsorgeuntersuchung. Im nächsten Schritt wird der zuvor bereits ausgedruckte Fragebogen vorbereitet. Dazu wird von der Hebamme oder dem Arzt das Datum des Interviews, die eigenen Kontaktdaten, sowie der Code oder Name der Schwangeren im Fragebogen eingetragen und anschließend an diese ausgehändigt. Im dritten Schritt startet die Bearbeitung des Fragebogens durch die Schwangere, die den Fragebogen selbstständig ausfüllt. Eventuell wird der

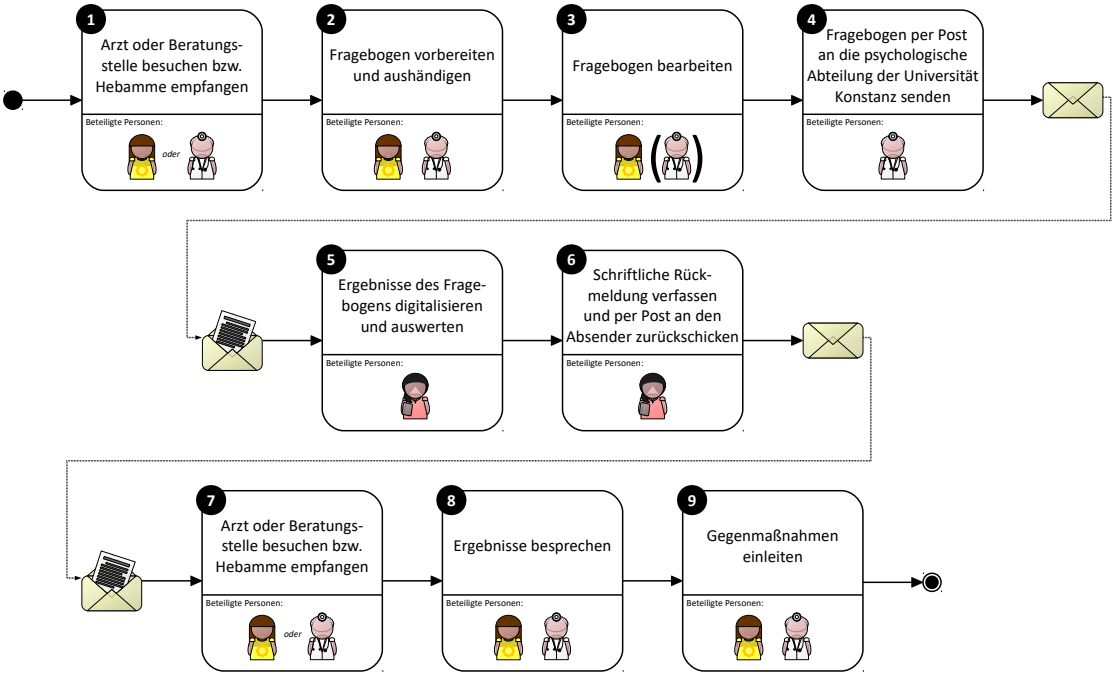


Abbildung 4.2: Momentaner Ablauf der Befragung zum KINDEX - Mum Screen

Fragebogen auch in Interviewform abgearbeitet. Bis der Fragebogen vollständig ausgefüllt ist, vergehen circa 20-30 Minuten [5]. Der ausgefüllte Fragebogen wird anschließend im vierten Schritt von der Hebamme oder dem Arzt in einen Briefumschlag gesteckt und per Post an die psychologische Abteilung der Universität Konstanz geschickt.

Nachdem der ausgefüllte Fragebogen dort angekommen ist, wird er im fünften Schritt von einem Psychologen weiter bearbeitet. Die Ergebnisse werden zunächst digitalisiert, sodass die weitere Verarbeitung der erhobenen Daten und die Auswertung der Ergebnisse elektronisch stattfinden kann. Die Daten können ebenfalls im gleichen Schritt der Statistik zugeführt werden. Im sechsten Schritt wird nach der Auswertung der Ergebnisse von einem Psychologen eine schriftliche Rückmeldung verfasst, in der mögliche Risikofaktoren sowie entsprechende Gegenmaßnahmen aufgeführt werden. Dieses Dokument wird wieder auf postalischem Wege an den Absender, also an die Praxis des Arztes, an die Beratungsstelle oder an die Hebamme, zurückgeschickt.

Im siebten Schritt muss sich die Schwangere in der Praxis des Arztes einfinden oder wird von ihrer Hebamme besucht, nachdem diese die Auswertung erhalten haben. In der Regel geschieht dies ebenfalls im Rahmen einer routinemäßigen Untersuchung. Die Ergebnisse des KINDEX - Mum Screen werden im achten Schritt mit der Schwangeren besprochen. Dabei werden der Schwangeren erklärt, welche Risikofaktoren vorliegen könnten. Im neunten Schritt werden entsprechend dieser Risikofaktoren letztendlich Gegenmaßnahmen eingeleitet oder Empfehlungen gegeben, um die Risiken zu mindern.

Es ist deutlich zu erkennen, dass zwischen der Beendigung der Befragung und dem Einleiten von Gegenmaßnahmen sehr viel Zeit vergeht. Dies liegt unter anderem daran, dass die Psychologen in Schritt 5 einen erheblichen Zeitaufwand bei der Digitalisierung der Daten haben. Des weiteren wird im sechsten Schritt jede schriftliche Rückmeldung einzeln verfasst, was ebenfalls viel Zeit in Anspruch nehmen kann. Zusätzlich wird der Ablauf durch den zweimaligen Versand per Post verzögert, wodurch mindestens zwei zusätzliche Tage mit eingerechnet werden müssen. Die größte Verzögerung entsteht aber dadurch, dass die Auswertung der Antworten nicht am selben Termin, an welchem die Befragung stattgefunden hat, erfolgen kann. Dementsprechend werden die Ergebnisse der Auswertung frühesten am darauffolgenden Termin mit der Schwangeren besprochen, wodurch bis zu sechs Wochen vergehen können [5].

Die Nutzung einer mobilen Applikation soll ermöglichen, dass die Auswertung des ausgefüllten Fragebogens direkt nach der Befragung stattfindet, sodass kein weiterer Termin mit der Schwangeren mehr notwendig ist, um die Ergebnisse zu besprechen. Zusätzlich soll so der benötigte Zeitaufwand der Psychologen ebenfalls reduziert werden. In Abbildung 4.3 wird der gewünschte Soll-Zustand des Ablaufs beschrieben.

Künftig wird im ersten Schritt die Schwangere zu einer routinemäßigen Vorsorgeuntersuchung die Praxis ihres Gynäkologen oder eine Beratungsstelle aufsuchen, beziehungsweise wird von ihrer Hebamme besucht. Im zweiten Schritt wird das mobile Gerät vorbereitet. Dazu wird die mobile Applikation gestartet und der passende Fragebogen ausgewählt. Ein vorheriges Ausdrucken des Fragebogens, sowie das Eintragen von Datum und Kontaktdaten entfällt, da dies von der Applikation übernommen wird und automatisch bestimmt wird, beziehungsweise nur einmalig nötig ist. Anschließend wird

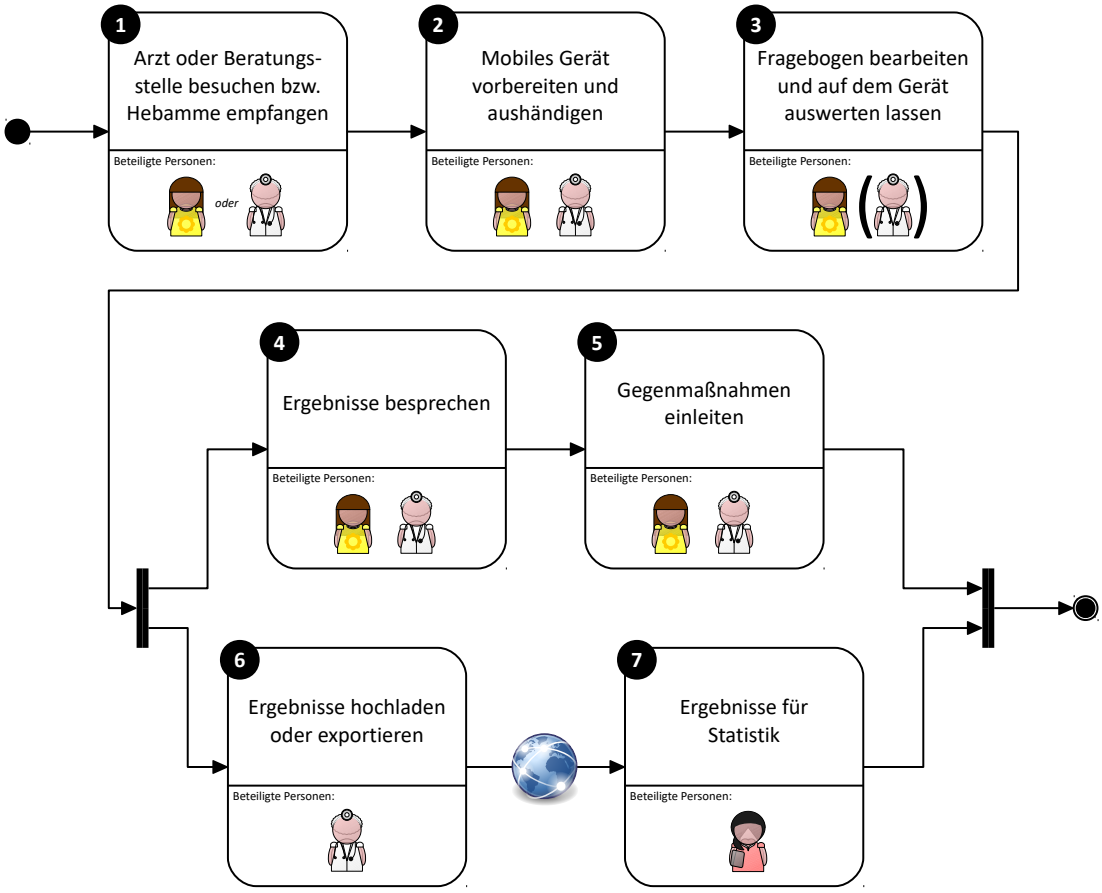


Abbildung 4.3: Gewünschter Ablauf der Befragung zum KINDEX - Mum Screen

das Gerät an die Schwangere übergeben und die Bearbeitung des Fragebogens startet damit als dritter Schritt. Direkt nach dem Ende der Befragung werden nun unmittelbar anhand eines gegebenen Algorithmus die Antworten analysiert und ausgewertet. Dies hat zur Folge, dass die Ärzte, Mitarbeiter der Beratungsstellen oder Hebammen nicht mehr auf eine Auswertung der externen Psychologen warten müssen.

So können schon in Schritt 4 die Ergebnisse besprochen und die möglichen Risikofaktoren erläutert werden. Das heißt auch, dass in Schritt 5, also noch während des selben Besuchs, bereits Gegenmaßnahmen eingeleitet oder Empfehlungen gegeben werden können.

Parallel zu Schritt 4 kann der Arzt im sechsten Schritt die Antworten des Fragebogens auf einen Server hochladen oder auf einen Computer exportieren. Da zum Beispiel das Hochladen auf einen Server nur ein paar Sekunden in Anspruch nimmt, könnte dies auch direkt nach der Befragung geschehen, sodass Psychologen sofort Zugriff auf die Daten haben. Die Daten werden diesen im siebten Schritt via Internet zur Verfügung gestellt. Dadurch fallen die Verzögerungen durch den Postversand weg. Des weiteren liegen die Daten bereits elektronisch vor, wodurch eine Aufbereitung der Daten von analoger in digitale Form nicht mehr notwendig ist. Dadurch können die Daten direkt der Statistik zugeführt werden. Das Verfassen von Rückmeldungen entfällt auch, da dies vollständig durch den gegebenen Auswertungsalgorithmus abgedeckt wird, der in der mobilen Applikation implementiert ist.

Der gewünschte Ablauf der Befragung zum KINDEX - Mum Screen würde für alle beteiligten Personen eine erhebliche Zeitersparnis mit sich bringen. Die benötigte Anzahl an Schritten wurde im Soll-Zustand von 9 auf 7 reduziert. Zusätzlich wäre es möglich, Aktivitäten parallel zu bearbeiten. Die in Kapitel 3 umgesetzten mobilen Applikationen setzen den gewünschten Ablauf zum Teil bereits um. Die Bedingungen, die dieser Ablauf an die eigene mobile Applikation stellt, definieren weitere Anforderungen, die ebenfalls in die Anforderungsanalyse mit einfließen.

5

Anforderungsanalyse

Diese Kapitel beschreibt die Anforderungen, die der KINDEX an die im Rahmen dieser Arbeit konzipierten und realisierten mobilen Anwendung stellt. Diese werden in funktionale und nichtfunktionale Anforderungen unterteilt (siehe Kapitel 5.1 und 5.2).

5.1 Funktionale Anforderungen

Die funktionalen Anforderungen beschreiben, welche Funktionen die mobile Applikation zwingend unterstützen muss.

AF 1 (Fragebogenauswahl): Beim Start der Anwendung soll eine Übersicht aller Fragebögen angezeigt werden, die zur Verfügung stehen. In dieser Übersicht soll dem Benutzer eine kurze Beschreibung sowie Metadaten (Name, Version) des Fragebogens angezeigt werden. Zusätzlich soll die Möglichkeit bestehen, den administrativen Bereich zu betreten (siehe **AF 8**). Weitere Informationen zu der mobilen Applikation, wie zum Beispiel die Version, sollen ebenfalls angezeigt werden können.

AF 2 (Fragebogenablauf): Der KINDEX soll in der mobilen Applikation ausgefüllt werden können. Dabei soll pro Seite nur eine Frage angezeigt werden. Teilfragen sollen nur angezeigt werden, wenn diese auch tatsächlich beantwortet werden sollen, sodass die kognitive Belastung des Benutzers möglichst gering gehalten wird. Es soll die Möglichkeit geben, Fragen zu überspringen oder nur zum Teil zu beantworten, da unter anderem emotional belastende Fragen gestellt werden. Ein Zurückspringen zu bereits beantworteten Fragen soll unterbunden werden,

sodass der Fragebogen nur in Vorwärtsrichtung ausgefüllt werden kann. Es soll außerdem zu jeder Zeit möglich sein, den Fragebogen abzubrechen, sodass die Antworten nicht gespeichert werden und kein zusätzlicher psychischer Druck auf den Benutzer ausgeübt wird. Damit dies nicht versehentlich geschieht, muss der Benutzer durch eine Warnmeldung darauf hingewiesen werden (siehe Abbildung 5.1).

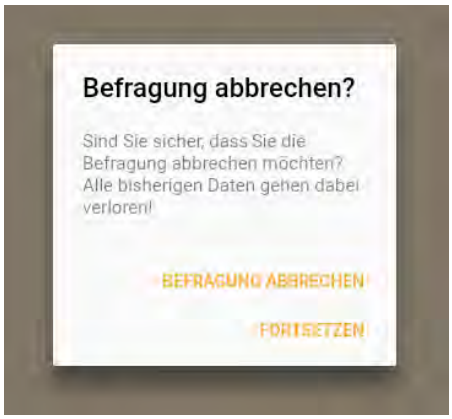


Abbildung 5.1: Warnhinweise beim Abbruch der Befragung

AF 3 (Mehrsprachigkeit): Der Fragebogen soll in mehreren Sprachen verfügbar sein. Dies hat zur Folge, dass alle Fragen, Beschriftungen und sonstige Texte der mobilen Applikation übersetzt und in der ausgewählten Sprache dargestellt werden müssen. So soll Benutzern, die kein oder nur ungenügend Deutsch sprechen, das Ausfüllen des Fragebogens ermöglicht werden.

AF 4 (Zeiterfassung): Zu jeder Antwort soll die Zeit gespeichert werden, die der Benutzer gebraucht hat, um diese Frage zu beantworten. Dies ist wichtig, da es ein Hinweis für den Arzt sein kann, dass die Frage für den Benutzer emotionale oder psychische Probleme hervorgerufen hat, wenn dieser für die Beantwortung überdurchschnittlich viel Zeit benötigt hat.

AF 5 (Speicherung der Antworten): Die Antworten des Fragebogens sollen am Ende der Befragung auf dem Gerät gespeichert werden, sodass ein späteres Betrachten der Antworten möglich ist. Dabei soll eine zufällige ID generiert werden, welche die Antworten eines Benutzers eindeutig identifiziert. Die ID darf keine Rückschlüsse

auf die Person ermöglichen. Zusätzlich soll es die Möglichkeit geben, dass diese ID durch eine Antwort zu einer Frage des Fragebogens ersetzt wird. Am Beispiel des KINDEX wird in Frage 3 nach dem Code einer Schwangeren gefragt, welcher sich aus dem ersten Buchstaben des Nachnamens und dem vollständigen Geburtsdatum zusammensetzt. Dieser Code soll die ID ersetzen können. Des weiteren soll Start- und Endzeitpunkt, sowie die verwendete Sprache gespeichert werden. Zu den Antworten der Fragen soll, die benötigte Zeit gespeichert werden (siehe **AF 4**). Falls auf der aktuellen Seite bereits beantwortete Fragen geändert werden, soll sowohl die aktualisierte als auch die ursprüngliche Antwort festgehalten werden, sodass bei der Auswertung geänderte Antworten eingesehen werden können. Diese Funktion kann für den Arzt ebenfalls ein Indiz für mögliche emotionale oder psychische Probleme des Benutzers liefern. Aus Gründen von Datenschutz und ärztlicher Schweigepflicht soll eine Zuordnung der Daten zum behandelnden Arzt erfolgen.

AF 6 (Auswertung der Antworten): Die Auswertung der Antworten soll direkt nach Beendigung der Befragung erfolgen. Dabei sollen die Antworten durch einen gegebene Algorithmus analysiert und bewertet werden (siehe **AF 13**).

AF 7 (Informationstexte): Zu Beginn einer Befragung soll es möglich sein, einen Informationstext anzuzeigen. Am Beispiel des KINDEX wird eine Einverständniserklärung angezeigt, die den Benutzer über die Verwendung der erhobenen Daten informiert. Am Ende eines Fragebogens soll ebenfalls ein Abschlusstext beziehungsweise eine Danksagung angezeigt werden, welche persönliche Daten enthalten kann (siehe **AF 12**).

AF 8 (Administrativer Bereich): Die mobile Applikation soll über einen administrativen Bereich verfügen. Dieser muss durch ein Passwort geschützt sein, damit nicht jeder Benutzer Zugriff darauf hat. Im administrativen Bereich finden sich Einstellungen um die mobile Applikation zu konfigurieren, wie zum Beispiel die Adresse des Servers, von welchem die Fragebögen bezogen werden. Des weiteren soll hier die Möglichkeit geboten werden, das Zugangspasswort zu ändern oder die Applikation in den Auslieferungszustand zurückzusetzen.

AF 9 (Übersicht über ausgefüllte Fragebögen): Im administrativen Bereich sollen die auf dem Gerät gespeicherten Fragebögen aufgelistet werden. Dabei soll erkenntlich sein, ob die Antworten eines Fragebogens bereits exportiert wurden.

AF 10 (Datenexport): Die mobile Applikation soll eine Schnittstelle zum Export der beantworteten Fragebögen bieten. Die Daten sollen anonym zur Erhebung von Statistiken zur Verfügung gestellt werden. Es soll sowohl der Export über eine USB-Schnittstelle als auch der Upload auf einen Server unterstützt werden.

AF 11 (Mehrbenutzerbetrieb): Damit die Zuordnung der Daten, wie in **AF 5** gefordert, funktioniert, müssen sich im administrativen Bereich unterschiedliche Benutzer anmelden können. Dadurch sollen nur die entsprechenden Fragebögen aufgelistet werden, die diesem konkreten Benutzer zugeordnet sind. Diese Funktion wird benötigt, da es im Falle von Gemeinschaftspraxen mehrere Ärzte gibt, welche die mobile Applikation nutzen.

AF 12 (Persönliche Daten): Es soll den Benutzern möglich sein, personenbezogene Daten zur Person oder zur Praxis in der mobilen Applikation zu hinterlegen.

AF 13 (Auswerungsalgorithmus): Der Auswertungsalgorithmus stammt von den Psychologen der Universität Konstanz, die diesen zur Auswertung der Fragebögen von Hand benutzt hatten. Dieser bedient sich hauptsächlich an einem einfachen Schema. Es wird überprüft ob eine Frage mit *Ja* oder *Nein* beantwortet wurde und je nach Antwort werden Risikofaktoren aufgelistet oder nicht. So wird dieses Vorgehen bei den Fragen 5, 6, 7, 10, 11, 12, 13, 15, 17, 18, 19, 20, 21, 22, 23, 24 und 25 angewendet. Des weiteren bedient sich die Auswertung an einfachen mathematischen Operationen. So wird bei Frage 4 überprüft, ob die Schwangere jünger als 21 Jahre ist. In Frage 8 wird zunächst der Wohnindex berechnet, welcher sich aus der Anzahl der Zimmer geteilt durch die Anzahl der darin wohnenden Personen, zusammensetzt. Anschließend wird überprüft, ob der resultierende Wert kleiner gleich 0,5 ist. Treffen diese Fälle zu, werden die entsprechenden Risikofaktoren aufgelistet. Es gibt aber auch Fragen, bei denen die Auswertung etwas komplexer ist. Dies ist bei bei Frage 14 der Fall. Hier sollen die Risikofaktoren angezeigt werden, wenn die Fragen 14b oder 14d im Bereich von 0 bis 3

oder die Fragen 14c oder 14e im Bereich von 7-10 liegen. Eine weitere komplexe Auswertung ist bei Frage 16 nötig. Hier werden den möglichen Antworten skalierte Werte zugewiesen. So sind die Fragen 16a und 16d von 0 bis 4 und die Fragen 16b und 16c invers von 4 bis 0 skaliert. Die entsprechenden Werte der Antworten werden dann aufsummiert. Liegt dieser Wert bei 12 oder mehr, dann sollen die Risikofaktoren angezeigt werden. Alle übrigen Fragen haben bei der Auswertung keine Relevanz, wie zum Beispiel Frage 9.

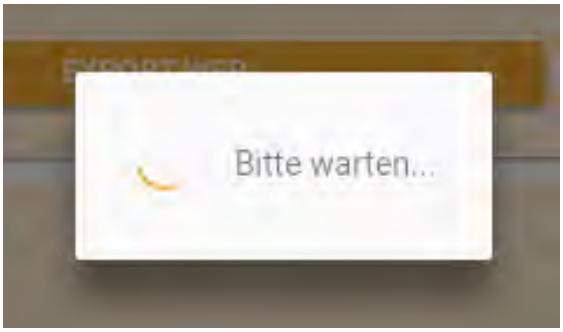
5.2 Nichtfunktionale Anforderungen

Die nichtfunktionale Anforderungen beschreiben, in welcher Qualität die in Abschnitt 5.1 geforderten funktionalen Anforderungen erbracht werden müssen.

NAF 1 (Usability): Die Benutzerfreundlichkeit der mobilen Applikation spielt eine essentielle Rolle. Das Grundverständnis für mobile Endgeräte der Benutzer kann sehr stark variieren. Am Beispiel des KINDEX wird dies deutlich, denn zum einen wird die mobile Applikation von Ärzten bedient, zum Anderen von den Schwangeren. Dabei stammen die Benutzer aus sämtlichen Altersschichten und könnten den Umgang mit mobilen Applikationen gewohnt sein, oder auch nicht. Deshalb sollte speziell auf die einfache und intuitive Bedienbarkeit der mobilen Applikation geachtet werden. Dazu gehört auch, dass der Benutzer erfährt, in welchem Zustand sich das System befindet. Rückmeldungen vom System über dessen Zustand sollen beispielsweise durch Pop-Ups (siehe Abbildung 5.2a) oder Ladebalken (siehe Abbildung 5.2b) erfolgen.

NAF 2 (Design): Beim Design soll hauptsächlich auf die Usability der mobilen Applikation geachtet werden. Dazu sollen die Benutzeroberflächen schlicht gehalten werden, sodass diese nicht überladen wirken und den Benutzer dadurch nicht überlasten. Das Design soll so gewählt werden, dass sich GUI-Elemente an den Richtlinien der zugrunde liegenden Betriebssysteme orientieren, wie beispielsweise *iOS Human Interface Guidelines* [15] oder *Material design* für Android [14].

(a) Fehlermeldung beim Export



(b) Ladevorgang

Abbildung 5.2: Hinweise über den Systemzustand

NAF 3 (Stabilität und Zuverlässigkeit): Ein fehlerfreier Betrieb der mobilen Applikation ist Voraussetzung für die Erfüllung der eigentlichen Funktion. Deshalb soll diese so gestaltet werden, dass Fehler erst gar nicht möglich sind, indem zum Beispiel in einer Frage nach dem Alter nur positive Ganzzahlen eingegeben werden können. Auftretende Fehler sollten behandelt werden, sodass der Benutzer sein Ziel trotzdem erreicht. Am Beispiel des KINDEX darf besonders die Speicherung und Auswertung der Fragebögen keine Fehler aufweisen, damit die Risikofaktoren vollständig und korrekt aufgezeigt werden können.

NAF 4 (Änderbarkeit und Wartbarkeit): Die Texte der Fragebögen, sowie deren Struktur soll einfach änderbar sein. Deshalb sollen die Fragebögen, die in der mobilen Applikation zur Verfügung stehen, nicht fest in die mobile Applikation eingebettet werden. Diese sollen in externe Dateien ausgelagert werden, welche von der mobilen Applikation eingelesen werden. Diese sollen dabei über das Internet von einem zentralen Server abgerufen und in der mobilen Applikation geladen werden. Dadurch können Änderungen auch von Novizen in diesen Dateien vorgenommen werden, ohne den Programmcode der Anwendung zu modifizieren. Eine Aktualisierung des Programmcodes bei einer Modifizierung eines Fragebogens ist dadurch nicht notwendig. Die mobile Applikation muss aufgrund dieser Anforderung gängige Elemente eines Fragebogens dynamisch darstellen können. Diese beinhalten:

- *Ja/Nein*-Fragen
- *Freitext*-Fragen

- *Single-Choice*-Fragen
- *Multiple-Choice*-Fragen
- *Skalen*-Fragen
- *Datums*-Fragen
- *Textpassagen* und *Überschriften*
- *Medien* (Bild, Audio, Video)

NAF 5 (Anforderungen an die Hardware): Die mobile Applikation soll auf möglichst vielen Tablets zum Einsatz kommen können. Zusammen hatten Android-Tablets und iPads im November 2016 einen Marktanteil von 99,04% an der Internetnutzung durch Tablets [16], weshalb die mobile Applikation für diese Plattformen entwickelt werden soll. Dabei sollen iPads ab iOS 8 und Android-Tablets ab Version 4.4 unterstützt werden.

6

Konzeption und Architektur

In diesem Kapitel wird die Konzeption und Architektur des Systems beschrieben. Dabei wird das System in einzelne Komponenten zerlegt, welche im weiteren Verlauf genauer beschrieben werden. Des weiteren wird erläutert, in welchem Zusammenhang einzelne Komponenten zueinander stehen und wie die Kommunikation zwischen diesen abläuft. Zusätzlich werden solche Aspekte behandelt, die sich nicht direkt auf die eigentliche Entwicklung der mobilen Applikation fokussieren, aber für deren tatsächlichen Betrieb notwendig sind. So wird auch beschrieben, welche Voraussetzungen an den Server gestellt werden und wie dessen Kommunikation mit der mobilen Applikation aussieht.

6.1 Gesamtsystemkonzept

Das Gesamtsystem wurde in drei Komponenten unterteilt. In Abbildung 6.1 wird dieses Konzept bildlich dargestellt. Der größte Teil ist die mobile Applikation, welche die Hauptkomponente des Systems darstellt. Diese wurde weitestgehend nach dem Model-View-Controller Muster konzipiert. Die Datenobjekte, die von der mobilen Applikation benötigt werden, stellen dabei das Model dar. Diese enthalten die Daten und Struktur der darzustellenden sowie eingelesenen Objekte. Die Präsentationsschicht wird durch die Elementen der *View*-Teilkomponente beschrieben. Dabei wurde die *View* in *Seiten* und *Komponenten* unterteilt. Die *Seiten* enthalten, abhängig von den Daten, unterschiedliche *Komponenten* (siehe Abschnitt 6.5). Die *View*-Komponenten nehmen auch die Eingaben des Benutzers entgegen, welche von den *Controller*-Elementen verarbeitet werden. Diese wurden, abhängig von ihren Funktionen, in separate Einheiten unterteilt. Der `GlobalService`-Controller ist dabei für die Datenkoordination innerhalb der

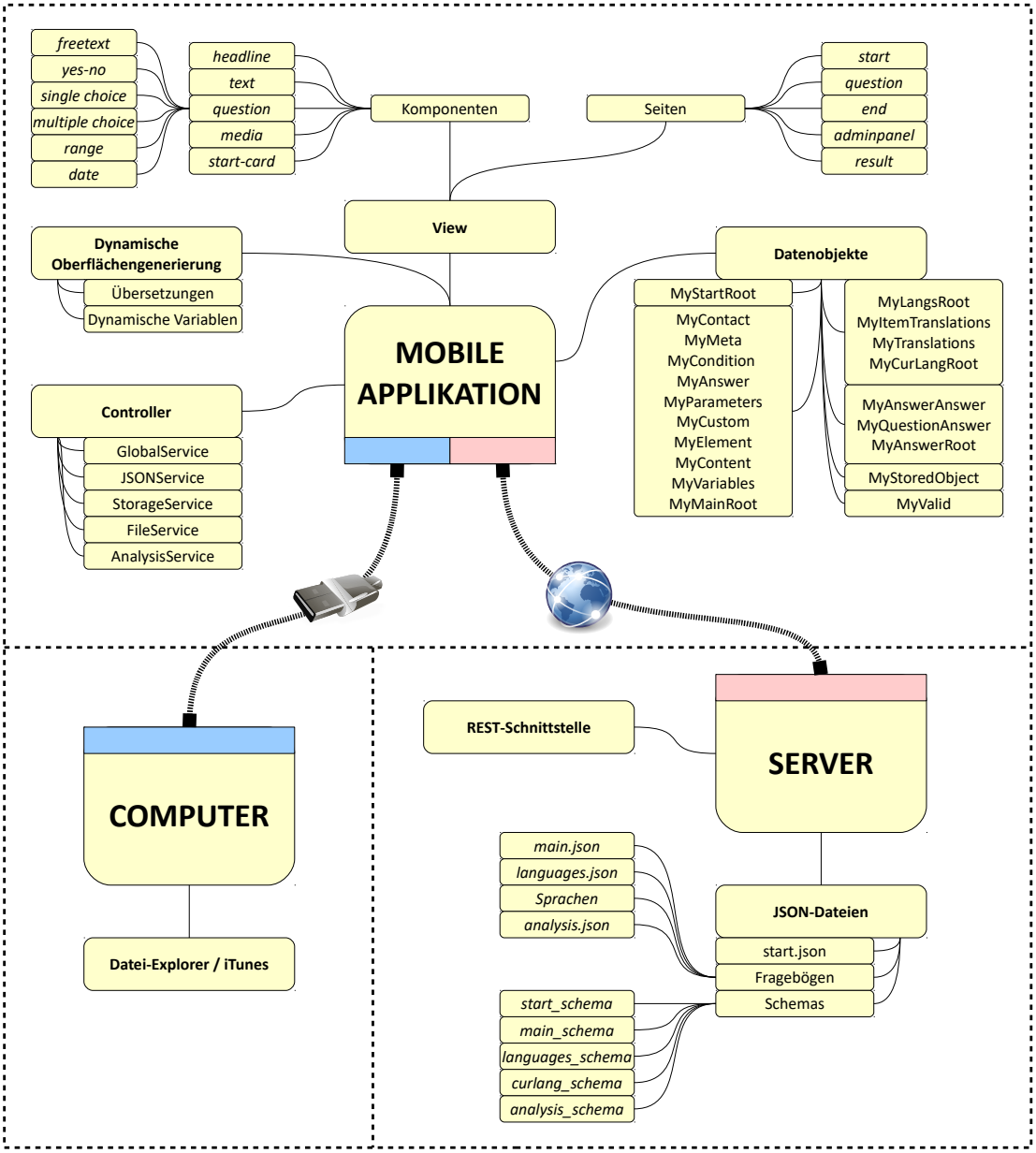


Abbildung 6.1: Konzeption des Gesamtsystems

mobilen Applikation verantwortlich und stellt der *View* Funktionen zur Verfügung. Der `JSONService`-Controller ist für die Kommunikation mit dem Server zuständig (siehe Abschnitt 6.4). Der `StorageService`-Controller wird für die Datenspeicherung auf dem Gerät verwendet (siehe Abschnitt 6.6). Der `FileService`-Controller beinhaltet Methoden zum Export via USB-Schnittstelle und der `AnalysisService`-Controller ist für die Auswertung ausgefüllter Fragebögen zuständig. Aufgrund der dynamischen Oberflächengenerierung, welche wegen der Mehrsprachigkeit und dem Einsatz von dynamischen Variablen, benötigt wird, muss teilweise von der Model-View-Controller Architektur abgewichen werden.

Die mobile Applikation kommuniziert über das Internet mit der Serverkomponente, welche der mobilen Applikation sowohl Daten zur Verfügung stellt als auch Daten entgegennimmt. Dabei bietet der Server eine REST-konforme Schnittstelle. Die Daten werden aktiv vom Client, durch eine der Ressource zugeordneten URL, angefordert oder geändert. Außerdem werden auf dem Server keine Zustände der mobilen Applikation gespeichert, was bedeutet, dass die Anfragen an den Server in sich geschlossen sind und dadurch alle benötigten Informationen für eine korrekte Antwort des Servers enthalten müssen. Dabei wird das HTTP-Protokoll und dessen GET- und POST-Methoden verwendet (siehe Abbildung 6.2). Daraus resultierend ist die Schnittstelle so definiert, dass Server- und Clientlogik komplett unabhängig voneinander agieren. Die Daten, die von der mobilen Applikation abgerufen werden, liegen auf dem Server als JSON-Dateien vor (siehe Abschnitt 6.2).

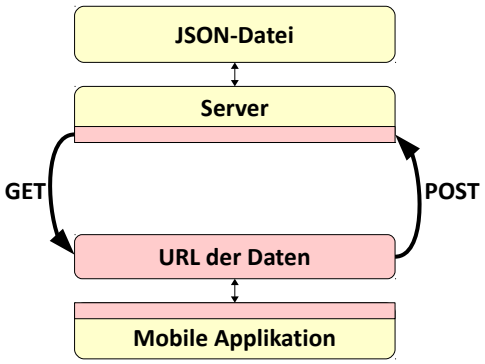


Abbildung 6.2: REST-konforme Schnittstelle zwischen mobiler Applikation und Server

Die übersichtlichste Komponente des Systems ist die Computerkomponente. Diese kommuniziert durch eine USB-Verbindung mit der mobilen Applikation und wird nur zum Export von ausgefüllten Fragebögen verwendet. Wird dazu ein iOS basiertes Gerät verwendet, so muss auf dem Computer iTunes installiert sein, um die Daten übertragen zu können. Bei Android hingegen wird das Gerät als Wechseldatenträger erkannt, bei dem der Benutzer Zugriff auf das Dateisystem hat und so die Dateien einfach kopieren kann.

6.2 Digitalisierung der Fragebögen

Fragebögen müssen in einem Format vorliegen, das sowohl von Mensch und Maschine einfach erstellt und gelesen werden kann. Dazu wurde die JavaScript Object Notation (kurz JSON) gewählt, welches diese Bedingung erfüllt. Das Format besteht dabei aus Key/Value-Paaren, welche zu Objekten zusammengefasst werden. Dabei können Objekte ebenfalls weitere Objekte enthalten. Eine weitere Struktur, die in JSON verwendet wird, sind geordnete Listen von Werten (Arrays). Der einem Key zugeordneten Wert kann schlussendlich einen String, einen numerischen Wert, einen boolschen Wert, den `null`-Wert, aber auch ein Objekt oder ein Array enthalten [17].

Bei der Digitalisierung eines Fragebogens wird dieser in mehrere JSON-Dateien aufgeteilt, sodass die eigentliche Struktur von der Sprache getrennt wird. Verfügt der Fragebogen über einen Auswertungsalgorithmus, so wird dieser ebenfalls in einer separaten Datei persistiert. Damit wird erreicht, dass zum Beispiel die Sprache eines Fragebogens durch eine neue Datei erweitert werden kann, ohne dass die Datei mit der strukturellen Beschreibung geändert werden muss. Infolge dieser Dateistrukturierung müssen die JSON-Dateien unter anderem Referenzen auf die Dateien enthalten, die insgesamt benötigt werden. Aufgrund der REST-Architektur des Servers muss dazu lediglich der Pfad der Dateien in den JSON-Dateien vermerkt werden. Die Strukturierung der Dateien, sowie die benötigten Verweise in den Dateien, wird in Abbildung 6.3 bildlich dargestellt.

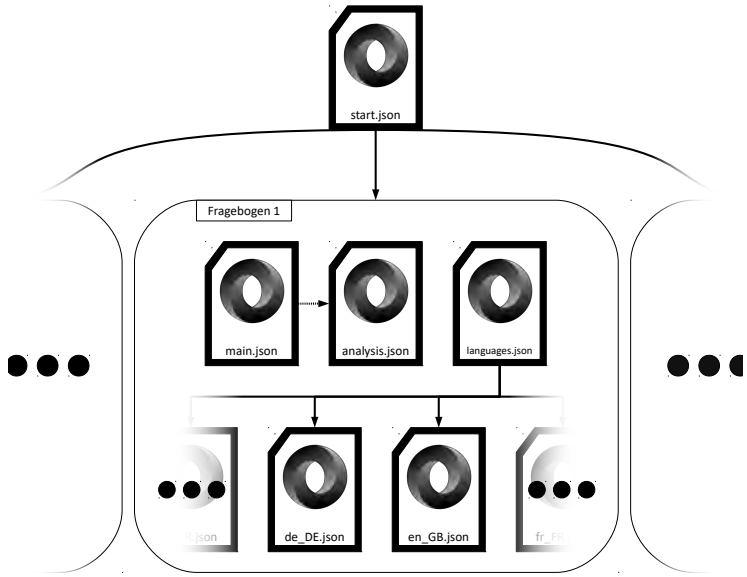


Abbildung 6.3: Referenzen der JSON-Dateien

Da das System mehrere Fragebögen unterstützt, wird zu Beginn eine Übersichtsdatei (`start.json`) eingelesen, in der alle verfügbaren Fragebögen aufgelistet sind. Diese besteht nur aus einem Array in der sich pro Fragebogen ein Objekt befindet, das dessen Namen (`name`) sowie den relativen Pfad (`path`) zu den weiteren Dateien angibt (siehe Listing 6.1).

```
1 [
2    {"name":"Kindex", "path":"kindex"},
3    {"name":"ComponentTest", "path":"componenttest"}
4 ]
```

Listing 6.1: `start.json`

Im Verzeichnis (`path`) finden sich die JSON-Dateien, die zu einem einzelnen Fragebogen zugeordnet werden. In Abbildung 6.4 wird das Schema der `main.json`-Datei grafisch verdeutlicht.

Der strukturelle Aufbau des Fragebogens wird in dieser Datei beschrieben (`content`). Zusätzlich befinden sich hier Metadaten (`meta`) zum Fragebogen, sowie Einträge für die dynamischen Variablen (`variables`) (siehe Listing 6.2).

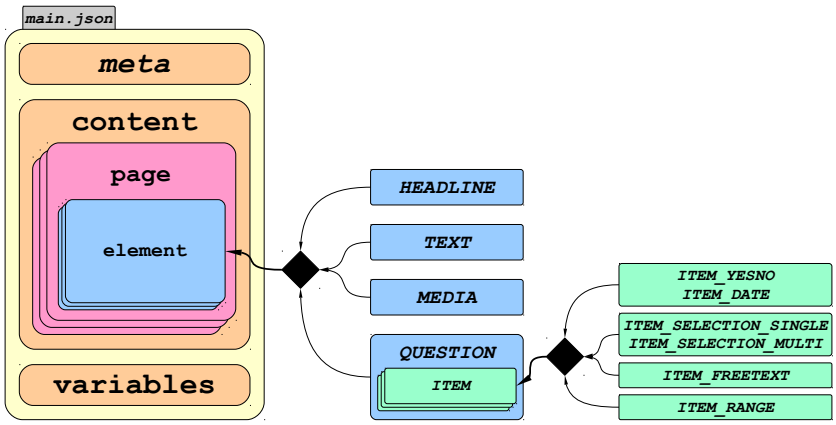


Abbildung 6.4: Grafische Darstellung des Aufbaus der `main.json`

```
{
   "meta":{
      "title":"KINDEX - Mum Screen",
      "picture":"kindex600x300.png",
      "version":"1.0",
      "info":"__info__",
      "contact":{
         "author":"Fabian Fischer",
         "email":"fabian-1.fischer@uni-ulm.de",
         "url":"www.uni-ulm.de/"
      },
      "header":"__header__",
      "footer":"__footer__",
      "note":"",
      "identifier":"p0_code",
      "analysis":true
   },
   "content":[ ... ],
   "variables":{"name":"q1_i1",...}
}
```

Listing 6.2: `main.json` des KINDEX

Die JSON-Datei beinhaltet drei Objekte. Das `meta`-Objekt enthält zusätzliche Daten zum Fragebogen, welche dazu dienen, den Fragebogen zu identifizieren. Deren Verwendung wird in Kapitel 7 genauer erläutert. Das `variables`-Objekt ist für die Nutzung von

dynamischen Variablen notwendig (siehe Abschnitt 6.3). Das `content`-Array wurde ausgeblendet, da dieses sehr umfangreich ist. Dort ist die komplette Struktur des Fragebogens hinterlegt. Es besteht aus `page`-Objekten, welche die Seiten der Fragebögen repräsentieren (siehe Listing 6.3).

```
{
   "id":"p_0",
   "name":"Code",
   "elements":[...]
}
```

Listing 6.3: `main.json` - `page`-Objekt

Dabei darf das `page`-Objekt nicht mit der tatsächlichen Seite des Papierfragebogens verwechselt werden. Vielmehr wird dieses dazu verwendet, eine sinnvolle Gruppierung von weiteren untergeordneten Elementen zu erhalten, welche im `elements`-Array zu finden sind. Diese Objekte repräsentieren verschiedene Elemente des Fragebogens, welche in vier verschiedene Typen eingeteilt wurde. Der Typ `HEADLINE` wird für Überschriften verwendet (siehe Listing 6.4).

```
{
   "id":"h_1",
   "type":"HEADLINE",
   "custom":{
      "condition":[...],
      "headline":"__a_headline__"
   }
}
```

Listing 6.4: `main.json` - Element vom Typ `HEADLINE`

Analog dazu dient der Typ `TEXT` für Textpassagen (siehe Listing 6.5).

```
{
   "id":"t_1",
   "type":"TEXT",
   "custom":{
      "condition":[...],
      "text":"__a_text__"
   }
}
```

Listing 6.5: `main.json` - Element vom Typ `TEXT`

Der Typ `MEDIA` wird benutzt, um mediale Inhalte einzubinden (siehe Listing 6.6). Auch wenn auf Papierfragebögen nur Bilder angezeigt werden können, wird hier zusätzlich die Möglichkeit geboten, Audio- und Videomaterial einzubinden.

```
{
   "id":"m_1",
   "type":"MEDIA",
   "custom":{
      "condition":[...],
      "caption":"__a_caption__",
      "type":"video/mp4",
      "source":"www.example.com/vid.mp4",
      "parameters":{
         "width":"320",
         "height":"240",
         "autoplay":false
      }
   }
}
```

Listing 6.6: `main.json` - Element vom Typ `MEDIA`

Das letzte Element ist vom Typ `QUESTION` und wird für die eigentlichen Fragen verwendet (siehe Listing 6.7).

```
{
   "id":"q_1",
   "export_name":"nameforexport",
   "type":"QUESTION",
   "custom":{
      "condition":[...],
      "question":"__a_question__",
      "answers":[...]
   }
}
```

Listing 6.7: `main.json` - Element vom Typ `QUESTION`

Da es für Fragen unterschiedliche Antwortmöglichkeiten gibt, müssen diese ebenfalls in einer weiteren Struktur repräsentiert werden. Dafür ist das Feld `answers` vorgesehen, welches in einem Array Objekte vom Typ `ITEM` aufnimmt. Die sechs Fragetypen, die in **NAF4** genannt wurden, können zum Teil in der selben Struktur dargestellt werden.

Für *Ja/Nein*-Fragen und *Datums*-Fragen kann so das selbe `ITEM` benutzt werden (siehe Listing 6.8).

```
{
   "id":"i_1",
   "export_name":"nameforexport",
   "condition":[],
   "required":true,
   "type":"ITEM_YESNO",
   "text_key":"__a_item_key__"
}
```

Listing 6.8: `main.json` - Element vom Typ `ITEM_YESNO`

Dabei wird für eine *Datums*-Frage der Wert des Keys `type` durch `ITEM_DATE` ersetzt. Ein weiteres `ITEM` ist vom Typ `ITEM_FREETEXT`, welches die Eingabe von Text definiert (siehe Listing 6.9).

```
{
   "id":"i_2",
   "export_name":"nameforexport",
   "condition":[],
   "required":true,
   "type":"ITEM_FREETEXT",
   "text_key":"__a_item_key__",
   "inputformat":"TEXT"
}
```

Listing 6.9: `main.json` - Element vom Typ `ITEM_FREETEXT`

Die Art der Eingabe wird durch den Key `inputformat` bestimmt. Dieser kann die Werte `TEXT` für eine normale Textzeile, `INTEGER` für eine Ganzzahl, `FLOAT` für eine Fließkommazahl und `TEXT_MULTILINE` für mehrzeilige Texte annehmen. *Single-Choice*- und *Multiple-Choice*-Fragen nutzen ebenfalls das selbe `ITEM` (siehe Listing 6.10).

```
{
   "id":"i_3",
   "export_name":"nameforexport",
   "condition":[  ],
   "required":true,
   "type":"ITEM_SELECTION_SINGLE",
   "text_key":"__a_item_key__",
   "choices":["__option_a__","__option_b__",...]
}
```

Listing 6.10: `main.json` - Element vom Typ `ITEM_SELECTION_SINGLE`

Für eine *Multiple-Choice*-Frage muss dafür ebenfalls der Wert des Keys `type` durch `ITEM_SELECTION_MULTI` ersetzt werden. Die Antwortmöglichkeiten werden dabei als Strings in einem Array gespeichert, der dem Key `choices` zugeordnet ist. Die letzte Frage für die ein `ITEM` umgesetzt wurde, sind *Skalen*-Fragen (siehe Listing 6.11).

```
{
   "id":"i_4",
   "export_name":"nameforexport",
   "condition":[  ],
   "required":true,
   "type":"ITEM_RANGE",
   "text_key":"__a_item_key__",
   "range":[0,10],
   "pin":"true",
   "sliderlabel":"IONICON, MINMAX, NOTHING",
   "ionicons":["left","right"]
}
```

Listing 6.11: `main.json` - Element vom Typ `ITEM_RANGE`

Hier wird im `range`-Key der minimale und maximale Wert der Skala definiert. Die Bedeutung der restlichen Felder wird in Kapitel 7 genauer behandelt.

Durch den Aufbau der `main.json` nach diesem Schema kann die Struktur gängiger Fragebögen in einer digitalen Form repräsentiert werden. Dabei werden keine tatsächlichen Texte oder sonstige Beschriftungen gespeichert, sondern nur Referenzen auf bestimmte Schlüssel in einer anderen JSON-Dateien. Diese enthalten alle benötigten Texte und Beschriftungen in einer bestimmten Sprache. Die verfügbaren Sprachen und die dazugehörigen Dateien werden in der `languages.json`-Datei aufgelistet (siehe Listing 6.12).

```
[
   {"code":"de_DE", "name":"Deutsch", "uri":"de_DE.json"},
   {"code":"en_GB", "name":"English", "uri":"en_GB.json"}
]
```

Listing 6.12: `languages.json`

Die in `uri` angegebenen Dateien enthalten die eigentlichen Texte in der angegebenen Sprache, auf welche sich die Verweise aus der `main.json` beziehen (siehe Listing 6.13).

```
{
   "de_DE":{
      "info":"Konstanzer Index zur Erhebung von ...",
      "header":"KINDEX - Mum Screen",
      "footer":"",
      "yes":"Ja",
      "no":"Nein",
      "p1_headline_welcome":"Herzlich Willkommen {{name}}",
      "p1_text_welcome":"Herzlich Willkommen zum Konstanzer...",
      "p0_code":"Code (1. Buchstabe Nachname ...)",
      "p0_i1_code":{
         "label":"",
         "placeholder":"Hier Code eingeben...",
         "measure":""
      }
   }
}
```

Listing 6.13: Kurzer Auszug aus `de_DE.json` für den KINDEX

Durch diesen Aufbau lassen sich innerhalb der vorhandenen Sprachdateien einfach Änderungen vornehmen. Zusätzlich ist die Erweiterung eines Fragebogens um eine neue Sprache vergleichsweise einfach und kann jederzeit erfolgen, ohne Änderungen an der Struktur vornehmen zu müssen.

6.3 Mehrsprachigkeit und dynamische Variablen

Damit die mobile Applikation Fragebögen in verschiedenen Sprachen darstellen kann, werden die Texte unabhängig von der Struktur gespeichert. Durch diese Aufteilung ist es einfacher, die Texte dynamisch zu ersetzen beziehungsweise Sprachen im Betrieb zu wechseln. Statt des Textes wird nur eine Referenz auf einen Text in einer anderen Datei gespeichert. Dadurch beinhalten alle Objekte, denen ein Textelement zugeordnet ist, eine Referenz auf die Übersetzung in den Sprachdateien. Dabei wird eine solche

Referenz speziell markiert. Diese beginnt und endet mit einem doppelten Unterstrich (`__variable__`). So finden sich bereits in den Metadaten die ersten Referenzen für die Übersetzung (siehe Listing 6.2).

Analog dazu wird mit dynamischen Variablen verfahren. Diese werden verwendet, falls bereits gegebene Antworten in einem Textelement angezeigt werden sollen. Beispielsweise wird zu Beginn des Fragebogens der Name abgefragt. Dieser soll in der Begrüßung wieder angezeigt werden. Dazu werden in der Sprachdatei Platzhalter in die übersetzten Texte eingefügt. Diese beginnen mit zwei sich öffnenden geschweiften Klammern, gefolgt von einer beschreibenden Variable und enden mit zwei sich schließenden geschweiften Klammern (`{{variable}}`). Die Referenz auf die Antwort, welche die Variable ersetzen soll, wird im `variables`-Objekt der `main.json` eingetragen (siehe Listing 6.2).

Der `GlobalService`-Controller übernimmt dabei unter anderem die technische Umsetzung dieser Verfahren. Dabei läuft die Übersetzung, sowie das Ersetzen der dynamischen Variablen, im gleichen Schritt ab. Als erstes wird die gegebene Referenz in der gewählten Sprachdatei gesucht. Wird diese gefunden, wird der Text auf dynamische Variablen untersucht, die im nächsten Schritt durch die in den `variables` der `main.json` referenzierte Antwort ersetzt wird. Der resultierende String wird dann in den entsprechenden View-Elementen angezeigt.

6.4 Validierung der JSONs

Um einen zuverlässigen Betrieb der mobilen Applikation zu gewährleisten, dürfen in den eingelesenen JSON-Dateien keine Fehler vorhanden sein. Zum einen muss der Aufbau der Objekte so angeordnet sein, dass die Datei in sich ein valides JSON darstellt, zum anderen müssen Objekte in einer definierten Struktur vorliegen, die von der mobilen Applikation verwendet werden. Diese Bedingungen werden durch JSON-Schemas beschrieben (ähnlich zu XML-Schema). Ein JSON-Schema ist dabei selbst als valide JSON-Datei gespeichert, welche die Struktur einer anderen JSON-Datei beschreibt (siehe Listing 6.14) [18].

```
1  {
2     "$schema":"http://json-schema.org/draft-04/schema#",
3     "type":"array",
4     "items":{
5        "type":"object",
6        "properties":{
7           "name":{"type":"string"},
8           "path":{"type":"string"}
9        },
10       "required":["name", "path"],
11       "additionalProperties":false
12    }
13 }
```

Listing 6.14: JSON-Schema für die `start.json`-Datei

Der Aufbau der `start.json`-Datei lässt sich durch das Beispiel des zugehörigen Schemas gut veranschaulichen. In der dritten Zeile wird zunächst definiert, von welchem Typ das JSON selbst ist. In der *start.json* ist dies ein Array. Die Elemente des Array sind, wie in Zeile 5 angegeben, einfache Objekte. Diese Objekte enthalten die Variablen `name` und `path`, in denen je ein String gespeichert ist. Dies wird durch `properties` in Zeile 6 spezifiziert. In Zeile 10 werden in `required` die Elemente aufgezählt, die auf jeden Fall in dem Objekt vorhanden sein müssen. Zusätzlich wird das Hinzufügen von weiteren Werten durch `additionalProperties` in Zeile 11 verhindert.

In die mobile Applikation eingelesene JSON-Dateien können anhand der Strukturinformationen, welche die JSON-Schemas enthalten, validiert werden, indem der Aufbau der Datei mit dem Schema verglichen wird. Dabei wird sichergestellt, dass alle von der mobilen Applikation benötigten Elemente in der richtigen Struktur vorliegen. Diese liest JSON-Dateien im `JSONService`-Controller ein. Dabei wird zunächst die JSON-Datei mit den Daten und anschließend das dazugehörige Schema eingelesen. Im nächsten Schritt wird überprüft, ob es sich bei der Datei um ein valides JSON handelt, das dem Schema entspricht. Ist dies der Fall, werden die Daten der mobilen Applikation zur Verfügung gestellt und der Fragebogen kann gestartet werden.

6.5 Aufbau der GUI-Elemente

Alle von der mobilen Applikation benötigten Daten werden durch den `JSONService`-Controller zur Verfügung gestellt. Dabei werden die validen Daten der JSON-Dateien in die Datenobjekte der mobilen Applikation eingelesen (siehe Abschnitt 6.4). Die in der `start.json` verfügbaren Fragebögen werden zu Beginn angezeigt. Dazu werden die Informationen zu je einem Fragebogen in einer `start-card`-Komponente angezeigt, die zunächst die Metainformationen aus der `main.json` sowie alle verfügbaren Sprachen aus der `languages.json` auflistet. Der Aufbau der GUI-Elemente auf der `question`-Seite folgt der Struktur, welche in der `main.json` hinterlegt ist. Die Seite und deren Komponenten werden analog zu dieser dynamisch aufgebaut. Dabei bestehen die Komponenten dieser Seite aus `headline`, `text`, `media` und `question`, wobei die `question`-Komponente weitere Elemente zur Beantwortung einer Frage enthält. Diese Komponenten repräsentieren die `ITEM`-Objekte aus der `main.json` und stellen die geforderten Funktionen zur Eingabe von Daten zur Verfügung (siehe Abbildung 6.5). Die technische Umsetzung der Seiten und Komponenten wird in Kapitel 7 genauer beschrieben.

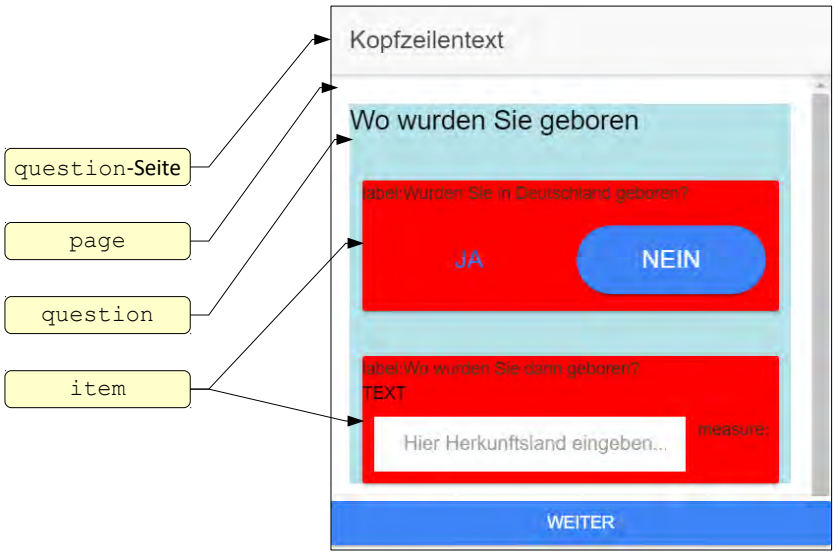


Abbildung 6.5: Screenshot aus einer frühen Entwicklungsphase, bei der die Strukturierung der einzelnen Komponenten ersichtlich wird

6.6 Persistieren der Ergebnisse

Die Persistierung der Ergebnisse erfolgt anhand der drei Ebenen des Fragebogens, die sich durch dessen Strukturierung in eine digitaler Form ergeben haben (siehe Abbildung 6.6).

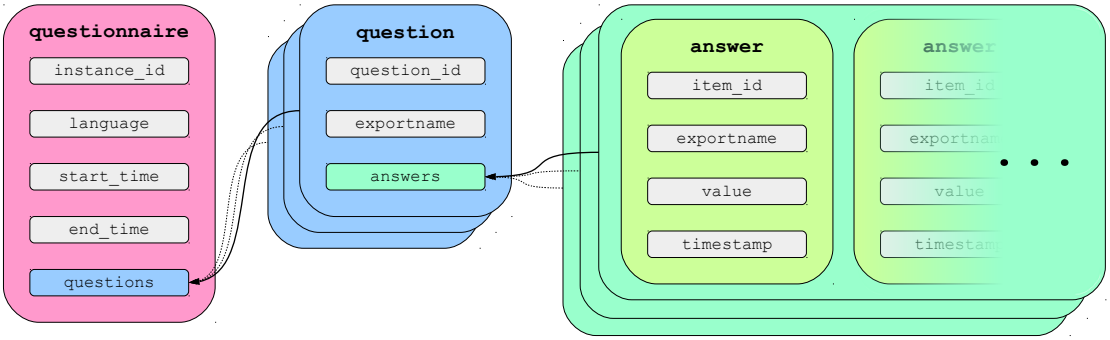


Abbildung 6.6: Struktur der gespeicherten Ergebnisse

Pro gegebener Antwort wird ein `answer`-Objekt erstellt und in einer Array-Struktur abgelegt. Dieses enthält die ID des Items, den Exportnamen, die Antwort auf die Frage und einen Zeitstempel. Wird die Antwort geändert, dann wird das bereits erstellte `answer`-Objekt nicht überschrieben. Stattdessen wird dem Array ein neues `answer`-Objekt hinzugefügt. Die resultierenden Arrays werden in das `answers`-Feld des `question`-Objekts gespeichert. Da zu einer Fragengruppierung mehrere Antworten zugeordnet werden können, besteht das Feld `answers` aus einem Array, der die `answer`-Arrays enthält. Weiterhin sind `question_id` und `exportname` im `question`-Objekt vermerkt. Dieses Objekt wird zusammen mit den selben Objekten anderer Frage als Array im `questions`-Feld des `questionnaire`-Objekts gespeichert. Zusätzlich wird eine Instanz-ID, die Sprache, sowie Start- und Endzeitpunkt gespeichert. Für die Speicherung dieser Daten auf dem Gerät ist der `StorageService`-Controller verantwortlich, der diese Daten am Ende einer Befragung auf dem Gerät ablegt und für spätere Zugriffe wieder in die mobile Applikation einlesen kann.

7

Implementierung

In diesem Kapitel wird zunächst das Ionic Framework vorgestellt, auf dem die im Rahmen dieser Arbeit entstandene mobile Applikation basiert. Anschließend werden dazu interessante Aspekte der Implementierung dargestellt, die aufzeigen, welche Teile des Frameworks hierfür benötigt wurden.

7.1 Ionic Framework

Das Ionic Framework ist ein Open-Source Projekt, mit dem webbasierte hybride mobile Applikationen entwickelt werden (siehe Kapitel 2.4). Hierfür werden die üblichen Webtechnologien (HTML5, JavaScript und CSS) verwendet. In der zweiten Version (Ionic2) kommt neben HTML5 und SASS (zuständig für die Erzeugung von CSS-Dateien [19]), TypeScript zum Einsatz, welches eine typisierte Obermenge von JavaScript darstellt [20]. Zusätzlich baut Ionic2 auf zwei weiteren frei verfügbaren Frameworks auf: Apache Cordova und Angular2 (siehe Abbildung 7.1).

Apache Cordova ist ebenfalls ein Framework für hybride mobile Applikationen. Auch hier findet die Entwicklung durch moderne Webtechnologien, wie HTML5, CSS und JavaScript statt. Dabei bietet Cordova eine plattformspezifische Basisiapplikation (Wrapper) in welcher die Webanwendung ausgeführt wird. Dieser Wrapper ermöglicht den Zugriff auf die nativen APIs eines Gerätes, sodass diese Schnittstellen in der Webanwendung zur Verfügung stehen. Bei der Entwicklung müssen dem Projekt für unterschiedliche Funktionen Plugins hinzugefügt werden, um deren APIs nutzen zu können. So gibt es beispielsweise Plugins für die Nutzung des Gerätespeichers oder

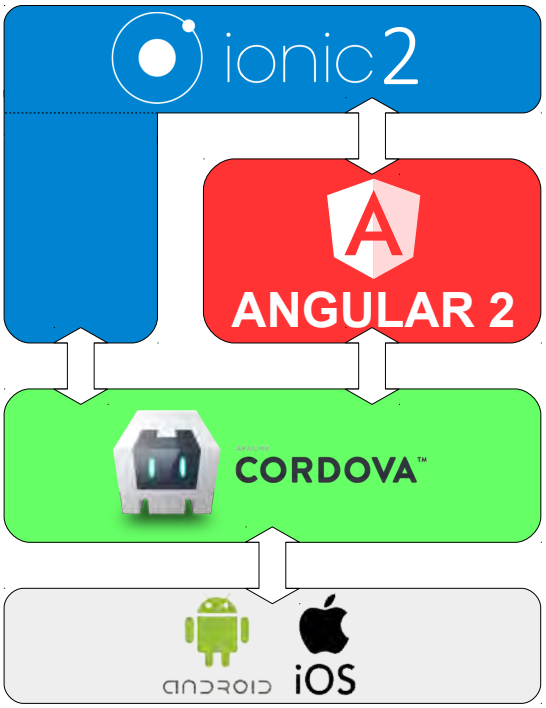


Abbildung 7.1: Aufbau des Ionic Frameworks

der Kamera. Cordova bietet allerdings keinerlei Unterstützung bei der Entwicklung des User-Interface, sondern nur die rein technische Möglichkeit, einer Webseiten-basierten Applikation Zugriff auf die Geräte-APIs zu ermöglichen [21].

Angular2 ist ein Framework, das auf die Entwicklung sogenannter Einzelseiten-Webanwendungen (Singel-Page Application) ausgelegt ist. Diese bestehen aus einer einzigen HTML-Seite, bei der Inhalte dynamisch nachgeladen werden. Hierzu erfolgt die Strukturierung in Modulen, welche für Darstellung der Benutzeroberfläche und die Einbindung von internen Funktionen aber auch externen Bibliotheken benötigt werden. Ein Modul setzt sich aus unterschiedlichen Elementen zusammen. Beispielsweise wird ein Modul zur Anzeige der Benutzeroberfläche durch eine TypeScript-Klasse und ein dazugehöriges HTML-Template definiert, welches mit speziellen Angular-Elementen erweitert wurde. Das HTML-Template enthält nur den strukturellen Aufbau der Seite. Deren Inhalt wird durch die TypeScript-Komponente verwaltet. Hierzu bietet Angular2

verschiedene Arten, den Inhalt mit den Elementen der Anzeige zu verknüpfen, unter anderem durch eine bidirektionale Anbindung. Dabei wird das Model durch Eingaben in der View geändert oder die View aktualisiert, wenn sich das Model geändert hat. Service-Elemente sind ein weiterer Teil eines Moduls. Deren Verwendung erstreckt sich von der Speicherung einfacher Variablen bis zur Bereitstellung komplexer Funktionen und Features [22].

Ionic2 konzentriert sich hauptsächlich auf die Gestaltung und die Interaktion mit den Elementen der Benutzerschnittstelle der mobilen Applikation. Ziel ist es, den Entwicklungsprozess des Frontends zu vereinfachen. Hierfür stellt Ionic2 eigene UI-Komponenten zur Verfügung, die das Look-and-Feel der Plattform, auf welcher die mobile Applikation ausgeführt wird, adaptieren. Zusätzlich enthält das Framework Icons, die ihr Aussehen ebenfalls der darunter liegenden Plattform anpassen, sogenannte Ionicons. Ein weiterer Bestandteil von Ionic2 ist Ionic Native. Dieser beinhaltet TypeScript-Wrapper, welche die Funktionen von Cordova Plugins in der Ionic-Applikation zur Verfügung stellen [23].

Für die mobile Applikation, welche im Rahmen dieser Arbeit entstanden ist, wurde Ionic2 zur Entwicklung der mobilen Applikation gewählt. Da dieses Framework sowohl Android- als auch iOS-Geräte unterstützt (siehe **NAF 5**), müssen keine separaten Applikationen geschrieben werden, wie es bei der nativen Entwicklung der Fall gewesen wäre. Ein weiterer Grund wird durch das Adaptieren des Look-and-Feel der darunterliegenden Plattform gegeben (siehe **NAF 2**). Zugriffe auf die Geräte-APIs stehen ebenfalls zur Verfügung (siehe **AF10**), was bei einer Web-Anwendung nicht möglich gewesen wäre. Da die Entwicklung einer webbasierten hybriden Anwendung mit dem Ionic Framework den benötigten Funktionsumfang abdecken kann, wurde die Option einer gemischten hybriden Applikation aufgrund der längeren Entwicklungszeit nicht gewählt.

7.2 Implementierung ausgewählter Komponenten

In diesem Abschnitt werden ausgewählte Komponenten der Implementierung vorgestellt. Dabei wird erklärt, welche Funktionen diese enthalten und wie sie mit Ionic2 und dessen Möglichkeiten umgesetzt wurden.

7.2.1 Service-Controller

Ein `Service` ist eine Controller-Komponente die global in der gesamten mobilen Applikation erreichbar ist. Diese enthält Variablen oder Funktionen, die von anderen Komponenten benötigt werden. Damit diese Zugriff auf die Elemente des `Service`-Controllers haben, wurden sie mit Angular-Syntax als `provider`-Elemente deklariert. Dadurch wird erreicht, dass diese Klasse in andere Komponenten importiert und verwendet werden kann.

GlobalService

Im `GlobalService` (siehe Abbildung 7.2) finden sich Variablen für die Speicherung der Daten eines Fragebogens, die vom `JSONService` eingelesen werden. Beim Start eines Fragebogens werden die Variablen an den `GlobalService` übergeben und den anderen Komponenten des Systems zur Verfügung gestellt.

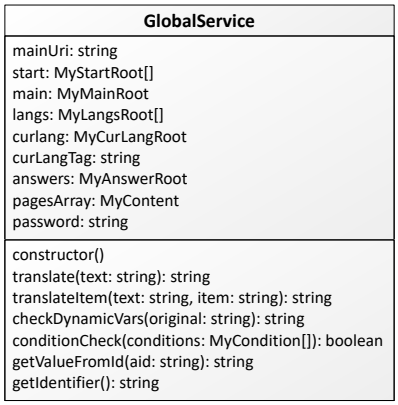


Abbildung 7.2: Aufbau des `GlobalService`

Des weiteren werden im `GlobalService` zwei Übersetzungsfunktionen implementiert (siehe Listing 7.1). Dabei ist die `translate`-Funktion für einfache Beschriftungen und die `translateItem`-Funktion für Übersetzungen von Items zuständig. Die Aufteilung in zwei Funktionen war notwendig, da übersetzte Texte der Items in `label`, `placeholder` und `measure` aufgetrennt wurden.

```
translate(text:string):string{
  try{
    let translated = this.curlang[this.curLangTag][text.slice(2,text.length -2)];
    if (typeof translated == 'undefined') throw new TypeError(); //Wert existiert nicht
    return this.checkDynamicVars(translated); //Ersetzen der dynamischen Variablen
  }catch(e){
    if(e instanceof TypeError){
      //keine Uebersetzung gefunden, die Referenz wird zurueckgegeben
      return text;
    }
  }
}
```

Listing 7.1: `translate`-Funktion des `GlobalService`

In den Übersetzungsfunktionen wird am Ende die Funktion `checkDynamicVars` aufgerufen, welcher der übersetzte String übergeben wird (siehe Listing 7.2). Diese prüft rekursiv, ob sich dynamische Variablen im String befinden. Ist keine (mehr) vorhanden, wird der String zurückgegeben. Falls der String dynamische Variablen enthält, wird die erste gefundene Variable ersetzt. Mit dem resultierenden String wird die Funktion erneut aufgerufen, bis keine dynamische Variablen mehr enthalten sind.

```
checkDynamicVars(original:string):string{
  let start = original.indexOf("{{"); //-1 wenn nicht vorhanden
  if (start == -1)  return original; //keine zu ersetzende Variable gefunden

  let end = original.indexOf("}}");
  let replaceOld = original.substring((start+2),end);
  if (typeof replaceOld == 'undefined') return "undefined";
  let key = this.main.variables[replaceOld]; //Referenz auslesen
  let replaceNew=this.getValueFromId(key); //neuen Wert erhalten

  return this.checkDynamicVars(original.replace("{{"+replaceOld+"}}",replaceNew));
}
```

Listing 7.2: `checkDynamicVars`-Funktion des `GlobalService`

Die `conditionCheck`-Funktion ist ein weiterer Bestandteil des `GlobalService` (siehe Listing 7.3). Diese bekommt als Parameter ein `MyCondition`-Objekt übergeben, welches aus einem Array mit Key/Value-Paaren besteht. Die Funktion überprüft für

alle Einträge des Arrays, ob der gespeicherte Wert zum Key mit dem übergeben Wert übereinstimmt. Ist dies für alle Arrayelemente der Fall, wird von der Funktion `true` zurückgegeben. Sobald aber nur ein Arrayelement die Bedingung nicht erfüllt, bricht die Überprüfung sofort ab und gibt `false` zurück.

```
1 conditionCheck(conditions:MyCondition[]):boolean{
2   condLoop:
3   for (let cond of conditions){
4     let savedValue=this.getValueFromId(cond.aid); //gespeicherten Wert auslesen
5     if(savedValue == cond.value) continue condLoop;
6     else return false; //Abbruch der weiteren Ueberpruefung
7   }
8   return true; //wird nur erreicht, wenn alle Bedingungen wahr wahren
9 }
```

Listing 7.3: `conditionCheck`-Funktion des `GlobalService`

JSONService

Der `JSONService` (siehe Abbildung 7.3) bietet der mobilen Applikation eine Schnittstelle für die Kommunikation mit dem Server. Dafür werden insbesondere zwei Funktionen zur Verfügung gestellt, die von anderen Komponenten aufgerufen werden.

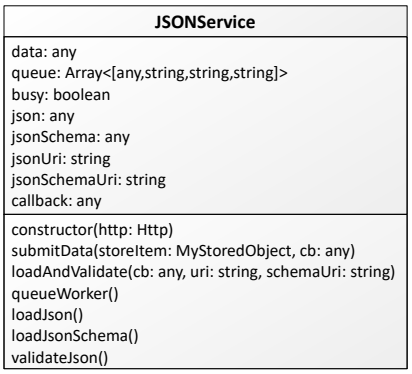


Abbildung 7.3: Aufbau des `JSONService`

Die `loadAndValidate`-Funktion wird für das Einlesen von JSON-Dateien benötigt, die auf einem externen Server liegen. Hierbei wird im `JSONService` der Vorgang in meh-

reren Schritten abgearbeitet. Zuerst werden alle benötigten Informationen gesammelt und in einer Warteschlange gespeichert (`queue`). Dies ist notwendig, da das Ionic Framework seit dem *Release Candidate 0* nur noch Singleton-`Services` bietet, das heißt, es kann maximal eine Instanz des `JSONService`-Controller geben [24]. Ursprünglich wurde für jede Kommunikationsoperation eine eigene Instanz verwendet, was aber, bedingt durch das Update, zum Warteschlangenmodell abgeändert wurde. Die Elemente innerhalb der Warteschlange werden sequentiell abgearbeitet. Darin befinden sich die URLs zu der JSON-Datei und dem zugehörigen Schema, sowie eine Callback-Funktion. Im nächsten Schritt wird das JSON mit den Daten und anschließend das Schema vom Server abgerufen. Gibt es bei der Kommunikation keine Probleme, werden die Daten validiert. Dazu wird die *is-my-json-valid*-Bibliothek[1] benutzt, welche in den `JSONService`-Controller eingebunden wurde (siehe Listing 7.4). Abschließend wird die Callback-Funktion aufgerufen, in welcher die Daten zurückgegeben werden.

```
1 validateJson(json:any, jsonSchema:any) {
2   let validator = is-my-json-valid;
3   let validate = validator(jsonSchema);
4   let valid = validate(json);
5   this.callback(valid, json);
6 }
```

Listing 7.4: `validateJson`-Funktion im `JSONService`

Die `submitData`-Funktion ist für das Ablegen der Daten auf dem Server verantwortlich. Da hier keine Validierung stattfindet, wird die Funktion direkt aufgerufen. Hierbei werden wieder eine Callback-Funktion, die URL zum Server und ein `MyStoredObject`, welches die Daten enthält, übergeben. Anschließend wird eine Callback-Funktion aufgerufen, die mitteilt, ob die Kommunikation mit dem Server und somit der Web-Export der Daten, erfolgreich war.

StorageService

Der `StorageService` (siehe Abbildung 7.4) bietet der mobilen Applikation eine Schnittstelle, die das Speichern von Daten auf dem mobilen Gerät ermöglicht. Dafür wurde

[1] `https://github.com/mafintosh/is-my-json-valid`

im `StorageService` das `Storage`-Modul des Ionic Frameworks implementiert, welches auf Basis des cordova-sqlite-storage-Plugins betrieben wird und somit SQLite zur Datenspeicherung auf dem mobilen Gerät zum Einsatz kommt. Dabei wird zu einem gegebenen Key ein gegebener Wert gespeichert.

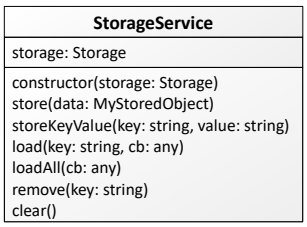


Abbildung 7.4: Aufbau des `StorageService`

Im `StorageService` werden zwei verschiedene Funktionen angeboten um Daten in den Speicher zu legen. Die `store`-Funktion bekommt ein `MyStoredObject` übergeben und speichert dieses mit der Instanz-ID als Key ab (siehe Listing 7.5).

```
1 public store(data:MyStoredObject){
2     this.storage.set(data.instance_id, data);
3   }
```

Listing 7.5: `store`-Funktion im `StorageService`

Der `storeKeyValue`-Funktion werden zwei Strings übergeben, und zwar einen Wert sowie einen Key unter welchem dieser Wert abgespeichert wird. Das Laden von Daten wird durch Callback-Funktionen realisiert. In der `load`-Funktion wird die Callback-Funktion mit dem Wert zu einem bestimmten Key aufgerufen. Die `loadAll`-Funktion ruft hingegen für jeden Key im Speicher die übergebene Callback-Funktion mit dem zum Key zugeordneten Wert auf. Das Löschen eines Werts wird mit der `remove`-Funktion, welche einen Key als Parameter besitzt, vorgenommen. Alternativ kann der komplette Speicher mit der `clear`-Funktion geleert werden, was zum Beispiel beim Zurücksetzen der mobilen Applikation verwendet wird.

7.2.2 Seiten und Komponenten

Ein Seiten-Modul besteht aus drei Bausteinen: Ein HTML-Template, welches den Aufbau und die Elemente des User Interface definiert; einer TypeScript-Komponente, in der Funktionen zur Kommunikation mit der Anwendung und der Benutzeroberfläche definiert sind; sowie eine SASS-Datei, die für das Design verantwortlich ist. Auf einer Seite können Komponenten eingebunden werden, die ebenfalls wieder aus einem HTML-Template und einer TypeScript-Datei bestehen. Für das Design einer Komponente wird die SASS-Datei der Vaterseite herangezogen. Durch spezielle, in den Komponenten definierte, HTML-Tags können diese im HTML-Template eingesetzt werden. Dabei können der Kind-Komponente über diese HTML-Tags Daten übergeben werden. Das Vaterelement kann auch Events empfangen, die von der Kind-Komponente ausgelöst wurden. Zusätzlich ist es möglich, dass eine Komponente ebenfalls weitere Komponenten enthält.

Startseite

Die Startseite, welche durch die Klasse `StartPage` (siehe Abbildung 7.5) definiert ist, wird nach der Initialisierung der mobilen Applikation angezeigt.

Im Konstruktor der `StartPage` wird als erstes die Funktion des `StorageService` aufgerufen, welche die Serveradresse aus dem Gerätespeicher ausliest und anschließend den `readServer`-Callback ausführt (siehe Listing 7.6).

```
1 readServer = (server:string) =>{
2   if (server==null) {
3     this.globalService.mainUri='http://kindex.johannes-schobel.at';
4   }else {
5     this.globalService.mainUri = server;
6   }
7   this.loadStartJson();
8 }
```

Listing 7.6: `readServer`-Callback in `StartPage`

Hier wird überprüft, ob es bereits einen Servereintrag im Gerätespeicher gab. Ist dies nicht der Fall, wird eine voreingestellte Serveradresse gewählt. In beiden Fällen wird

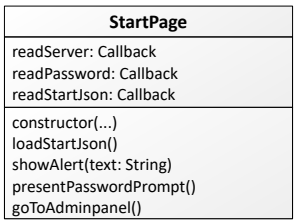


Abbildung 7.5: Aufbau der `StartPage`

diese Adresse in eine Variable des `GlobalService` gespeichert. Erst danach wird die Funktion zum Einlesen der `start.json` aufgerufen, welche vom `JSONService` eingelesen wird. Dieser führt anschließend den `readJson`-Callback aus (siehe Listing 7.7).

```
1 readJson = (valid:boolean, type:string, myJson:any) =>{
2     if (valid == true) {
3       this.globalService[type] = myJson; //in globalService speichern
4     } else {
5       this.showAlert("Fehler beim Einlesen"); //Fehlermeldung anzeigen
6     }
7   }
```

Listing 7.7: `readJson`-Callback in `StartPage`

Wurde die Datei erfolgreich validiert, werden die Daten in eine Variable des `GlobalService` gespeichert. Andernfalls wird eine Fehlermeldung angezeigt. Der nächste Schritt wird durch die Angular2-Directive `*ngFor` im HTML-Template ausgelöst (siehe Listing 7.8).

```
1 <ion-content padding>
2   <div *ngFor="let qst of globalService.start">
3     <my-start-card [qst]="qst"></my-start-card>
4   </div>
5 </ion-content>
```

Listing 7.8: Ausschnitt des HTML-Templates der `StartPage`

Für jedes `qst`-Objekt vom `MyStartRoot`-Typ im Array `globalService.start` wird ein `<div>`-Element, sowie dessen Inhalt, angezeigt. Dies beinhaltet das HTML-Tag `<my-start-card>`. Dadurch wird für jedes `MyStartRoot`-Element eine Komponente

der `MyStartCard`-Klasse erstellt, welcher zugleich das aktuelle `qst`-Objekt übergeben wird. Zusätzlich wird das HTML-Template der `MyStartCard` an die Stelle des `<my-start-card>`-Tags gesetzt.

`MyStartCard`-Komponente Die `MyStartCard`-Komponente (siehe Abbildung 7.6) zeigt Informationen zu einem Fragebogen an, welcher in der `start.json` aufgeführt werden.

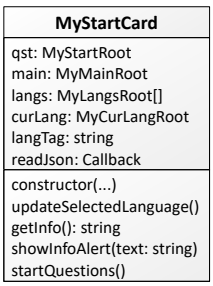


Abbildung 7.6: Aufbau der `MyStartCard`-Komponente

Um weitere Informationen zu einem bestimmten Fragebogen zu erhalten, wird im Konstruktor der `MyStartCard` die `main.json` geladen. Der `JSONService` führt nach erfolgreichem Download die `readJson`-Callbackfunktion aus (siehe Listing 7.9).

```
1  readJson = (valid:boolean, prop:string, myJson:any) =>{
2    if (valid == true) {
3      if (prop === 'main') {
4        this.main=myJson;
5        this.jsonService.loadAndValidate(this.readJson,this.globalService.mainUri+'/'+
             this.qst.path+'/languages.json', this.globalService.mainUri+'/schemas/
             languages_schema.json', 'langs');
6      }else if (prop === 'langs') {
7        this.langs=myJson;
8        this.jsonService.loadAndValidate(this.readJson,this.globalService.mainUri+'/'+
             this.qst.path+'/'+this.langs[0].uri, this.globalService.mainUri+'/schemas/
             curlang_schema.json', 'curLang');
9        this.langSelection=this.langs[0];
10     }else if (prop === 'curLang') {
11       this.curLang=myJson;
12       this.langTag=this.langSelection.code;
13     }
```

```
14      }else {
15          this.showInfoAlert("Fehler beim Einlesen ("+qst.name+")");
16      }
17  }
```

Listing 7.9: `readJson`-Callback in der `MyStartCard`

Am Anfang der Funktion wird überprüft, ob es sich um ein valides JSON handelt. Sollte dies nicht der Fall sein, wird eine Fehlermeldung angezeigt. Ansonsten wird überprüft, zu welchem Datenobjekt das JSON zugeordnet werden muss. Dabei behandelt die Funktion drei Fälle: Wurde die `main.json` eingelesen, wird diese im `MyMainRoot`-Objekt gespeichert. Anschließend wird die Funktion zum Laden der Sprachdatei aufgerufen. Falls die `languages.json` eingelesen wurde, wird diese im `MyLangsRoot`-Array gespeichert. Danach wird das erste Element gewählt und die dazugehörige Sprachdatei geladen. Der dritte Fall wäre das Einlesen einer Sprachdatei, welche im `MyLangRoot`-Objekt gespeichert wird. Zusätzlich wird hier noch der aktuelle Sprachcode in der `langTag`-Variable gesetzt. Der `MyStardCard` stehen nun alle Daten zur Verfügung, die benötigt werden um die Benutzeroberfläche des HTML-Templates darzustellen (siehe Abbildung 7.7).

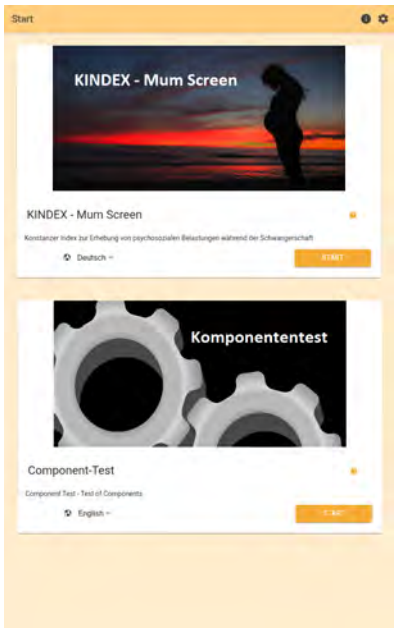


Abbildung 7.7: Aufbau der `StartPage`-Seite mit `MyStartCard`-Komponenten

Die angezeigten Elemente werden alle aus dem `MyMainRoot`-Objekt gelesen. Die Beschreibung der Inhalte der Fragebögen wird mit Hilfe der bereits vorgestellten Übersetzungsmethoden des `GlobalService` übersetzt (siehe Listing 7.1). Die Sprachauswahl für einen Fragebogen findet auf der `MyStardCard` statt. Hier kann durch ein Auswahlfeld eine andere Sprache gewählt werden. Durch das bei einer Änderung ausgelöste Event wird eine Funktion ausgeführt, welche die neue Sprachdatei einliest. Diese wird dann wieder in der `readJson`-Methode behandelt. Da der Text der Beschreibung an die Übersetzungen gebunden sind, werden diese im gleichen Schritt automatisch aktualisiert. Sobald der Start-Button gedrückt wird, werden alle benötigten Information des gewählten Fragebogens in die Variablen des `GlobalService` kopiert und anschließend auf die Fragebogenansicht gewechselt.

Fragebogenansicht

Die Fragebogenansicht wird in der `QuestionViewPage`-Klasse (siehe Abbildung 7.8) umgesetzt.

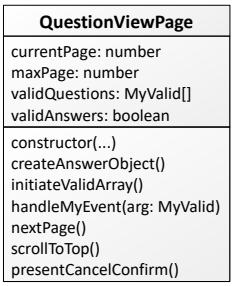


Abbildung 7.8: Aufbau der `QuestionViewPage`

Im Konstruktor der `QuestionViewPage` wird zuerst der `MyAnswerRoot`-Array instanziiert, der die Antworten zu den einzelnen Fragen speichert und im `GlobalService` hinterlegt wird. Anschließend wird der `MyValid`-Array erstellt, welcher zur Überprüfung auf gültige Eingaben in einer Seite benutzt wird. Weiterhin wird eine zufällige Instanz-ID generiert, welche mit einem Zeitstempel des Startzeitpunkts und dem Code der verwendeten Sprache im `MyAnswerRoot`-Objekt gespeichert wird (siehe Listing 7.10).

```
constructor(private nav: NavController, public globalService:GlobalService, public
    alertCtrl:AlertController) {
  this.globalService.answers = new MyAnswerRoot();
  this.initiateValidArray();
  this.globalService.answers.instance_id = Math.floor(Math.random()*10000000000);
  this.globalService.answers.started_at = new Date();
  this.globalService.answers.language = this.globalService.curLangTag;
}
```

Listing 7.10: Konstruktor der `QuestionViewPage`

Dadurch sind die Voraussetzungen für das Ausfüllen des Fragebogens erfüllt. Die Anzeige wird durch Komponenten aufgebaut (siehe Abbildung 7.9). Dazu dienen die Klassen `MyHeadline`, `MyText`, `MyMedia` und `MyQuestion`. Diese werden anhand spezieller HTML-Tags im Template der `QuestionViewPage` eingebunden (siehe Listing 7.11).

```
<div *ngFor="let element of globalService.main.content[currentPage].elements">
  <!--ELEMENT IS HEADLINE-->
  <div *ngIf="element.type == 'HEADLINE'">
    <my-headline [element]="element" (myevent)="handleMyEvent($event)"></my-headline>
  </div>
  <!--ELEMENT IS TEXT-->
  <div *ngIf="element.type == 'TEXT'">
    <my-text [element]="element" (myevent)="handleMyEvent($event)"></my-text>
  </div>
  <!--ELEMENT IS QUESTION-->
  <div *ngIf="element.type == 'QUESTION'">
    <my-question [element]="element" (myevent)="handleMyEvent($event)"></my-question>
  </div>
  <!--ELEMENT IS MEDIA-->
  <div *ngIf="element.type == 'MEDIA'">
    <my-media [element]="element" (myevent)="handleMyEvent($event)"></my-media>
  </div>
</div>
...
<button ion-button block [disabled]="!validAnswers" (click)="nextPage()">{{
    globalService.translate("__next__")}}</button>
```

Listing 7.11: Ausschnitt aus dem HTML-Template der `QuestionViewPage`

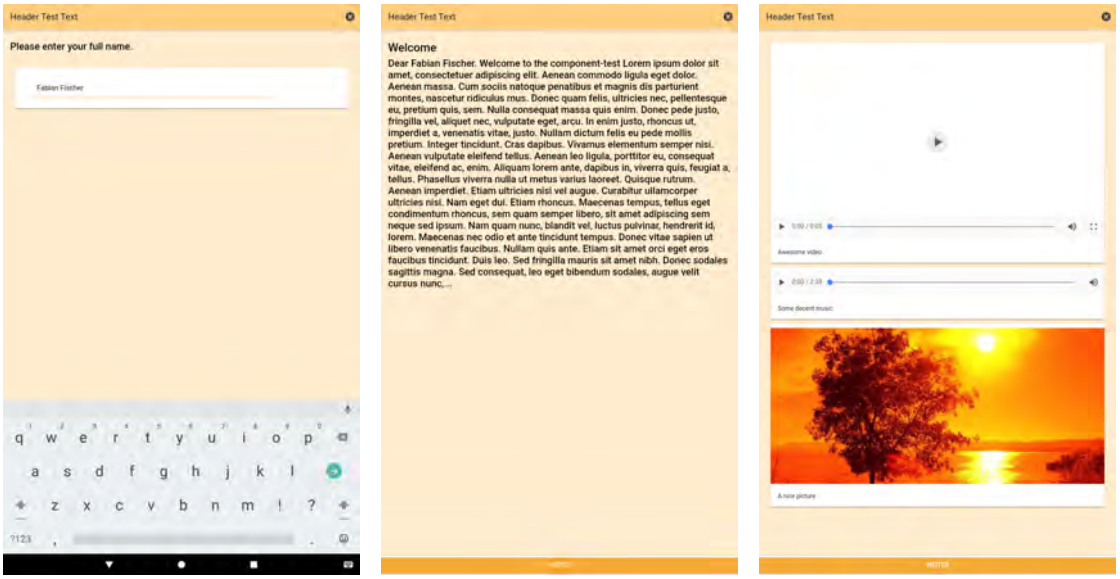


Abbildung 7.9: Screenshots der `QuestionViewPage` mit unterschiedlichen Elementen

Die Anzeige wird für ein `MyContent`-Objekt aus dem `content`-Array des `MyMainRoot`-Objekts aufgebaut. Dazu wird anhand des Typs eines Elements die passende Komponente durch die `*ngIf`-Directive gewählt. Hierbei findet eine Bindung der aktuellen `QuestionViewPage` an Events statt, die von den Kindkomponenten (`my-headline`, `my-text`, `my-question` oder `my-media`) ausgelöst werden und in der Funktion `handleMyEvent` der `QuestionViewPage` bearbeitet werden (siehe Listing 7.12). Zusätzlich wird der Kindkomponente ein `element`-Objekt vom Typ `MyElement` übergeben, welches alle benötigten Informationen zur Darstellung der Komponente enthält.

```
1 handleMyEvent(arg:MyValid) {
2   this.validQuestions[arg.id] = arg;
3   let keyArray = Object.keys(this.validQuestions);
4   let allValid = true;
5   for (let key of keyArray) {
6     if (this.validQuestions[key].valid == false) {allValid = false; break;}
7   }
8   this.validAnswers=allValid;
9 }
```

Listing 7.12: `handleMyEvent`-Funktion der `QuestionViewPage`

Hier wird durch das ausgelöste Event ein `MyValid`-Objekt übergeben, welches Informationen über den aktuellen Zustand der Kindkomponente enthält. Das `MyValid`-Objekt enthält die ID und den Validierungszustand. Dieses wird im `validQuestions`-Objekt abgespeichert. Anschließend wird überprüft, ob alle Elemente der `QuestionViewPage` valide Eingaben enthalten. Ist dies der Fall wird die `validAnswers`-Variable auf `true` gesetzt, was zur Folge hat, dass der Weiter-Button aktiviert wird. Dadurch kann die `nextPage`-Funktion aufgerufen werden, wodurch die nächste Seite angezeigt wird. Diese Funktion inkrementiert dabei nur die `currentPage`-Variable, wodurch die Daten aus dem nächsten Element des `content`-Arrays gelesen werden und sich die Benutzeroberfläche anhand der Angular-Datenbindung automatisch aktualisiert.

Seitenelemente der `QuestionViewPage` Seitenelemente der `QuestionViewPage` sind Komponenten, die durch vier Klassen definiert sind (siehe Abbildung 7.10).

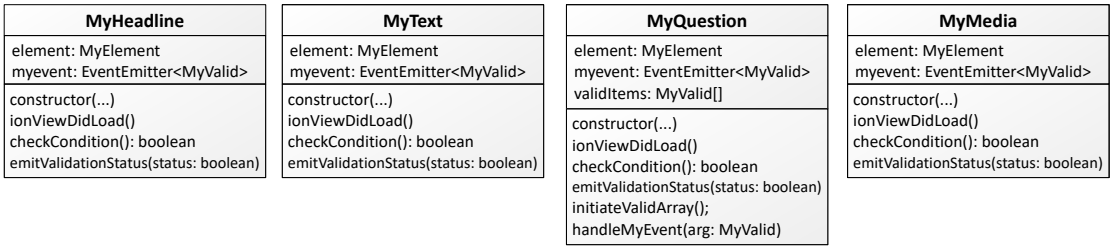


Abbildung 7.10: Seitenelemente `MyHeadline`, `MyText`, `MyQuestion` und `MyMedia`

Sie stellen typische Elemente eines Fragebogens dar:

`MyHeadline`: Wird zur Darstellung einer Überschrift verwendet.

`MyText`: Wird benutzt, um Textpassagen darzustellen.

`MyQuestion`: Zeigt Fragen an, wobei einer Frage ein oder mehrere `Items` zugeordnet werden, welche die Art der Benutzereingabe bestimmen.

`MyMedia`: Ermöglicht Bilder, Audioclips und Videos einzubinden. Die Medieninhalte erhalten zusätzlich eine optionale Beschriftung.

Eine Funktion, die alle Komponenten besitzen, ist die `checkCondition`-Funktion (siehe Listing 7.13). Falls ein Element nur unter bestimmten Bedingungen angezeigt werden soll, überprüft diese Funktion, ob alle erforderlichen Bedingungen erfüllt werden.

```
checkCondition():boolean{
  if(this.item.condition.length == 0) return true; //keine Bedingungen
  else{
    let showItem:boolean;
    showItem = this.globalService.conditionCheck(this.item.condition);
    if (showItem==false) this.emitValidationStatus(true); //valide Eingabe bei
        ausgeblendeten Elementen
    return showItem;
  }
}
```

Listing 7.13: `checkCondition`-Funktion in den Seitenkomponenten

Die Funktion wird von einem HTML-Tag durch die Angular-Directive `*ngIf` aufgerufen, welche den gesamten Inhalt der Komponente beinhaltet (siehe Listing 7.14). Wenn von `checkCondition` ein `false` als Rückgabewert geliefert, so wird der komplette Inhalt ausgeblendet (siehe Abbildung 7.11).

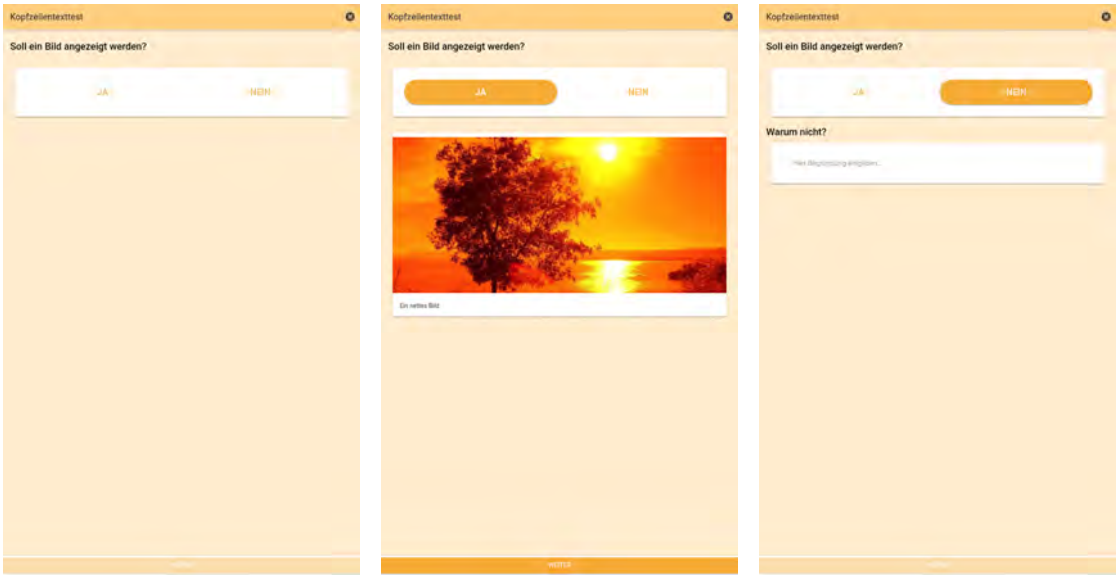
```
<div *ngIf="checkCondition()">
  <!-- Aufbau der Komponente -->
</div>
```

Listing 7.14: Funktionsaufruf von `checkCondition` im HTML-Template

Zusätzlich besitzen alle Seitenkomponenten die `emitValidationStatus`-Funktion (siehe Listing 7.15). Diese löst Events aus, welche die `handlyMyEvent`-Methode in der `QuestionViewPage` ausführen, die über den validen oder invaliden Zustand der Komponente informieren. Zusätzlich wird überprüft, ob eine Eingabe überhaupt benötigt wird, falls die Frage als optional markiert wurde.

```
emitValidationStatus(valid:boolean){
  if(this.item.required == true ) {
    this.myevent.emit(new MyValid(this.item.id, valid));
  }else{
    this.myevent.emit(new MyValid(this.item.id, true));
  }
}
```

Listing 7.15: `emitValidationStatus`-Funktion in den Seitenkomponenten



(a) Initialer Zustand (b) Bedingtes Seitenelement (c) Bedingtes Antwortelement

Abbildung 7.11: Screenshots bedingter Elemente der mobilen Applikation

Bei den `MyHeadline`, `MyText` und `MyMedia` Komponenten wird die Validierungsfunktion direkt in der `ionViewDidLoad`-Funktion mit `true` als Parameter ausgeführt, da hier keine Benutzereingabe stattfindet und die Elemente direkt nachdem sie angezeigt werden, als valide gelten. Die `MyQuestion`-Komponente besitzt ebenfalls Kindkomponenten, die validiert werden müssen, welche diese Klasse um zusätzliche Funktionen erweitert. Dabei läuft der Validierungsvorgang nach dem selben Prinzip ab wie zwischen der `QuestionViewPage` und deren Kindkomponenten. Das heißt: Erst wenn alle Kindkomponenten valide Eingaben signalisiert haben, löst die `MyQuestion`-Komponente das Event aus, welches von der `QuestionViewPage` registriert wird (siehe Abbildung 7.12).

Antwortelemente der `MyQuestion`-Komponente Die Antwortelemente werden in sechs Klassen aufgeteilt, deren Funktionen aber zu einem großen Teil identisch sind. Der Hauptunterschied findet sich in den HTML-Templates, die für unterschiedliche Fragen bestimmte GUI-Elemente zum Einlesen der Benutzereingaben verwenden. Die

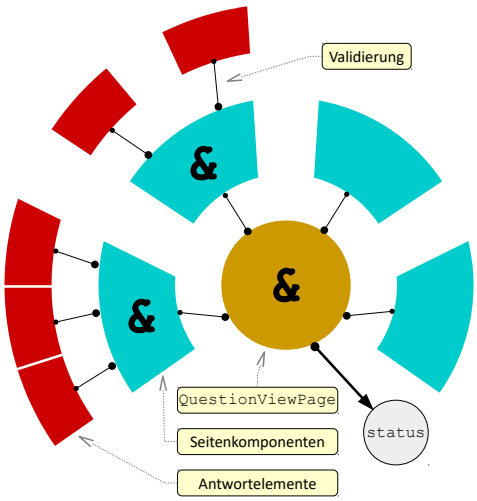


Abbildung 7.12: Beispielhafter Ablauf zur Validierung der Seiten- und Antwortelemente auf der `QuestionViewPage`

TypeScript-Komponente kann als `MyItem`-Objekt generalisiert werden (siehe Abbildung 7.13). Hierbei hängt nur die `modelChanged`-Funktion von der konkreten Klasse ab.

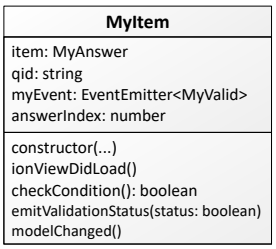


Abbildung 7.13: Beispielhafter Aufbau des `MyItem`

`MyItemYesNo`-Komponente Dieses Item besteht aus einer Gruppierung zweier Ionic Buttons, die für einfache *Ja-/Nein*-Fragen verwendet werden. Wird der Button gedrückt, wird er farblich hinterlegt und wird als ausgewählt betrachtet. Es ist nicht möglich, beide Buttons zur gleichen Zeit auszuwählen.

`MyItemFreeText`-Komponente Das *Freitext*-Item besteht aus einem `ion-input`-Element, das als gewöhnliches Textfeld dargestellt wird. Die Eingabe von Daten

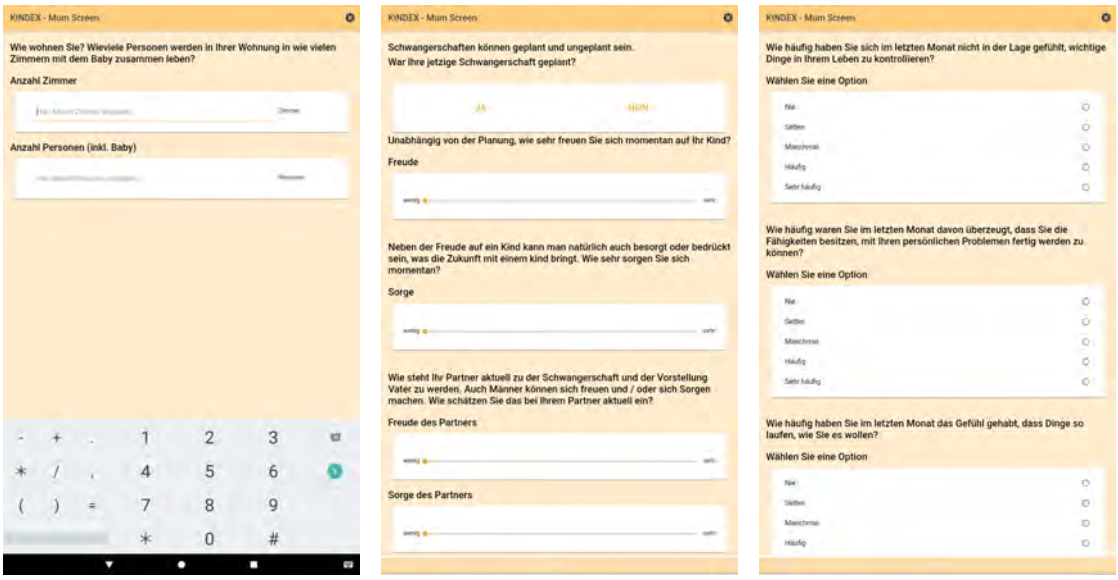
kann hier zum Beispiel nur auf Zahlen begrenzt werden. Dabei wird am mobilen Gerät direkt die geeignete Tastatur angezeigt (siehe Abbildung 7.14a).

`MyItemDate`-Komponente Zur Eingabe von einem *Datum* wurde das `ion-datetime`-Element verwendet. Dies bietet eine einfache Möglichkeit, ein Datum einzulesen.

`MyItemMultipleChoice`-Komponente Das Item für *Multiple-Choice*-Fragen basiert auf einem `ion-checkbox`-Element, das pro Eintrag im Datenarray ein Checkbox-Element generiert. Dadurch erhält man eine Liste, in der mehrere Einträge markiert werden können (siehe Abbildung 7.14c).

`MyItemSingleChoice`-Komponente Die *Single-Choice*-Komponente ist ähnlich aufgebaut wie die *Multiple-Choice*-Variante. Hier wird anstelle des `ion-checkbox`-Element, ein `ion-radio`-Element mit einem `ion-label` verwendet.

`MyItemSlider`-Komponente Für *Skalen* werden `MyItemSlider`-Komponenten verwendet, welche aus einem `ion-range`-Element bestehen. Hier kann der minimale und maximale Wert definiert, sowie die Beschriftung der ion-range Komponente geändert werden. Es wird zusätzlich die Möglichkeit geboten, kleine Icons (*Ionicons*) als Ersatz für die Beschriftungen zu nutzen (siehe Abbildung 7.14b).



(a) Frage 8 (b) Frage 14 (c) Frage 16

Abbildung 7.14: Screenshots der `Items` aus dem KINDEX-Fragebogen

Wenn der Benutzer einen Wert der Antwort ändert, wird die `modelChanged`-Funktion aufgerufen, die überprüft, ob die Eingabe valide ist. Die Eingabe wird sofort in einem Array abgespeichert und eventuelle nachträgliche Änderungen werden durch eine Indexerhöhung in das nächste Feld abgelegt. Zusätzlich wird ein Zeitstempel gespeichert, damit berechnet werden kann, wie lange die Beantwortung der Frage gedauert hat. Dies wird am Beispiel der `MyItemYesNo`-Klasse deutlich (siehe Listing 7.16).

```
1 modelChanged(sel:string){
2   this.buttonSelected=sel;
3   this.globalService.answers.questions[this.qid].answers[this.item.id][this.answerIndex
        ] = new MyAnswerAnswer(this.item.id, this.item.export_name, this.buttonSelected,
         new Date().toString());
4   this.emitValidationStatus(true);
5   this.answerIndex++;
6 }
```

Listing 7.16: `modelChanged`-Funktion der `MyItemYesNo`-Klasse

Zusammenfassend kann gesagt werden, dass die mobile Applikation anhand der vorgestellten Beispiele zur Implementierung dazu in der Lage ist, den KINDEX und weitere ähnlich strukturierte Fragebögen darzustellen. Die Hauptfunktionen werden dabei durch die Service-Controller zur Verfügung gestellt, welche von den Seiten und Komponenten benötigt werden. Auf die Verwaltungsseite sowie die Auswertungsseite wurde in diesem Kapitel nicht näher eingegangen, da deren Implementierung diesem Prinzip entspricht und die selben Teile des Ionic Frameworks zum Einsatz kommen.

8

Zusammenfassung

Das Ziel der Arbeit war es, eine mobile Applikation mit modernen Webtechnologien zu entwickeln, die den KINDEX, einen psychologischen Fragebogen zur Erhebung und Bewertung von Risikofaktoren während der Schwangerschaft, umsetzt.

Zu Beginn wurden die Grundlagen geschaffen, die für die Entwicklung mobiler Aplikationen relevant sind. Hierbei wurde der Fokus speziell auf den KINDEX gelegt. So wurde erklärt, wie herkömmliche Befragungen durch papierbasierte Fragebögen ablaufen und welche Probleme diese mit sich bringen. Da der KINDEX bisher ein papierbasierter Fragebogen ist, wurde speziell auf die Probleme der Ärzte, Hebammen und Mitarbeiter der Beratungsstellen, durch welche die Befragung stattfindet, eingegangen. Auch die Schwierigkeiten, welche die Psychologen der Universität Konstanz hatten, wurden behandelt. Deswegen wurden Lösungsansätze aufgezeigt, die auf einer elektronische Variante des Fragebogens basieren. Dem zu Folge wurden ebenfalls verschiedenen Entwicklungsarten für mobile Applikationen aufgezeigt.

Anschließend wurden zwei vorhandene mobile Applikationen betrachtet, die den KINDEX zum einen für das iOS- und zum anderen für das Android-Betriebssystem konzipiert und realisiert hatten. Anhand des Funktionsumfangs konnten erste Anforderungen an die im Rahmen dieser Arbeit erstellte mobile Applikation gestellt werden.

Um die Anforderungen genauer zu bestimmen, wurde der Ablauf der Befragungen modelliert. Hierbei wurde der Ist-Zustand des KINDEX veranschaulicht, sodass die Probleme verdeutlicht und Lösungen genauer bestimmt werden konnten. Anhand dieser Anforderungen wurde ein Soll-Zustand des Ablaufs modelliert, in welchen die Lösungsvorschläge integriert wurden.

Auf Basis dieser Erkenntnisse wurde die Anforderungsanalyse durchgeführt, wodurch die Anforderungen aufgezeigt und präzisiert werden konnten. Dazu wurden diese in funktionale und nichtfunktionale Anforderungen aufgeteilt.

Anschließend wurde die mobile Applikation anhand der identifizierten Funktionen konzipiert. Dazu wurde zunächst die Architektur des Gesamtsystems entwickelt und dargestellt, welche aus drei Komponenten besteht. Diese Komponenten wurden genauer behandelt. Auf die Kommunikation der mobilen Applikation mit der Serverkomponente wurde intensiv eingegangen, da der Server definiert, in welcher Struktur dieser die Daten für die mobile Applikation zur Verfügung stellt. So wurde das JSON-Format zur Speicherung von Fragebögen gewählt, dessen Daten über eine REST-konforme Schnittstelle übertragen werden.

Daraufhin konnten Aspekte der Implementierung diskutiert werden. Zunächst wurde das Ionic Framework vorgestellt. Dies basiert auf zwei weiteren Frameworks (Angular2 und Cordova) und kommt zur Entwicklung webbasierter hybrider Applikationen für mobile Geräte zum Einsatz. Bei der Implementierung wurde besonderen Wert auf eine hierarchische Strukturierung der einzelnen Bestandteile der mobilen Applikation gelegt. Dadurch werden die angezeigten Elemente erst zur Laufzeit generiert, wodurch im Fall von Änderungen deutlich mehr Flexibilität gewonnen wird. Auch wenn im Rahmen dieser Arbeit nur wenige Anforderungen nicht mit der geforderten Qualität erfüllt wurden, wie zum Beispiel die Mehrbenutzer-Eigenschaft, setzt die daraus resultierende mobile Applikation die Grundanforderungen für den Betrieb um und erreicht somit das geforderte Ziel.

8.1 Ausblick

Die realisierte mobile Applikation bietet im jetzigen Entwicklungsstadium bereits einen großen Funktionsumfang. Durch die Nutzung der JSON-Dateien eines externen Servers sind Änderungen oder Erweiterung der Fragebögen einfach realisierbar. So wäre es durchaus denkbar, dass der KINDEX in Zukunft nicht nur auf Deutsch oder Englisch verfügbar ist, sondern auch in andere Sprachen übersetzt wird. Aufgrund der Auslage-

rung in externe Dateien, kann dies auch von Novizen übernommen werden, da dazu der Programmcode der mobilen Applikation nicht angepasst werden muss. Dadurch könnte einem breiten Spektrum der Gesellschaft der Zugang zum KINDEX in der eigenen Sprache ermöglicht werden.

Dies beschränkt sich aber nicht nur auf unterschiedliche Sprachen. So wäre es ebenfalls vorstellbar, dass in der mobilen Applikation auch andere Fragebögen bearbeitet werden, für welche die betreffenden Dateien erstellt werden. Die mobile Applikation wurde bewusst für einen universalen Einsatz ausgelegt.

Möglicherweise könnte die mobile Applikation in Zukunft auch so erweitert werden, dass Änderungen an den Fragebögen vorgenommen oder sogar neue Fragebögen erstellt werden können. Dafür müsste eine Benutzeroberfläche entwickelt werden, welche dem Benutzer die Möglichkeit gibt, die verschiedenen Elemente, aus denen ein Fragebogen besteht, per Drag and Drop an die entsprechende Stelle zu ziehen und diese Komponente anschließend zu konfigurieren [25]. Wünschenswert wäre auch die Erweiterung der Benutzerverwaltung für den Mehrbenutzerbetrieb. Dort könnten dann, abhängig von der Funktion, Rechte hinterlegt werden, die das Bearbeiten und Neuerstellen von Fragebögen beschränken.

Literaturverzeichnis

[1] Statista: Anteile der einzelnen Gerätetypen an der Internetnutzungsdauer in Deutschland in den Jahren 2013 bis 2016. Website (2017) https://de.statista.com/statistik/daten/studie/455003/umfrage/anteile-der-geraetetypen-an-der-internetnutzungsdauer/; abgerufen am 17. Januar 2017.

[2] Schobel, J., Pryss, R., Schickler, M., Reichert, M.: Towards flexible mobile data collection in healthcare. In: 29th IEEE International Symposium on Computer-Based Medical Systems (CBMS 2016). (2016) 181–182

[3] Marcano Belisario, José S., e.a.: Comparison of self-administered survey questionnaire responses collected using mobile apps versus other methods. The Cochrane Collaboration (2014)

[4] Schobel, J., Schickler, M., Pryss, R., Reichert, M.: Process-Driven Data Collection with Smart Mobile Devices. Institute of Databases and Information Systems, Ulm University, Germany (2015)

[5] Liebrecht, M.: Technische Konzeption und Realisierung einer mobilen Anwendung für den Konstanzer-Index zur Erhebung von psychosozialen Belastungen während der Schwangerschaft. Institute of Databases and Information Systems, Ulm University, Germany, Diplomarbeit (2012)

[6] Lane, Shannon J., e.a.: A review of randomized controlled trials comparing the effectiveness of hand held computers with paper methods for data collection. BMC Medical Informatics and Decision Making (2006)

[7] Carlbring, Per, e.a.: Internet vs. paper and pencil administration of questionnaires commonly used in panic/agoraphobia research. Department of Behavioural Sciences, Linköping University, Sweden (2005)

[8] Palermo, Tonya M., e.a.: A randomized trial of electronic versus paper pain diaries in children: impact on compliance, accuracy, and acceptability. Department of Pediatrics, Division of Behavioral Pediatrics and Psychology, Rainbow Babies and Children's Hospital, Cleveland, OH, USA (2003)

[9] IBM Corporation: Overview: IBM Worklight projects, applications, environments, and skins. Website (2014) `http://www.ibm.com/support/knowledgecenter/SSZH4A_5.0.6/com.ibm.worklight.help.doc/devref/c_overview_projects_apps_envs_skins.html`; abgerufen am 15. Dezember 2016.

[10] Blanco, A.S.: DEVELOPMENT OF HYBRID MOBILE APPS Using Ionic framework. Mikkeli University of Applied Sciences, Finnland, Bachelorarbeit (2016)

[11] Schobel, J., Schickler, M., Pryss, R., Nienhaus, H., Reichert, M.: Using Vital Sensors in Mobile Healthcare Business Applications. Institute of Databases and Information Systems, Ulm University, Germany (2013)

[12] Steppich, I.: Konzeption und Realisierung einer mobilen Anwendung zur Unterstützung von Schwangeren. Institute of Databases and Information Systems, Ulm University, Germany, Bachelorarbeit (2016)

[13] Sadabadi, A.T.: Rapid prototyping for software projects with user interfaces. Department of Computer Systems and Informatics, State Engineering Uinversity of Armenia (2009)

[14] Google: Material Design. Website (2016) `https://material.io/guidelines/`; abgerufen am 17. Januar 2017.

[15] Apple: iOS Human Interface Guidelines. Website (2017) `https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/`; abgerufen am 19. Januar 2017.

[16] Statista: Marktanteile der führenden Betriebssysteme an der Internetnutzung mit Tablets in Deutschland von September 2012 bis November 2016. Website (2016) `https://de.statista.com/statistik/daten/studie/306850/umfrage/marktanteile-der-tablet-betriebssysteme-an-der-internetnutzung-in-deutschland/`; abgerufen am 17. Januar 2017.

[17] ECMA International: ECMA-404 The JSON Data Interchange Standard. Website (2016) http://www.json.org/; abgerufen am 23. Januar 2017.

[18] IETF: JSON Schema: A Media Type for Describing JSON Documents. Website (2016) http://json-schema.org/latest/json-schema-core.html; abgerufen am 23. Januar 2017.

[19] Hampton Catlin, Natalie Weizenbaum, Chris Eppstein, e. a.: Sass. Website (2016) http://sass-lang.com/documentation/#sass_gem_version_inline_docs; abgerufen am 25. Januar 2017.

[20] Microsoft: TypeScript. Website (2016) https://www.typescriptlang.org/; abgerufen am 25. Januar 2017.

[21] The Apache Software Foundation: Cordova Introduction. Website (2015) http://cordova.apache.org/docs/en/latest/guide/overview/index.html; abgerufen am 25. Januar 2017.

[22] Google: Angular 2 Architecture Overview . Website (2017) https://angular.io/docs/ts/latest/guide/architecture.html; abgerufen am 25. Januar 2017.

[23] Ionic: v2 Framework Docs. Website (2017) http://ionicframework.com/docs/; abgerufen am 25. Januar 2017.

[24] Ionic: 2.0.0-rc.0 (2016-09-28). Website (2016) https://github.com/driftyco/ionic/blob/master/CHANGELOG.md#200-rc0-2016-09-28; abgerufen am 26. Januar 2017.

[25] Schobel, J., Pryss, R., Schickler, M., Reichert, M.: A configurator component for end-user defined mobile data collection processes. In: Demo Track of the 14th International Conference on Service Oriented Computing (ICSOC 2016). (2016)

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Fabian Fischer Matrikelnummer: 754572

Erklärung

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

Ulm, den .....

Fabian Fischer