ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Medizin
Institut für Medizinische Systembiologie

Entwicklung einer erweiterbaren Struktur zur Abbildung medizinischer Instrumente

Masterarbeit an der Universität Ulm

Vorgelegt von:
Maximilian Scheurich
maximilian.scheurich@uni-ulm.de

Gutachter:
Prof. Dr. Hans A. Kestler
Prof. Dr. Manfred Reichert

Betreuer:
Dr. Johannes Schobel

2020

Fassung 21. Mai 2020

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Die Erfassung und Verarbeitung von Daten spielt in der Medizin eine große Rolle. Von der Patientenaufnahme bis hin zur Abbildung von Krankheitsverläufen und Therapieansätzen werden Daten erfasst, verarbeitet und analysiert. Zahlreiche Arbeiten haben sich in den letzten Jahren mit dieser Thematik auseinander gesetzt um diesen Weg zu digitalisieren und zu optimieren. Strukturen, wie sie in standardisierten Protokollen oder Dateiverwaltungssystemen zu finden sind, gewährleisten eine typsichere und einheitliche Möglichkeit Daten zu sichern und zu übertragen. Motivation dieser Arbeit ist es nicht nur eine Struktur zu schaffen, die die Abbildung von medizinischen Instrumenten ermöglicht. Vielmehr zielt diese Arbeit darauf ab eine einheitliche Grundlage für datenverarbeitende Systeme in der Medizin zu erstellen.

Trotz der vielen guten Ansätze in Bezug auf die Erfassung von Daten im Bereich der Medizin, kommt es aufgrund fehlender Standardisierung oftmals zu Problemen bei der Verarbeitung und dem Austausch dieser Informationen. Ein Lösungsansatz bietet eine gemeinsame Struktur, welche unabhängig von dem Anwendungssystem verwendet werden kann. Ziel dieser Arbeit soll es nicht sein eine vollendete Datenstruktur zu präsentieren. Viel mehr soll das Ergebnis eine Grundlage schaffen, auf die zukünftige Projekte aufbauen können. Es soll eine Vorgabe geschaffen werden, die aber zugleich eine dynamische Erweiterung ermöglicht und zukünftige Vorhaben nicht begrenzt. Eine Grundvoraussetzung für das Gelingen ist die strukturelle Vereinfachung ein medizinischer Datenerfassung.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.

Ein besonderes Dankeschön geht an meinen Betreuer Dr. Johannes Schobel, der für hilfreiche Anregungen und konstruktive Kritik bei der Erstellung dieser Arbeit gesorgt hat.

Inhaltsverzeichnis

1

Einleitung

Folgende Abschnitte bringen dem Leser die Motivation dieser Arbeit, sowie die damit einhergehende Zielsetzung und auftretende Probleme näher. Zuletzt folgt eine Beschreibung des Aufbaus dieser Arbeit.

1.1 Motivation

Die Erfassung und Verarbeitung von Daten spielt in der Medizin eine große Rolle. Von der Patientenaufnahme bis hin zur Abbildung von Krankheitsverläufen und Therapieansätzen werden Daten erfasst, verarbeitet und analysiert. Zahlreiche Arbeiten haben sich in den letzten Jahren mit dieser Thematik auseinander gesetzt um diesen Weg zu digitalisieren und zu optimieren. Einen Beweis und die Grundlage für den Bedarf einer Digitalisierung in diesem Bereich liefert unter anderem die Arbeit [31].

Die Universität Ulm, insbesondere das Institut für Datenbanken und Informationssysteme, hat dazu mit den veröffentlichten Arbeiten [19, 21, 40, 42, 43, 44, 45, 46] einen großen Beitrag geleistet.

Strukturen, wie sie in standardisierten Protokollen oder Dateiverwaltungssystemen zu finden sind, gewährleisten eine typsichere und einheitliche Möglichkeit Daten zu sichern und zu übertragen. Motivation dieser Arbeit ist es nicht nur eine Struktur zu schaffen, die die Abbildung von medizinischen Instrumenten ermöglicht. Vielmehr zielt diese Arbeit darauf ab eine einheitliche Grundlage für datenverarbeitende Systeme in der Medizin zu erstellen.

1.2 Problemstellung

Trotz der vielen guten Ansätze in Bezug auf die Erfassung von Daten im Bereich der Medizin, kommt es aufgrund fehlender Standardisierung oftmals zu Problemen bei der Verarbeitung und dem Austausch dieser Informationen. Die Arbeit [24] zielt darauf ab Daten von schwangerer Frauen zur Erfassen und zu Bearbeiten. Dagegen befasst sich die Arbeit [14] mit der Erfassung von Astmahnpatienten. Aufgrund fehlender Standarisierung in Bezug auf die zugrundeliegende Datenstruktur ist es nur möglich diese beiden Ansätze durch die Entwicklung mehrerer Schnittstellen in Einklang zu bringen.

Ein Austausch von Patientendaten zweier medizinischer Institutionen, die nicht das selbe Anwendungssystem zur Erfassung und Abbildung von Daten besitzen stoßen auf ähnliche Probleme. Für den Austausch ist in diesem Fall entweder eine manuelle Eingabe der Daten oder eine erneute Befragung oder Untersuchung des Patienten nötig. In beiden Fällen ist der Patient der Leid tragende. Eine wiederholte Analyse kostet Zeit und belastet den Patienten unnötig. Eine manuelle Eingabe wiederum belastet die Ressourcen des medizinischen Instituts.

Ein Lösungsansatz bietet eine gemeinsame Struktur, welche unabhängig von dem Anwendungssystem verwendet werden kann.

1.3 Zielsetzung

Ziel dieser Arbeit soll es nicht sein eine vollendete Datenstruktur zu präsentieren. Viel mehr soll das Ergebnis eine Grundlage schaffen, auf die zukünftige Projekte aufbauen können.

Es soll eine Vorgabe geschaffen werden, die aber zugleich eine dynamische Erweiterung ermöglicht und zukünftige Vorhaben nicht begrenzt. Eine Grundvoraussetzung für das Gelingen ist es durch ein strukturelle Vereinfachung ein komplexes Systems in ein kompliziertes zu überführen, welches beherrschbar ist.

1.4 Struktur der Arbeit

Die Arbeit ist wie folgt in neun Kapitel unterteilt. Die Definition und Bedeutung medizinischer Instrumente wird in Kapitel 2 geklärt. In Kapitel 3 werden mehrere verwandte Arbeiten vorgestellt. Es werden die Kernaspekte dieser Arbeiten erläutert, um eine Einordnung der zu entwickelnden Datenstruktur vorzunehmen.

Kapitel 4 beschäftigt sich mit der Erhebung von Daten, sowie mit der daraus resultierenden Technik zur Entwicklung einer erweiterbaren Software im Sinne der Endbenutzer-Entwicklung, die in Kapitel 2 erläutert wird.

Das Kapitel 5 befasst sich mit den Anforderungen an die Datenstruktur. Sowohl funktionale, als auch nicht-funktionale Anforderungen werden in diesem Abschnitt definiert.

Das Kapitel 6 soll einen Einblick in den Aufbau der Datenstruktur geben und wird durch das folgende Kapitel 7 ergänzt, in dem die wichtigsten verwendeten Technologien vorgestellt werden. Kapitel 8 nimmt Bezug auf die in Kapitel 5 definierten Anforderungen anhand einer exemplarischen Erweiterung der Datenstruktur ein. In Kapitel 9 wird abschließend ein Fazit über das behandelte Thema gezogen und eine Prognose in Hinblick auf zukünftige Arbeiten erstellt.

2

Grundlagen

Diese Kapitel bringt dem Leser einige Grundlagen näher und schafft ein gemeinsames Verständnis für den Begriff „medizinische Instrumente“.

2.1 Medizinische Instrumente

Der Begriff „medizinisches Instrument“ wird traditionell mit praktischen Werkzeugen in der Medizin assoziiert. Eine Erweiterung der Bedeutung dieses Begriffs schließt heutzutage psychologische Hilfsmittel, zur Messung menschlichen Verhaltens mit ein. Darunter fallen Fragebögen, so wie Interviews [22].

Diese Arbeit beschränkt sich auf die Instrumente in den Bereichen des Gesundheitswesen und der Psychologie. Bei der Betrachtung eines medizinischen Instrumentes unter dem Aspekt des psychologischen Hilfsmittels lassen sich viele wiederkehrende Bereiche erkennen. Diese werden im Verlauf dieser Arbeit als Elementen eines medizinischen Instrumentes bezeichnet. Das Ziel dieser Arbeit ist es diese Elemente zu definieren, zu kategorisieren und schlussendlich in einer Datenstruktur abzubilden.

Elemente medizinischer Instrumente

Im Verlauf dieser Arbeit wird der Begriff „medizinisches Instrument“ häufig mit dem englischen Wort für Fragebogen (engl.: Questionnaire) gleichgesetzt. Es handelt sich bei einem Fragebogen um eine konkrete Form eines medizinischen Instruments.

Die Tabelle 2.1 soll einen Überblick über die in dieser Arbeit behandelten Elemente geben. Es wird zwischen vier Arten von Elementen unterscheiden. Die *Strukturelemente* bilden das Gerüst des `Questionnaires`. Sie sind Träger der *Datenerfassungselemente* und dienen zur Abbildung komplexer Filterfragen. Unter den *Ereigniselemente* sind alle Elemente versammelt, die eine Bedingung für ein Ereignis tragen oder den Ausgang bestimmen. Die *Beschreibungselemente* dienen der Informationsmitteilung. Es kann sich hierbei um Bilder, als auch um längere Erklärungstexte handeln. Die *Datenerfassungselemente* beinhalten alle Elemente, die direkt oder indirekt Patientendaten erfassen können. Hierzu zählen auch Sensoren, welche keine aktive Bearbeitung benötigen [37].

Strukturelemente

Page	Gruppiert mehrere Datenerfassungselemente.
Start	Besitzt die gleichen Eigenschaften wie `Page`, kann aber nur einmal in einem medizinischen Instrument vorkommen.
End	Besitzt die gleichen Eigenschaften wie `Page`, kann aber nur einmal in einem medizinischen Instrument vorkommen.
Gateways	
AND-Gateway	Eröffnet mehrere Pfade, die in beliebiger Reihenfolge bearbeitet werden können.
XOR-Gateway	Eröffnet mehrere Pfade, die sich über eine Bedingung gegenseitig ausschließen.
LOOP-Gateway	Ermöglicht es einen Pfad n-mal zu durchlaufen, mit $n \in \mathbb{N} \setminus \{0\}$.

Ereigniselemente

Case	Markiert Pfade mit einem Fall, bei der Verwendung eines XOR-Gateways.
Condition	Beinhaltet die Bedingung eines XOR- oder LOOP-Gateways.
Event	Führt eine Aktion aus, wenn dieses Element betreten wird.

Beschreibungselemente

Text	Hält Texte zur genaueren Beschreibung für den Benutzer bereit.
Custom	Hält eine benutzerdefinierte Informationsübergabe bereit.
Headline	Überschrift zur Beschreibung eines Abschnittes.
Media	Hält verschiedene Informationsquellen bereit, wie Bilder oder Videos.

Datenerfassungselemente	
Sensor	Ermöglicht das einlesen von Daten über verschiedene Sensoren, wie Blutdruck- oder Pulsmessungen.
Questions	
Matrix	Erlaubt Auswahlmöglichkeiten in einer Matrixanordnung.
FreeInt	Erlaubt die Eingabe einer Zahl.
FreeText	Erlaubt die Eingabe eines Textes.
FreeDate	Erlaubt die Eingabe eines Datums.
Slider	Erlaubt die Eingabe über einen Schieberegler.
DropDown	Erlaubt die Auswahl aus einem Dropdown-Menü.
Buttongrid	Erlaubt die Eingabe über ein Schaltflächenraster.
Likert	Erlaubt die Erfassung von persönlichen Einstellungen nach der Likert-Skala.
Ranking	Erlaubt eine Einordnung.
YesNo	Erlaubt nur Antworten mit „Ja“ und „Nein“.
MultipleChoice	Erlaubt die Auswahl mehrere Artikel.
Distribution	Erlaubt eine Antwort über eine Verteilung.
SingleChoice	Erlaubt die Auswahl von genau einem Artikel

Tabelle 2.1: Gliederung der Struktur-, Ereignis-, Beschreibungs- und Datenerfassungselementen

2.2 Internationalisierung

Internationalisierung (i18n) spielt eine zentrale Rolle bei der benutzerorientierten Softwareentwicklung. Sie bietet die Möglichkeit Texte, Zahlen und Daten an die sprachlichen und kulturellen Eigenschaften des Benutzer oder des Patienten anzupassen ohne den Quellcode ändern zu müssen [23].

Grundlage für die erfolgreiche Einbindung von Sprachpaketen sind die Datenobjekte, auf welchen das Programm aufbaut. Für eine erfolgreiche Umsetzung der Datenstruktur ist es deshalb unabdinglich frühzeitig die Internationalisierung einzuplanen. Zu diesem Zweck wurde eine Schnittstelle definiert, die eine einheitliche Erstellung von Sprachpake-

ten erlaubt. Dem Benutzer soll es bei der Erstellung eines `Questionnaire` außerdem möglich sein eigene Texte in Abhängigkeit von den eingebundenen Sprachpaketen zu setzten.

Das Listing 2.1 zeigt, wie ein `TextElement` mit den Sprachen Deutsch und Englisch erstellt werden kann.

```
export class TextElement extends BaseElement {
  @IsI18n()
  headline: I18n = {};
  @IsI18n()
  text: I18n = {};
}

const text = new TextElement();
text.headline = {
  de: "Überschrift",
  en: "Headline"
};
text.text = {
  de: "Das ist ein Beispieltext",
  en: "This is a sample text"
};
```

Listing 2.1: Internationalisierung am Beispiel `TextElement`

2.3 Endbenutzer Entwicklung

Laut des Artikels [33] zählen sich 90 Millionen Amerikaner zu einer Gruppe, die einfache Programmieraufgaben in ihrer täglichen Arbeit bewältigt. Mit der steigenden Digitalisierung und dem täglichen Einsatz von Computern soll diese Zahl über die nächsten Jahre noch weiter ansteigen.

Unter dem Begriff „Endbenutzer Entwicklung“ versteht man in der Informatik die Entwicklung eine Software, die es dem Benutzer ermöglicht Erweiterungen und Anpassungen vorzunehmen. Das bekannteste Beispiel hierfür ist die Makroprogrammierung in Tabellendokumenten, wie LibreOffice oder Microsoft Excel. Die Endbenutzer Entwicklung ist

auch unter Endbenutzer Programmierung bekannt, welche sich auf Werkzeuge bezieht mit denen der Endbenutzer eine Änderung von Software-Artefakten vornehmen kann, ohne erweiterte Programmierkenntnisse zu besitzen [18, 28, 36].

Ein Teilgebiet der Endbenutzer Entwicklung beschäftigt sich mit grafischen Programmiersprachen, die es dem Benutzer ermöglichen Code durch visuelle Elemente zu schreiben und nicht wie bei den üblichen Programmiersprachen durch die Eingabe eines Textes. Durch die Verwendung von Blöcken, die über Linien verbindet werden können, soll es dem Benutzer erleichtert werden Programme leichter zu verstehen.

Für diese Arbeit wird das Konzept der Endbenutzer Entwicklung auf die Erstellung einer medizinischen Instruments übertragen. Jedem Benutzer muss es möglich sein ein medizinisches Instrument zu erstellen.

Die Arbeit [41] dient als Vorlage bei der Entwicklung einer Anwendung, die es dem Benutzer in nur wenigen Schritten ermöglichen soll Anpassungen vorzunehmen, ohne erweiterte Programmierkenntnisse zu haben.

3

Verwandte Arbeiten

Dieses Kapitel setzt sich mit verwandten Arbeiten und deren Umsetzung auseinander. Es soll den aktuellen Stand des behandelten Themas widerspiegeln, sowie erste Erkenntnisse bezüglich der Anforderungen und Problemstellung liefern.

3.1 FHIR

Der FHIR (Fast Health Interoperability Resources) Standard der Stiftung HL7® FHIR® dient dem Austausch von Daten im Gesundheitswesen. Einer der Vorgänger von FHIR war HL7 v2 und ist in vielen Krankenhausgemeinschaften vertreten.

Dieser Standard dient dem Austausch von Gesundheitsinformationen, so wie elektronischer Krankenakteninformationen. FHIR beschränkt sich allerdings nicht nur auf Patientendaten, die dessen Gesundheitszustand betreffen. Es lassen sich mithilfe des FHIR Standards ganze Prozesse abbilden, die sowohl Patienten als auch Ärzte, Geräte und Organisationen betreffen. Daten von und über den Patienten können ebenfalls erfasst werden. Mit FHIR können nicht nur Diagnosen und Allergien, sowie familiäre Zusammenhänge abgebildet werden, sonder auch die Medikation eines Patienten überwacht werden.

Die Abbildung 3.1 zeigt die grundlegenden Datentypen des FHIR Standards. Es wird unterscheiden zwischen primitiven und komplexen Datentypen. Im Rahmen dieser Arbeit sind vor allem Zweitere von Bedeutung. Mithilfe der Elemente `Identifier`, `HumanName` und `Address` lassen sich ganz leicht Informationen eines Patienten abbilden, die nicht unmittelbar in Zusammenhang mit seiner Gesundheit stehen. Sogar

wiederkehrende Ereignisse lassen sich durch das `Period`-Element wiedergeben. Die Datentypen lassen sich sowohl in XML, als auch in JSON abbilden [15].

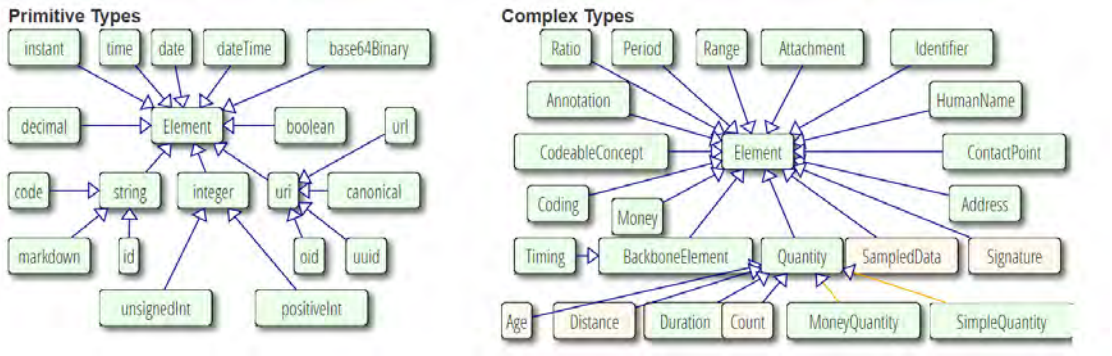

Abbildung 3.1: FHIR Datentypen [6]

Der Aufbau eines Fragebogens nach FHIR Standard besteht aus n Elementen, für die gilt $n \in \mathbb{N} \wedge n \geq 0$. Der Fragebogen selbst enthält Metadaten, wie den Verfasser, das Erstellungsdatum und den Titel. Ein Element des Fragebogens kann in eine von drei Kategorien eingeteilt werden. Die Anzeige-Elemente spiegeln Texte wieder, die Anweisungen, Hintergrundinformationen oder andere informative Inhalte besitzen können. Ein Anzeige-Element hat, wie auch eine Gruppen-Element keine Möglichkeit Daten zu erfassen. Ein Gruppen-Element gliedert den Inhalt des Fragebogens und kann mehrere Frage-Elemente enthalten. Ein Frage-Element besitzt eine spezifische Frage, welche eine Antwort zulässt. Die Art der Frage bestimmt dabei sowohl die Antwort, als auch den Type des Frage-Elements [12].

Der FHIR Standard ist ein mächtiger Apparat zur Abbildung von Daten, die auf das Gesundheitswesen bezogen sind. Eine so umfangreiche Datenstruktur hat, wie auch schon der Vorgänger HL7 v3, seinen Preis zu Gunsten der Komplexität der Bedienung.

3.2 openEHR

openEHR ist eine Lösung für Informationssystem im Gesundheitswesen. Die verschiedenen Bereiche von openEHR werden durch die internationale gemeinnützige Organisation

openEHR verwaltet. Abbildung 3.2 gibt einen Überblick über die Zusammensetzung des Systems.

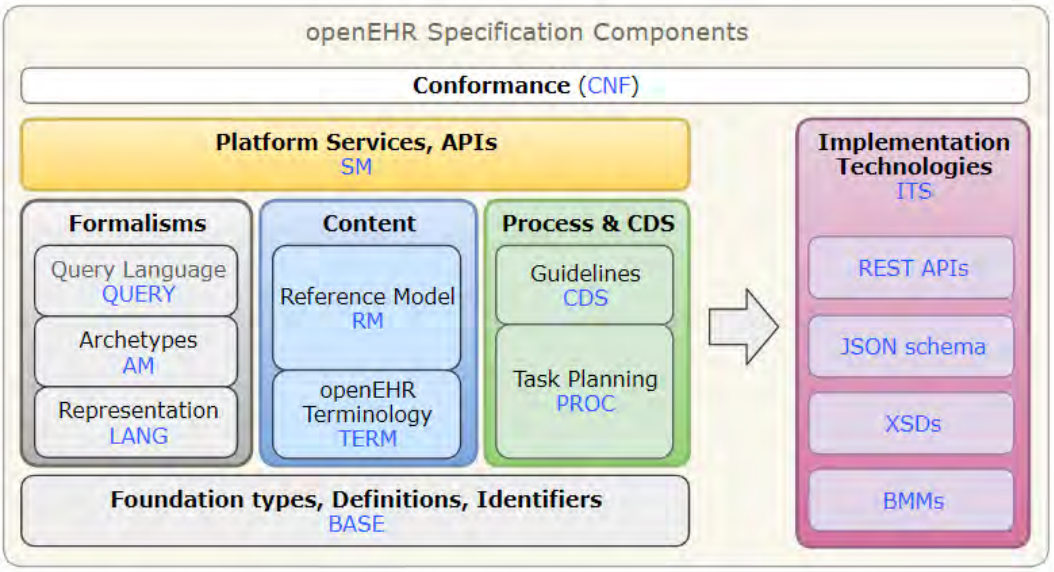

Abbildung 3.2: openEHR Überblick [11]

Die relevanten Aspekte des openEHR Systems beschränken sich im Rahmen dieser Arbeit auf die Bereiche *Reference Model* und *Foundation types, Definitions, Identifiers*. Während in dem Abschnitt zu *Foundation types, Definitions, Identifiers* grundlegende primitive Datentypen spezifiziert und definiert werden, werden in Abschnitt *Reference Model* konkretere Vorgaben in Hinblick auf die Informationsaufnahme von und für Patienten gemacht. Die Abbildung 3.3 zeigt den Aufbau eine Datenstruktur anhand einer exemplarischen Gewichtsmessung.

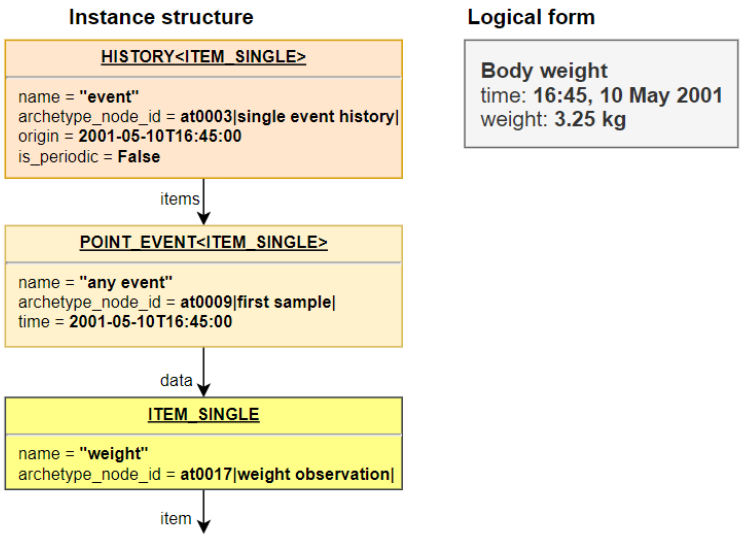

Abbildung 3.3: Beispiel Datenstruktur einer Gewichtsmessung [5]

Das Element `HISTORY` beinhaltet einen einmaligen Event, gekennzeichnet durch das Attribut `is_periodic = False`. Das Item `POINT EVENT`, welcher den Event festhält referenziert die Daten. In diesem Fall wird eine `ITEM_SINGLE` erzeugt, das schlussendlich die Werte, in diesem Fall das Gewicht, speichert.

Die openEHR Datenstruktur zeichnet sich vor allem durch die Detailgenauigkeit aus die sie bietet. Wie allerdings die Spezifikationen zeigen, welche unter https://specifications.openehr.org/ nachgelesen werden können, ist die Datenstruktur und die Bereiche darum komplex und lassen sich nicht mit Leichtigkeit in bestehende Systeme übertragen [26].

3.3 CDA

CDA (Clinical Document Architecture) ist eine auf XML basierende Datenstruktur zur Abbildung von klinischen Daten. Einer der Vorteile von XML ist es, dass es sowohl von Menschen gelesen werden, als auch von Maschinen bearbeitet werden kann. Dennoch unterscheidet die CDA Spezifikation zwischen freien und strukturierten Abschnitten die

speziell für die Verarbeitung vorgesehen sind. In Listing A.1 ist dies deutlich zu sehen. In Zeile 9 beginnt das frei Modul zur Wiedergabe von Vitalparametern eines Patienten. In Zeile 21 beginnt ein gesonderter Abschnitt zur Verarbeitung durch Maschinen für den gleichen Inhalt. Es werden hier allerdings, wie Zeile 25 Zeigt genau Vorgaben zu Werten und Einheiten gemacht.

Die CDA Datenstruktur ermöglicht es des weiteren Bilder und andere Multimediainhalte aufzunehmen. Der Aufbau eines CDA konformen XML-Dokuments lässt sich aus Tabelle 3.1 entnehmen [17].

CDA-Header	Patienteninformationen, Autor, Erstellungsdatum, Dokumenttyp, Anbieter, etc.
CDA-Body	Klinische Details, Diagnose, Medikamente, Nachsorge, etc. Wird als freier Text in einem oder mehreren Abschnitten präsentiert und kann optional auch kodierte Einträge enthalten.
CDA Level 1	Die Wurzelhierarchie und die uneingeschränkteste Version des Dokuments. Ebene Eins unterstützt die volle CDA-Semantik und hat eine begrenzte Kodierfähigkeit für den Inhalt. Ein Beispiel für eine Beschränkung des Dokumenttyps auf Ebene Eins wäre eine „Entladungszusammenfassung“ mit nur textlichen Anweisungen.
CDA Level 2	Zusätzliche Einschränkungen für das Dokument durch Vorlagen auf der Ebene „Abschnitt“ (freier Text). Ein Beispiel für eine Beschränkung der Ebene zwei wäre eine „Entlassungszusammenfassung“ mit einem als Medikamente kodierten Abschnitt.
CDA Level 3	Zusätzliche Einschränkungen für das Dokument auf der Ebene „Eingang“ (kodierter Inhalt) und optionale zusätzliche Einschränkungen auf der Ebene „Abschnitt“. Ein Beispiel für eine Einschränkung der Ebene drei wäre eine „Entlassungsübersicht“ mit einem als Medikamente kodierten Abschnitt mit kodierten RxNORM-Einträgen für jedes Medikament.

Tabelle 3.1: CDA-Struktur [1, 2, 3]

3.4 QuestionSys

QuestionSys [13] ist ein Projekt des Instituts für Datenbanken und Informationssysteme der Universität Ulm, welches sich mit der Erstellung und Auswertung von mobilen Fragebögen im Bereich der Psychologie befasst. Das Projekt entstand auf Grundlage der Doktorarbeit von Dr. Johannes Schobel [37]. Ziel dieses Projektes ist die Minimierung der Arbeit, die bei der Entwicklung von komplexen Auswertungsbögen anfällt. Die Anwendung des Systems bezieht sich hauptsächlich auf die Benutzung mobiler Endgeräte.

Wie die Abbildung 3.4 verdeutlichen soll, ermöglicht QuestionSys eine übersichtliche und einfach zu handhabende Darstellung eines komplexen Arbeitsschritts bei der Erstellung von geführten Fragebögen. Die Erstellung eines Fragebogens ist in Form von Prozessmodellen definiert. Die Bereitstellung eines Editors ⓐ für eine prozessorientierte Modellierung eines Fragebogens erlaubt dem Fachexperten eine einfache Verwendung. Darüber hinaus unterstützt QuestionSys den gesamten Lebenszyklus eines digitalen Fragebogens, bis hin zur Evaluierung durch einen Fachexperten. Dieser wird ebenfalls in seiner Aufgabe durch eine Datenaufbereitung ⓑ unterstützt. Diese Arbeit konnte zeigen, dass elektronische Fragebögen den Fachexperten von kostspieligen manuellen Aufgaben entlasten, wie die Übertragung, Transformation und Analyse der gesammelten Daten [35, 41, 43].

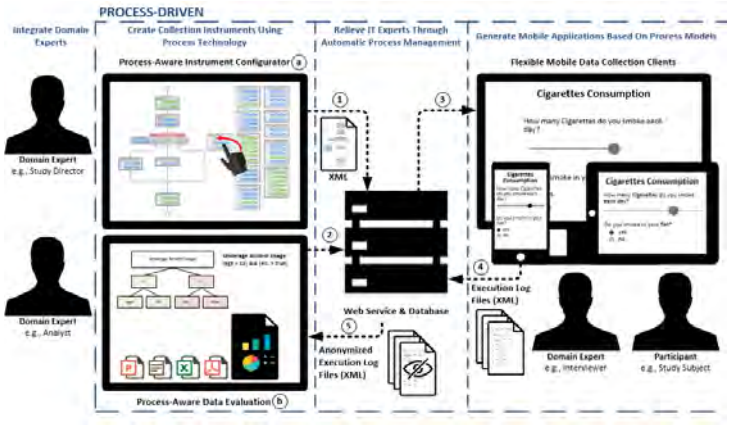

Abbildung 3.4: QuestionSys Projekt-Übersicht [13]

4

Konzept

Dieses Kapitel soll dem Leser Einsatzmöglichkeiten der Datenstruktur, sowie Vorteile einer gemeinsamen Basis vermitteln. Es werden in dieser Arbeit zwischen zwei Einsatzbereichen unterschieden. Im Bereich der Datenerfassung geht es darum welche Informationseingabemöglichkeiten einem Benutzer und einem Patienten zur Verfügung stehen. Bei der Datenverarbeitung geht es um die Speicherung und die Präsentation auf stationären und mobilen Endgeräten. Die Datenanalyse wird, aufgrund des komplexen Umfangs, in dieser Arbeit nicht behandelt.

4.1 Datenerfassung

Um ein Konzept für eine Struktur zu entwickeln, die medizinische Instrumente abbilden soll, ist es notwendig die Einsatzziele dafür zu kennen. Der Bereich Datenerfassung spielt eine große Rolle, da die Informationsgewinnung essenziell für die Informationswiedergabe ist. In den nächsten zwei Abschnitten wird skizziert wie eine mögliche Datenerfassung durch den Benutzer/Ersteller eines medizinischen Instrumentes aussehen und durch den Patienten Daten erfasst werden können. Einen guten Anhaltspunkt liefert die Arbeit [34], welche sich intensiv mit der Crowd Sensing Datenerfassung beschäftigt.

Benutzer

Die Abbildung 4.1 zeigt einen Konfigurator der einem Benutzer, in diesem Fall einem möglichen Psychologen, die Möglichkeit gibt ein medizinisches Instrument in Form eines *Questionnairs* zu erstellen. Unabhängig von der Oberfläche, die in Abbildung 4.1 gezeigt

wird kann man feststellen, dass für die Erstellung eines medizinischen Instrumentes häufig wiederkehrende Elemente verwendet werden. Daraus lässt sich eindeutig ableiten, dass eine objektorientierte Herangehensweise von Vorteil ist. Es wird außerdem klar, dass eine Abstraktionen von verschiedenen Elementen Vorteile bei sich wiederholenden Datensätzen zur Folge hat. Durch eine Abstraktion und objektorientierte Vorgehensweise lassen sich komplexere Aufbauten abspeichern und wiederverwenden.

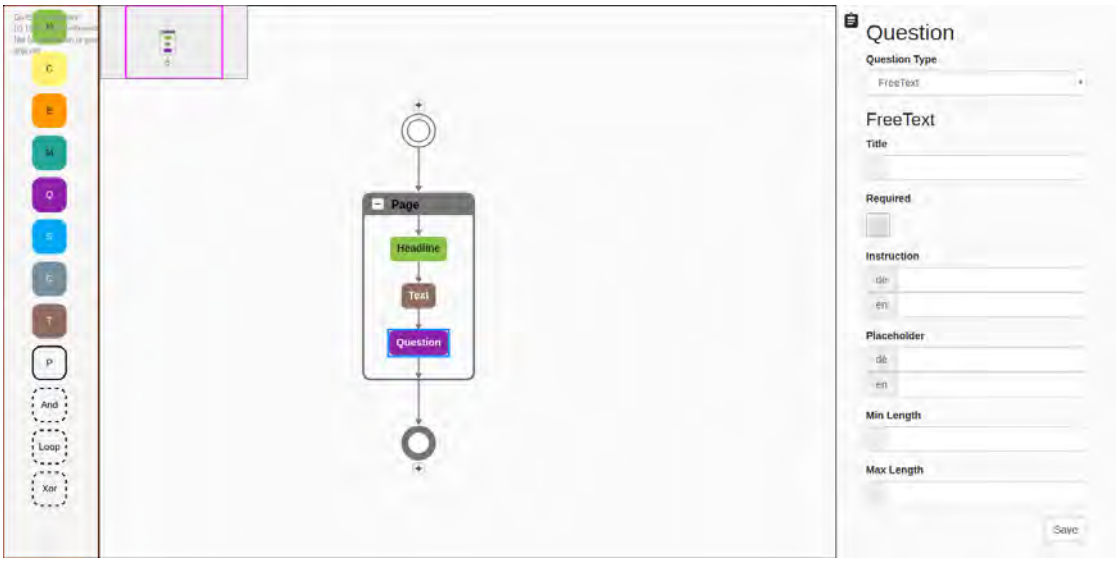

Abbildung 4.1: Erstellungsvorgang eines medizinischen Instrumentes

Patienten

Auf Seiten des Patienten ist für die Datenerfassung schematisch die Abbildung 4.2 dargestellt. Es werden in der Abbildung 4.2 verschiedene Arten von Datenerfassungselementen gezeigt, die es zu Untersuchen gilt. Die ersten drei Elemente erfassen Informationen des Patienten über Texteingabe, Schieberegler und der Eingabe eines Datums. Elemente vier und fünf heben sich aufgrund ihrer komplexeren Struktur von den ersten drei ab. Es wird eine andere Struktur, bzw. ein komplexer Datentyp benötigt für die Abbildung der Itemauswahl in Element vier und die Tabelle in Element fünf.

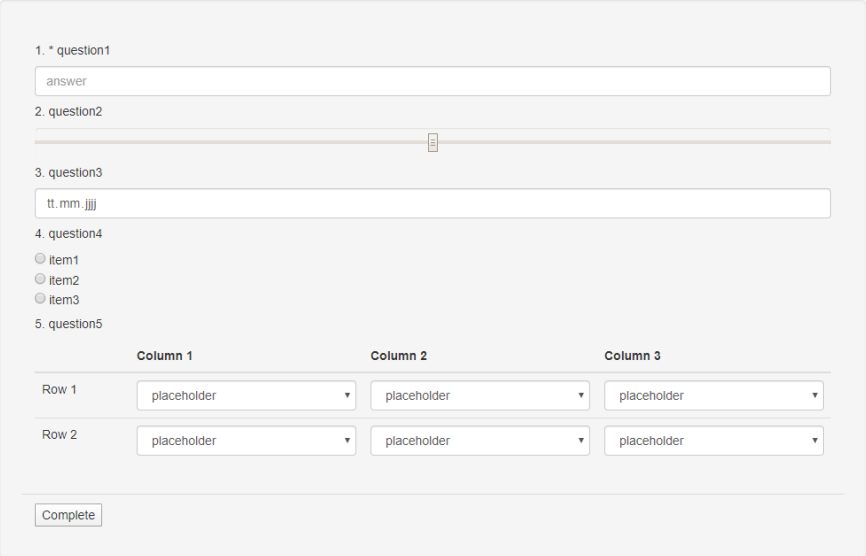

Abbildung 4.2: Schematische Darstellung des digitalen Fragebogens aus Sicht des Patienten

Durch Abbildung 4.2 wird außerdem klar, dass die zu entwickelnde Datenstruktur Eigenschaften enthalten muss, welche die Darstellung, bzw. die Art der Elemente für den Entwickler seitens der Anwendung unterstützen. Ein objektorientierter Ansatz stellt sich auch in diesem Kontext als vorteilhaft heraus.

4.2 Datenverarbeitung

Die Datenverarbeitung spielt bei der Entwicklung einer Struktur zur Abbildung von medizinischen Instrumenten eine zentrale Rolle. Variable und horizontal ausgeprägte Strukturen haben den Vorteil flexibel anpassbar zu sein und könne einfacher auf Ausnahmen reagieren. Soll eine große Menge an Daten in kurzer Zeit verarbeitet werden, so sind statische Strukturen im Vorteil. Ein vergleichbares Phänomen ist im Bereich von NoSQL und SQL Datenbanken zu sehen. In der Arbeit [27] wird diesbezüglich Stellung genommen.

In den kommenden zwei Abschnitten gilt es herauszufinden welchen Einfluss der Aufbau der Struktur, die in dieser Arbeit entwickelt werden soll, auf die Datenverarbeitung hat.

Backend

Die Abbildung 4.3 zeigt die Schnittstelle des QuestionSys Projektes, welches dem Institut für Datenbanken und Informationssysteme (DBIS) angehört. Diese API (Programmierschnittstelle) entstand auf Basis der Dissertation von Dr. Johannes Schobel [37]. Die unter der Bezeichnung „Instruments“ aufgelisteten Pfade der Schnittstelle zeigen alle Methoden, die für eine Speicherung, Bearbeitung und Löschung eines medizinischen Instrumentes zu Verfügung stehen. Bei der darunterliegenden Datenbank handelt es sich um eine Postgres SQL Datenbank. Der Vorteil gegenüber einer NoSQL Datenbank oder einer anderen SQL Datenbank liegt in der Fähigkeit von PostgreSQL neben der performanten Datenverarbeitung auch mit JSON Objekten umgehen zu können. Durch diese Besonderheit wurde darauf geachtet, dass eine Anpassung der Datenstruktur der medizinischen Instrumente keine Neuentwicklung des Backends benötigt wird.

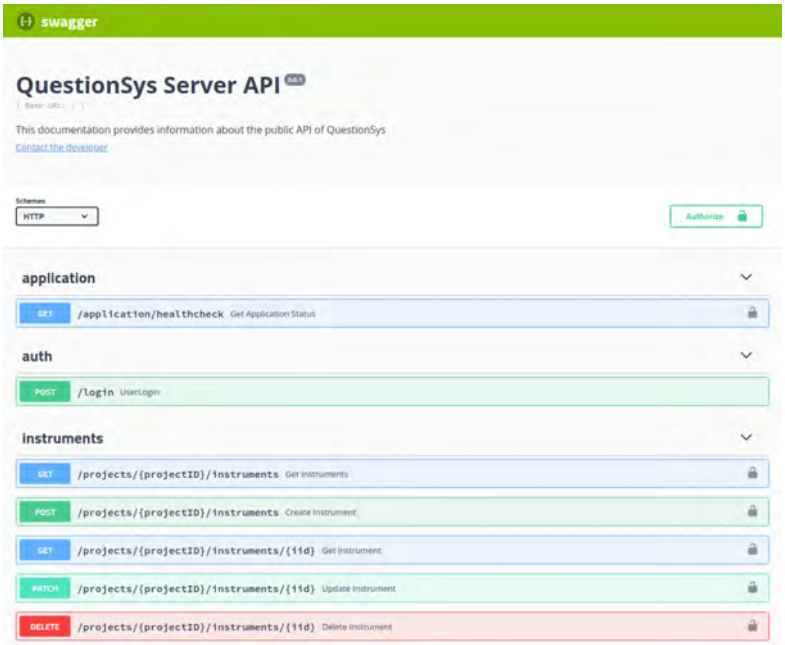

Abbildung 4.3: QuestionSys Server API

Endgeräte

Eine Studie des Pew Research Center [47] im Jahr 2019 hat ergeben, dass 76% aller an der Umfrage beteiligte Erwachsenen in Ländern mit fortgeschrittener Wirtschaft ein Smartphone besitzen. In Entwicklungsländern sind es im Durchschnitt 45%. Diese Tatsache, sowie die immer stärker wachsende digitale Branche legt nahe, insbesondere Smartphones bei der Betrachtung von Benutzerendgeräten zu berücksichtigen. Das wurde auch schon in den Arbeiten [35, 45] deutlich, die sich mit der Datenerfassung auf mobilen Endgeräten beschäftigt haben.

Eine Analyse der am häufigsten verwendeten Programmiersprachen auf GitHub von RedMonkey [30] ergab, dass sich JavaScript, Python und Java unter den Top drei der beliebtesten Programmiersprachen auf GitHub befinden, wie das Diagramm 4.4 zeigt.

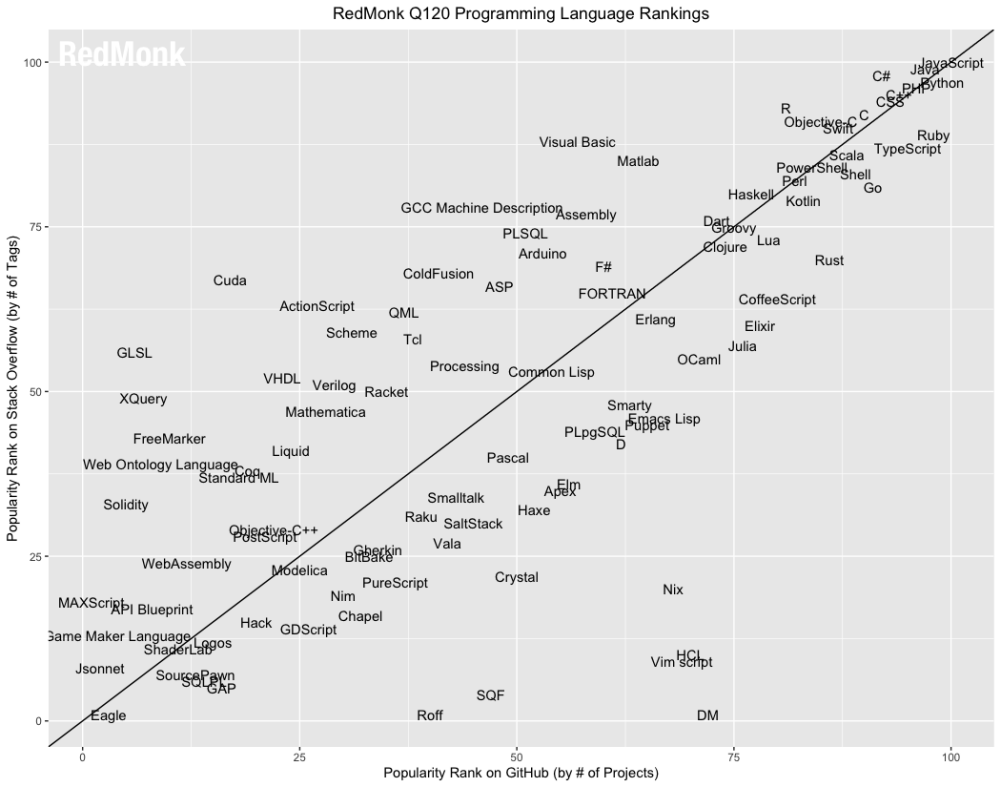

Abbildung 4.4: RedMonk Q120 Programming Language Rankings [30]

Diese Analyse, sowie die Tatsache, dass immer mehr Menschen ein eigenes Smartphone besitzen gibt Aufschluss über das zu verwendende Datenobjekt für diese Arbeit. Sowohl JavaScript, als auch Python und Java unterstützen nativ oder mithilfe von Standartbibliotheken die JavaScript Object Notation (JSON). Für die Verwendung von JSON spricht ebenfalls die Plattformunabhängigkeit, denn Framework wie Ionic [10] bieten Entwicklern die Möglichkeit unabhängig von dem Betriebssystemen Applikation auf Basis eines Webentwicklers (HTML, CSS, JavaScript) zu schreiben. Das hat den Vorteil, dass Programmiercode webbasierte Frontend Anwendungen für herkömmliche Browser (Chrome, Firefox, IE, Safari) bei der Entwicklung für Smartphone Anwendungen wiederverwendet werden können.

Die Wahl des Datenobjekts, welches in dieser Arbeit für die Umsetzung der Struktur eingesetzt wird, fällt auf JSON. Die in der Arbeit [29] aufgezeigten Unterschiede von JSON und XML tragen einen wesentlichen Anteil dieser Entscheidung. Es wird insbesondere TypeScript verwendet, welches eine objektorientierten Programmieransatz bietet, somit wird ebenfalls der Daten erfassende Anteil abgebildet.

5

Anforderungsanalyse

In diesem Kapitel werden anhand der gegebenen Problemstellung Anforderungen an das System definiert. Es wird zunächst eine Aufgabenanalyse durchgeführt und die Einsatzziele untersucht. Darüber hinaus wird eine Benutzerprofilanalyse erstellt, um den Umfang der Anforderungen an die Struktur genauer einordnen zu können. In Abschnitt 5.2 wird bei der Systemaufgabenanalyse zwischen funktionalen und nicht-funktionalen Anforderungen unterschieden.

5.1 Aufgabenanalyse

In diesem Abschnitt werden die genauen Aufgaben der zu entwickelnden Struktur anhand einer Benutzerprofilanalyse eingegrenzt. Dieser Prozess spielt vor allem bei der Entwicklung von Anwendungssystemen eine große Rolle. Zu beachten ist, dass die Dramenstruktur die im Rahmen dieser Arbeit entwickelt werden soll, die Grundlage bildet auf der zukünftige Anwendungssystem arbeiten werden. Es ist gerade deswegen von großer Wichtigkeit diese Faktoren bei der Entwicklung der Struktur miteinzubeziehen. Je detaillierter die Anforderungen an die Struktur herausgearbeitet werden, desto einfacher ist es funktionsfähige und brauchbare Anwendungssysteme dafür herzustellen.

5.1.1 Einsatzziele

Die Einsatzziele von Fragebögen sind über alle Bereiche des Lebens verteilt. Man begegnet ihnen bei Online-Umfragen, sowie bei den örtlichen Behörden in Form von Formularen. In anderen Bereichen, wie zum Beispiel der Psychologie ist die Absicht der

Verwendung von Fragebögen offensichtlicher. Unabhängig von der Art des Auftretens erfüllen Fragebögen im Allgemeinen den Zweck, Informationen und Meinungen bezüglich eines im Fragebogen angesprochenen Themas zu sammeln. Die Auswertung führt häufig zu einem besseren Verständnis der Verhaltensweise von Personen in bestimmten Situationen oder zur Aufnahme von Personendaten ohne analytischen Hintergrund.

Insbesondere bei der Entwicklung von Fragebögen im Bereich der Psychologie kann es zu hohem Arbeitsaufwand kommen. Der Grund hierfür ist der analytische Aspekt, welcher über eine gezielte Fragestellung erreicht werden soll. Bei der Beantragung eines Führerscheins zum Beispiel sind die gestellten Fragen unabhängig von der Personengruppe immer gleich aufgestellt. Eine psychologische Umfrage bezüglich des Trinkverhaltens hingegen muss zwischen Personengruppen differenziert und teilweise nach Antwortmöglichkeiten gefiltert werden. Ein hervorragende Einsatzmöglichkeit von digitalen Fragebögen liefert der Artikel „Outpatient Tinnitus Clinic, Self-Help Web Platform, or Mobile Application to Recruit Tinnitus Study Samples?“ [32], in welchem die Sammlung von Informationen zum Krankheitsbild „Tinnitus“ beschrieben wird.

Besonders in der Psychologie würde man von einer verbesserten Entwicklungsmethode profitiert. Somit fokussiert sich das Einsatzziel, welches im Rahmen dieser Arbeit betrachtet wird, auf die Entwicklung psychologischer Fragebögen.

5.1.2 Benutzerprofilanalyse

Softwareentwicklung, insbesondere die Entwicklung von Benutzerschnittstellen ist abhängig von der jeweiligen Benutzergruppe. Da bei einer homogenen Gruppe von Nutzern, welche sich in relevanten Eigenschaften ähneln, auch ähnliche Erwartungen an Benutzerschnittstellen vorausgesetzt werden können, fasst man sie in einer Benutzerkategorie zusammen. Für jede Benutzerkategorie wird ein Benutzerprofil erstellt, welches Aufschluss über die Entscheidung im Hinblick einer Benutzerschnittstelle liefern kann. Unter der Prämisse, dass sich die Benutzergruppe hauptsächlich auf den Bereich der Forschung, insbesondere der Psychologie, beschränkt, wird ein Psychologe die Stelle des Repräsentanten einnehmen.

Die Erwartungen und Kenntnisse dieses Benutzerprofils können wie folgt beschrieben werden. Ein Psychologe besitzt in der Regel umfangreiche Kenntnisse im Bereich der Psychologie. Aufgrund seiner Ausbildung beschränken sich die IT-Kenntnisse häufig nur auf die Verwendung von Anwendungen. Im Bereich der Anwendungssoftware wird eine simple und intuitive Bedienung vorausgesetzt, die effektives Arbeiten ermöglicht. Dennoch kann davon ausgegangen werden, dass Funktionalität vor Design gestellt wird. Diese Beurteilung schließt sich aus dem Berufsprofil, welches als rational und strukturiert beschrieben wird.

Basierend auf den relevanten Eigenschaften eines Psychologen lässt sich schließen, dass jede Person als Benutzer in Frage kommt, welche ein Grundverständnis von Anwendungssoftware besitzt, keine Ausbildung in der Softwareentwicklung absolviert hat und mit der Erstellung von Fragebögen vertraut ist.

5.2 Systemaufgaben

Der folgende Abschnitt beschäftigt sich mit den Aufgaben, die an die Datenstreukur gestellt werden. Es werden alle zuvor angesprochenen Funktionen in konkrete Anforderungen formuliert.

Die Priorisierung der Anforderungen wird im Sinne der MoSCoW-Priorisierung [48] durchgeführt. Die MoSCoW-Priorisierung ist eine Methode, die im Bereich des Projektmanagements erfolgreich zur Priorisierung der Anforderungsumsetzung anhand ihrer Wichtigkeit eingesetzt wird.

Nach MoSCoW wird folgender Wertebereich differenziert:

- **MUSS** (eng.: MUST, unbedingt erforderlich)
- **SOLL** (eng.: SHOULD, sollte umgesetzt werden)
- **KANN** (eng.: COULD, kann umgesetzt werden, sollten wichtigere Anforderungen schon abgeschlossen sein)
- **SPÄTER** (eng.: WONT, wird für einen späteren Zeitpunkt vorgemerkt)

5.2.1 Funktionale Anforderungen

In diesem Abschnitt geht es um die konkreten funktionalen Aufgaben, die das System erfüllen soll. Die aufgelisteten Anforderungen werden in nachfolgenden Tabellen priorisiert und beschrieben.

Allgemeine funktionale Anforderungen

Es werden grundsätzliche Anforderungen an die zu entwickelnde Struktur, die eine Abbildung medizinischer Instrumente darstellt, aufgelistet.

BEZEICHNUNG	PRIORITÄT	BESCHREIBUNG
Speichern	*MUSS*	Der Benutzer muss die Möglichkeit haben ein medizinisches Instrument in serialisierter Form für eine spätere Bearbeitung zwischenlagern können.
Laden	*MUSS*	Der Benutzer muss das abgelegte medizinisches Instrument zur weiteren Bearbeitung nach der Speicherung abrufen können.
Löschen	*MUSS*	Der Benutzer muss ein medizinisches Instrument löschen können.
Import	*SOLL*	Der Benutzer soll bereits erstellte Teilabschnitte eines medizinischen Instrumentes zur weiteren Verarbeitung importieren können.
Export	*SOLL*	Der Benutzer soll ein medizinisches Instrument exportieren können. Diese Funktion soll zu Speicherzwecken oder auch zur weiteren Verwendung des Fragebogens in einem externen System benutzt werden können.

Metadaten	*SOLL*	Der Benutzer soll persönliche Daten, sowie ein Erstellungsdatum an ein medizinisches Instrument anhängen können. Das bietet die Möglichkeit einer besseren Datenverwaltung bei steigender Anzahl von Datensätzen.
Validierung	*MUSS*	Der Benutzer hat die Möglichkeit sein Datenobjekt auf Korrektheit zu überprüfen.

Tabelle 5.1: Allgemeine funktionale Anforderungen

Funktionale Anforderungen an die einzelnen Elemente der Struktur

Es werden nachfolgend Anforderungen definiert, die sich direkt auf die verschiedenen Typen von Fragebogenelemente beziehen.

BEZEICHNUNG	PRIORITÄT	BESCHREIBUNG
	Strukturelemente	
Page	*MUSS*	Der Benutzer muss eine Seitenstruktur erstellen können, die es ermöglicht Datenerfassungs- und Beschreibungselemente aufzunehmen.
Start	*MUSS*	Der Benutzer muss einen Startpunkt für ein medizinisches Instrument festlegen können.
End	*MUSS*	Der Benutzer muss einen Endpunkt für ein medizinisches Instrument festlegen können.
AND-Gateway	*MUSS*	Der Benutzer muss gleichberechtigte Pfade erstellen können, welche den gleichen Bedingungen wie dem Ausgangspfad unterliegen.

XOR-Gateway	*MUSS*	Der Benutzer muss sich ausschließende Pfade erstellen können.
LOOP-Gateway	*MUSS*	Der Benutzer muss sich wiederholende Pfade erstellen können.
	Ereigniselemente	
Case	*MUSS*	Der Benutzer muss einem Pfad ein Fallergebnis zuweisen können.
Condition	*MUSS*	Der Benutzer muss einem Gateway eine Bedingung zuweisen können.
Event	*SOLL*	Der Benutzer soll die Möglichkeit haben, einem Pfad ein Event zuweisen zu können.
	Beschreibungselemente	
Text	*MUSS*	Der Benutzer muss einer Seitenstruktur Beschreibungstexte beifügen können.
Custom	*KANN*	Der Benutzer kann benutzerdefinierte Elemente einer Seitenstruktur hinzufügen.
Headline	*MUSS*	Der Benutzer muss Überschriften an eine Seitenstruktur setzten können.
Media	*MUSS*	Der Benutzer muss Medieninhalte in eine Seitenstruktur einbinden können.
	Datenerfassungselemente	

Sensor	*MUSS*	Der Benutzer muss eine Schnittstelle für einen Sensor in die Seitenstruktur einbinden können.
FreeInput	*MUSS*	Der Benutzer muss freie Eingabeelemente in Form von Text-, Zahl- und Dateneingaben, in die Seitenstruktur aufnehmen können.
Slider	*MUSS*	Der Benutzer muss Schieberegler als Mittel zu Datenerfassung in die Seitenstruktur aufnehmen können.
Choice	*MUSS*	Der Benutzer muss Fragen zur Datenerfassung mit vordefinierten Auswahlmöglichkeiten in die Seitenstruktur einbinden können.
Likert	*MUSS*	Der Benutzer muss Fragen zur Messung persönlicher Einstellungen nach dem Likert-Skala Verfahren einbinden können.
Ranking	*MUSS*	Der Benutzer muss eine Datenerfassung mithilfe von Einstufungsfragen einbinden können.

Tabelle 5.2: Funktionale Anforderungen

5.2.2 Nicht-funktionale Anforderungen

In diesem Abschnitt werden die nicht-funktionalen Anforderungen an der Struktur definiert. Diese spezifizieren im Wesentlichen die Qualitätsansprüche an die Struktureigenschaften. Die aufgelisteten Anforderungen werden in nachfolgenden Tabellen priorisiert und beschrieben.

BEZEICHNUNG	**PRIORITÄT**	**BESCHREIBUNG**

Benutzbarkeit	*MUSS*	Die Struktur muss selbsterklärend und übersichtlich sein.
Portierbarkeit	*MUSS*	Das Struktur muss auf unterschiedlichen Plattformen zugänglich sein.
Korrektheit	*MUSS*	Die Struktur muss ein medizinisches Instrument korrekt und vollkommen abbilden können.
Flexibilität	*KANN*	Die Struktur kann auf unterschiedliche Datenobjekte übertragen werden.
Erweiterbarkeit	*MUSS*	Das Struktur muss sich mit wenig Aufwand erweitern oder ändern lassen.

Tabelle 5.3: Nicht-funktionale Anforderungen

6

Architektur

Dieses Kapitel beschäftigt sich mit der grundlegenden Datenstruktur und den Relationen der einzelnen Elemente. Zunächst wird in Abschnitt 6.1 ein Überblick über die Struktur gegeben und geklärt, wie entsprechend Designentscheidungen gefällt wurden. Im Anschluss daran wird in Abschnitt 6.1.1 die entwickelte Struktur anhand Pseudo-UML-Klassendiagramme vorgestellt. In Abschnitt 6.2 wird der Konfigurator vorgestellt, der parallel bei der Entwicklung der Datenstruktur entstanden ist. Dieser Konfigurator zeigt die Anwendbarkeit der Datenstruktur im Sinne eines „Proof of Concept“. Hierzu wird in Abschnitt 6.2 durch das Aufzeigen der Modellvalidität bei der Erstellung eines medizinischen Instruments mithilfe des Konfigurators Stellung genommen.

6.1 Aufbau

Durch das Aufzeigen einer formalen Grammatik soll in diesem Abschnitt der Aufbau und Zusammenhang eines medizinischen Instrumentes (`Questionnaires`) erklärt werden. Die Bereiche eines `Questionnaire`, einer `Group` und einer `Node` werden mithilfe von „`{`“ und „`}`“ abgegrenzt. Verbindungen oder auch `Links` werden über „`,`“ dargestellt. Startpunkt eines medizinischen Instruments ist das Instrument selbst. Auf ein `Questionnaire` folgen immer `Groups`, wie die Abbildung 6.1 zeigt.

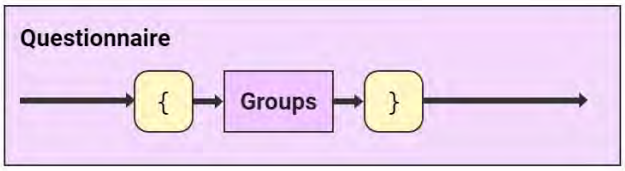

Abbildung 6.1: Formale Grammatik eines Questionnaires

`Groups` lassen sich durch eine `Group` oder die Verkettung von `Group` und `Groups` ausdrücken. Eine `Group` besteht entweder aus `Groups` oder `Nodes`. Durch die Möglichkeit der Einbindung von `Groups` in `Group` können komplexere Gebilde mithilfe mehrerer Verzweigungen entstehen. Der Vorteil einer solchen Anordnungsmöglichkeit wird in den nachfolgenden Kapitel deutlich. In der Abbildung 6.2 wird der Aufbau von `Groups` und einer `Group` grafisch dargestellt.

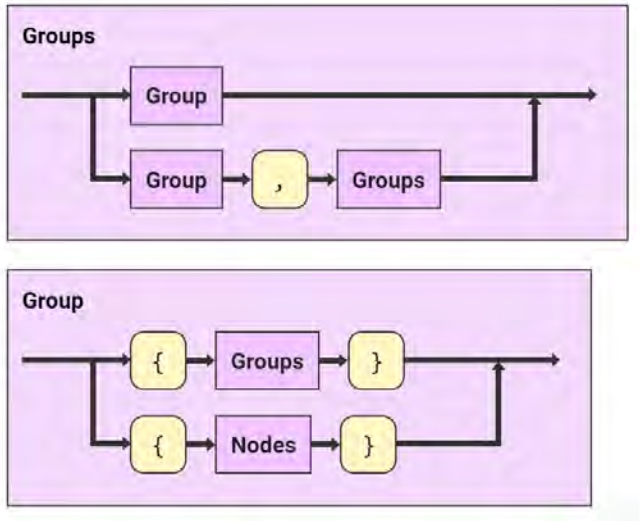

Abbildung 6.2: Formale Grammatik der Groups

Die Abbildung 6.3 beschreibt alle möglichen Zusammensetzungen der `Nodes`. `Nodes` können, wie auch `Groups`, aus einer `Node` oder einer Verkettung von `Node` und `Nodes` bestehen. Durch diese Regel kann eine `Group` eine oder mehrere `Nodes` enthalten. Die formale Grammatik terminiert mit der Auflösung einer `Node`, die eine `Node` mit `Element` zurück liefert.

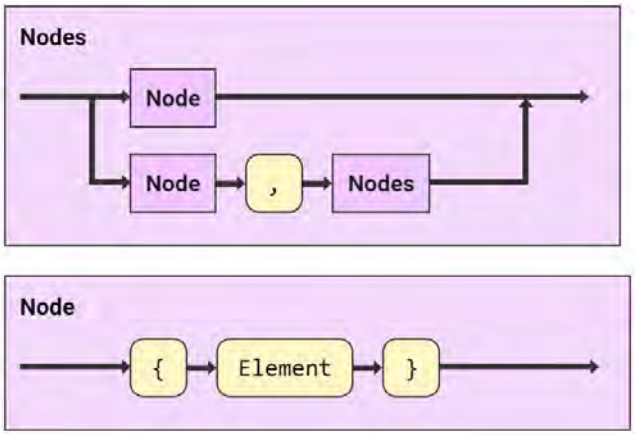

Abbildung 6.3: Formale Grammatik der Nodes

Durch das Relationsschaubild 6.4 werden Zusammenhänge der in Tabelle 2.1 vorgestellten Strukturelemente erläutert. Das Schaubild zeigt von links nach rechts die Zusammensetzung eines validen medizinischen Instruments. Der Einstieg ist definiert durch das `Start`-Element auf das eine oder mehrere `Group`-Elemente folgen können. Es kann sich bei einer `Group`, wie das Schaubild zeigt, auch um das besondere `Page`-Element handeln. Durch die dotierten Bereiche sind in den folgenden Elementen die unterschiedlichen `Gateways`-Gruppen gekennzeichnet. Beginnend mit dem `AND-Gateway`, welches es erlaubt mehrere parallele Pfade zu beschreiten. An dieser Stelle ist es möglich, sowhol beide `Pages` zu durchlaufen, als auch die nächste `XOR-Gateway`-Gruppe zu erreichen. Ein `XOR-Gateway` erlaubt es einem Patienten, anders als die `AND-Gateway`-Gruppe, nur eine Pfad bei der weiteren Ausführung des medizinischen Instrumentes zu beschreiten. Es kann folglich entweder die `Page` durchlaufen werden oder aber das nächste `LOOP-Gateway` betreten werden. Ein `LOOP-Gateway` besitzt die Eigenschaft den Patienten einen definierten Pfad wiederholt durchlaufen zu lassen. Gekennzeichnet wird diese Besonderheit durch die rückläufige Verbindung. Nach durchlaufen des medizinischen Instrumentes bildet ein `End`-Element immer den Abschluss.

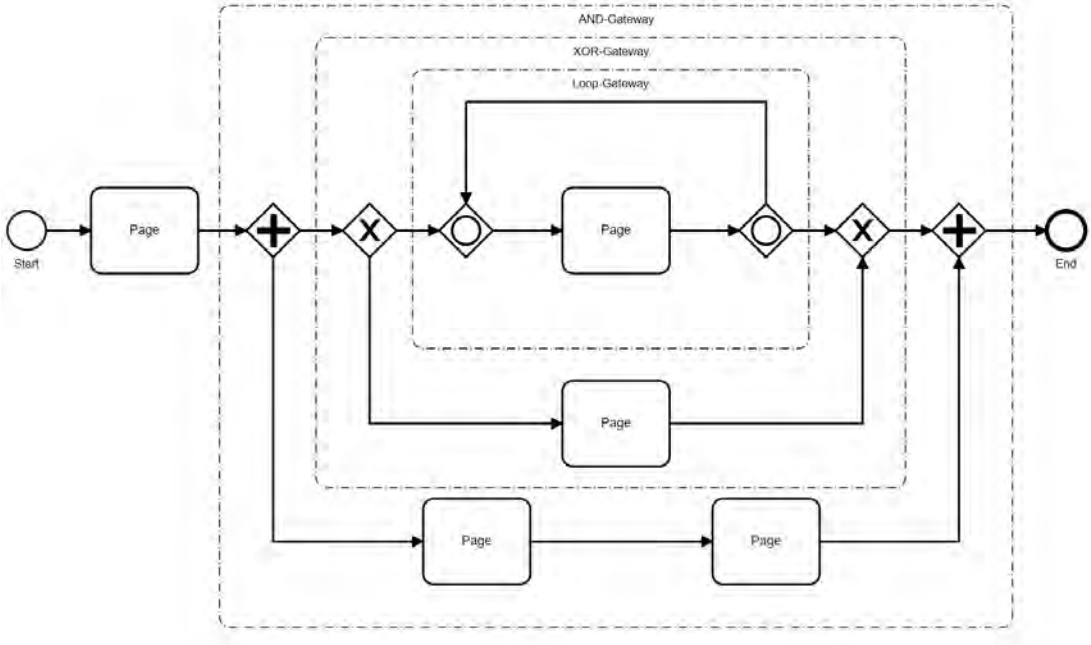

Abbildung 6.4: Relationsschaubild der Strukturelemente [25]

6.1.1 Datenstruktur

Die entwickelte Datenstruktur soll in diesem Abschnitt anhand mehrere UML-Klassendiagramme vorgestellt werden. Die gezeigten UML-Klassendiagramme entsprechen aus Anschauungszwecken nicht der Norm und spiegeln nur den schematischen Aufbau der Struktur wider. Sie werden im Verlauf dieser Arbeit als „Pseudo-UML-Klassendiagramme“ bezeichnet.

Questionnaire

Zentrales Element der Struktur ist das medizinische Instrument (`Questionnaire`). Abbildung 6.5 zeigt den `Questionnaire`, welcher für die Darstellung im Konfigurator von den Klasse `go.Diagram` erbt. Das Klassendiagramm zeigt außerdem die verschiedenen Abhängigkeiten und Zusammenhänge, die für eine medizinisches Instrument bestehen.

Ein `Questionnaire` enthält Metadaten, die sowohl den Ersteller, als auch eine Beschreibung und die Version des Instruments festhalten. Über die Schnittstelle `Changelog` soll es in einer späteren Version dieser Struktur möglich sein Zwischenstände des Objektes wiederherzustellen. Der `Questionnaire` ist außerdem die zentrale Sammelstelle für Medieninhalte. Diese Wahl wurde aufgrund einer besseren Offlineverarbeitung getroffen. Medieninhalte, die in Datenerfassungs- oder Beschreibungselementen eingebunden werden, stellen Verknüpfungen dar, die auf die tatsächlichen Inhalte zeigen. Eines der wichtigsten Eigenschaften bei der Erstellung ist die Verwendungsmöglichkeit verschiedenen Sprachen. Hierfür können mithilfe des Klasse `AvailableLanguages` unterschiedlichste Sprachpakete hinzugefügt werden, die über ein Interface definiert sind. Dem Ersteller eines `Questionnaire` kann es so auch ermöglicht werden eigenen Sprachpakete zu installieren.

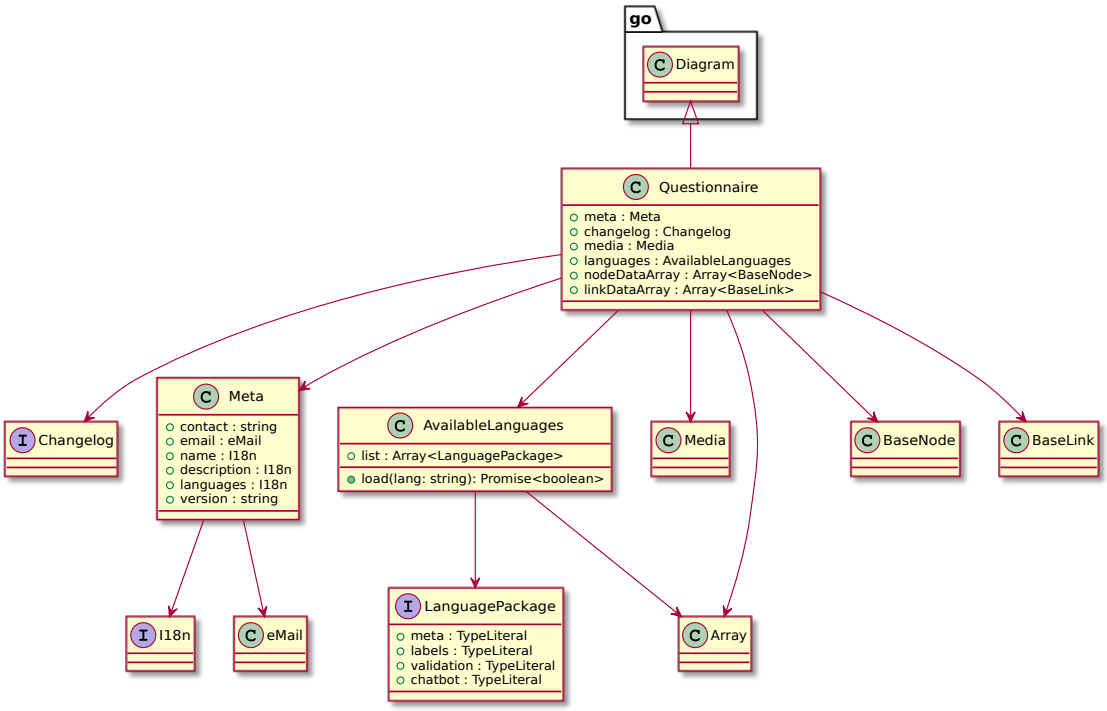

Abbildung 6.5: Questionnaire Pseudo-UML-Klassendiagramm

Wie die Abbildung 6.5 zeigt werden `Links` und `Nodes` über Arrays eingebunden. Das Pseudo-UML-Klassendiagramm in Schaubild 6.6 definiert, wie ein `Link` des `linkDataArray` auszusehen hat.

Link

Ein `Link` dient der Verknüpfung von Elementen aus Tabelle 2.1, die an `Nodes` gebunden sind. Unabhängig von der Art des `Links` besitzt jedes Objekt einen Startpunkt (`from`) und einen Endpunkt (`to`), sowie die Attribute `fromPort` und `toPort`, die den grafischen Start- und Endpunkt bei der grafischen Darstellung im Konfigurator definieren. Über das `category`-Attribut werden alle `Links` kategorisiert, sodass bei einem Verlust der Metadaten einer Klasse die Kategorisierung erhalten bleibt.

Die einzige Ausnahme stellt der `ConditionalLink` dar, welcher ein Ereigniselement, das `CaseElement` binden kann.

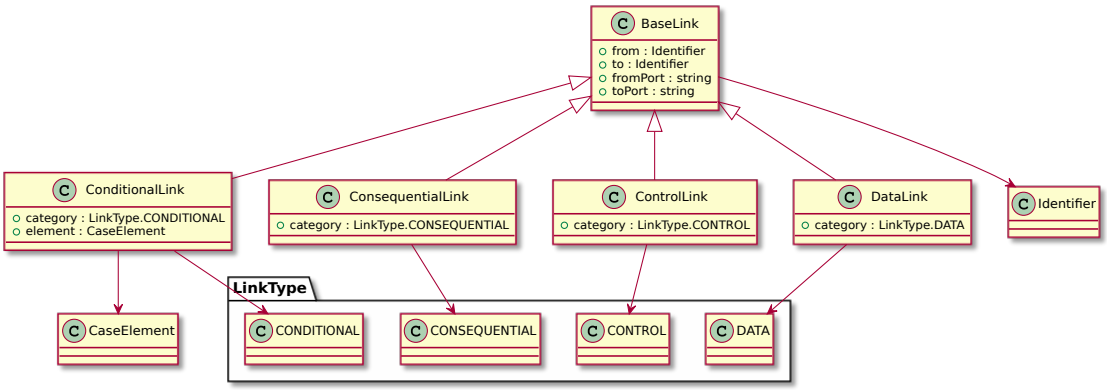

Abbildung 6.6: Links Pseudo-UML-Klassendiagramm

Nodes

Das in Abbildung A.1 gezeigte Pseudo-UML-Klassendiagramm skizziert alle möglichen `Nodes`, die das `nodeDataArray` binden kann.

Basis aller `Nodes` stellt die `BaseNode`-Klasse dar. Sie besitzt einen `key`, der zur eindeutigen Identifizierung dient, eine optionale Gruppenangehörigkeit, ein Element und

eine Kategorie. Die `group`-Eigenschaft ist im Falle einer Element tragender `Node` verpflichtend und dadurch gegeben, dass eine solche `Node` nur zwischen einen `DataLink` eingefügt werden kann. Ein `DataLink` existiert nur innerhalb einer `Page`, somit hat jede `Node`, die ein Element trägt auch automatisch eine Gruppenzugehörigkeit durch eine Referenz auf die `Page` in der Sie sich befindet. Jede `Node`, die von der Klasse `BaseNode` erbt besitzt eine Kategorie, die Sie eindeutig kategorisiert.

Drei `Nodes` bilden in diesem Schaubild eine Ausnahme. Die `CommentNode` bietet dem Benutzer bei der Erstellung eines medizinischen Instrumentes eine Stütze. Die `TransmitterNode` und `ReceiverNode` stellen Anfang und Ende der Strukturelemente „Gateway“ aus der Tabelle 2.1 dar.

Groups

Jede `Group` in Abbildung 6.8 ist eine `Node` mit der Eigenschaft `isGroup`. Diese Besonderheit erlaubt es einer `Group` mehrere `Nodes` zu umfassen. In Ausnahmefällen können `Groups` weitere `Groups` enthalten. Betroffen ist die `AndGroup`, die `LoopGroup` und die `XorGroup`. Eine solche Kombinationsmöglichkeit erlaubt es komplexere Abfragen zu gestalten.

Bei einer Patientenaufnahme in einem Krankenhaus könnte eine Frage bezüglich des Suchtverhaltens wie Schaubild 6.7 zeigt aussehen.

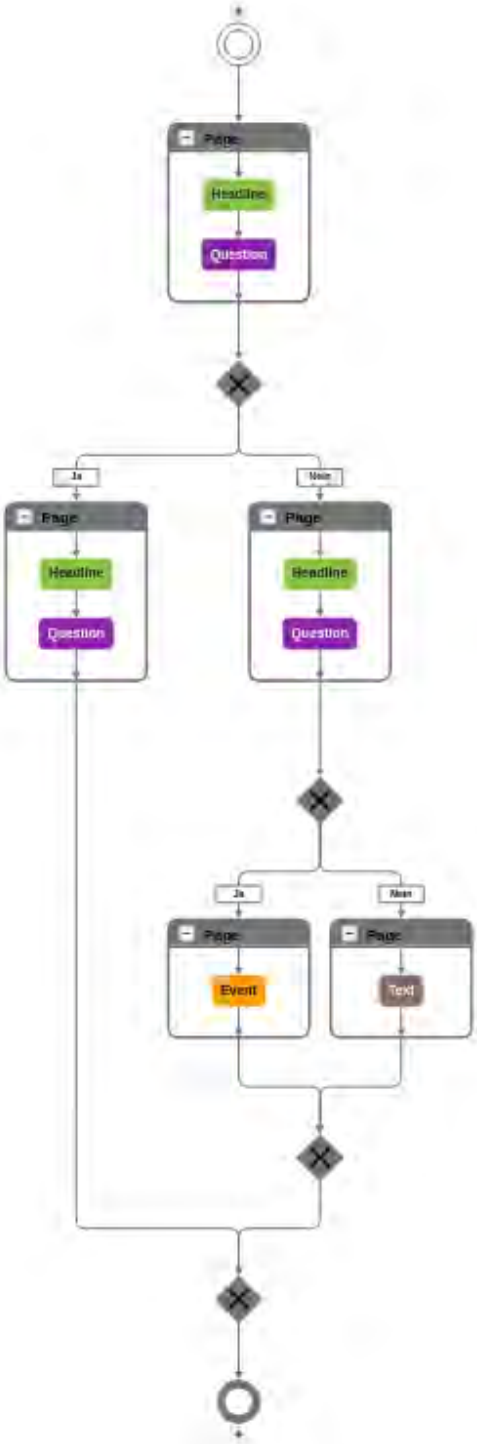

Abbildung 6.7: Fragebogenablauf bezüglich des Suchtverhaltens eines Probanden

Das Schaubild 6.7 besteht aus drei Teilen, in denen ein Patient mehrere Entscheidungen treffen kann. Auf Ebene **1** könnte die Frage nach einer Abhängigkeit eines Substanz stehen. Je nach Ausgang variiert der nächste Schritt. Bei einer Zustimmung der vorherigen Frage werden Einzelheiten bezüglich der Substanz in Ebene **2** erfragt und Vorgang ist damit abgeschlossen. Bei einer Negierung erfolgt eine weitere Entscheidungsfrage, ob Personen im näheren Umfeld Kontakt mit der Substanz hatten. Je nach Ausgang der Entscheidungsfrage in Ebene **2** erfolgt in Ebene **3** entweder ein Event (z.B.: Weiterleitung auf eine Website die zur Suchberatung ausbildet) oder ein Informationstext, welcher präventiv auf die Gefahren der Substanz hinweist.

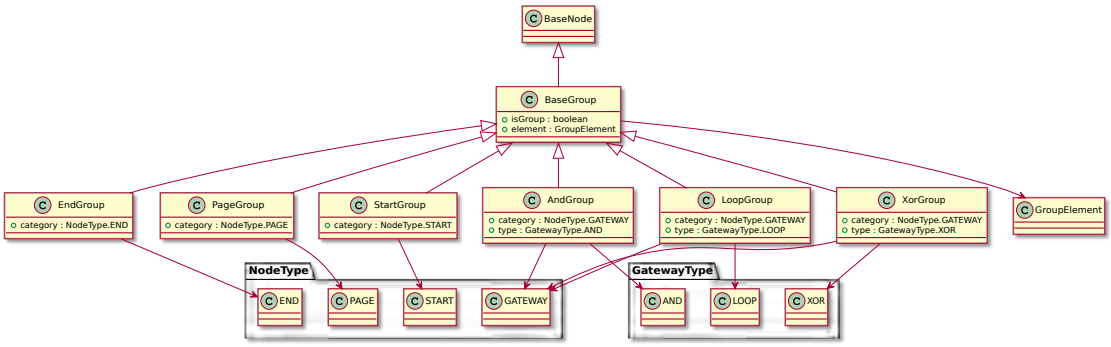

Abbildung 6.8: Groups Pseudo-UML-Klassendiagramm

Weitere Ausnahmen bilden die Gruppen `PageGroup`, `StartGroup` und `EndGroup`. Diese Gruppen können keine weiteren `Groups` enthalten. Das einfügen von `Nodes`, welche die Elemente eines medizinischen Instruments widerspiegeln ist weiterhin möglich. Die `StartGroup` und `EndGroup` sind außerdem durch ihre Typisierung als solche unveränderbar und können nur ein einziges mal in einem medizinischen Instrument (`Questionnaire`) auftauchen.

Elements

Abbildung A.2 zeigt die Struktur der Ereignis-, Beschreibungs- und Datenerfassungselementen. Zu den Ereigniselementen gehören die `CaseElement`, `CoditionElement` und `EventElement` Elemente, wie der Tabelle 2.1 zu entnehmen ist. Die Elemente

TextElement, CustomElement, HeadlineElement und MediaElement sind Teil der Beschreibungselemente.

Jedes Element erbt von der Klasse BaseElement und besitzt somit die title-Eigenschaft. Dieses Attribut dient der Identifizierung eines Elements durch das menschliche Auge.

Ein CaseElement besitzt ein case-Attribut, welches einen möglichen Ausgang eines ConditionElement besitzt. Das ConditionElement enthält zwei Attribute. Das reference-Attribut, welches eine Referenz auf ein vorheriges Ergebnis beinhaltet, und das condition-Attribut, das mithilfe des reference-Attributs eine Bedienung für den weiteren Verlauf des Questionnaires stellt. Ein EventElement enthält eine Action. Eine mögliche Action könnte etwa das versenden einer E-Mail oder der Aufruf einer externen Website sein.

Die Beschreibungselemente dienen der Darstellung von Informationen. Ein HeadlineElement stellt eine Überschrift in einer Page dar. Die Elemente TextElement und CustomElement können Informationen in Form eines Textes oder im Falle des CustomElements eine benutzerdefinierter Form darstellen. Das MediaElement erfüllt diese Aufgabe durch die Darstellung von Bildern und Videos.

Die Datenerfassungselemente, bestehend aus dem SensorElement und QuestionElement, besitzen die Eigenschaft Informationen direkt oder indirekt von einem Patienten zu gewinnen. Die Datenerfassung und Anbindung eines Sensors über ein SensorElement wird in dieser Arbeit nicht weiter erläutert. Jedes QuestionElement besitzt ein type-Attribut, das das Element eindeutig identifiziert, ein required-Attribut, welches festlegt ob eine Eingabe verpflichtend ist, eine instruction in I18n-Format und ein result-Attribut, welches abhängig von dem Typ des QuestionElements genauer definiert ist. Die Informationsgewinnung mithilfe eines QuestionElements wird aufgrund des komplexen Umfangs nachfolgen in mehreren Teilabschnitten geklärt.

In Schaubild A.3 werden mehrere konkrete QuestionElements vorgestellt. Anhand des FreeIntQuestion-Elementes kann die Funktionsweise der freien Datenerfassungselementen, unter die ebenfalls die FreeFloatQuestion, die FreeTextareaQuestion, die FreeTextQuestion und die FreeDateQuestion fallen, erklären. Die

`FreeIntQuestion` besitzt neben den vererbten Attributen eine `unit`-Eigenschaft, welche die Einheit in der die Angaben gemacht werden, festlegt. Das `placeholder`-Attribut wird als Platzhalter in `I18n`-Format verwendet. Eine Einschränkung des Ergebnisses wird durch die Attribute `min` und `max` sichergestellt. Das letzte Attribut `result` beinhaltet das Ergebnis in typabhängiger Form. Eine Ausnahme stellt das `MatrixQuestion`-Element dar, welches Eingaben in Matrixform über `Items` zulässt. Ein `Item`, bestehend aus einem `key` und einem `value`, bildet eine eindeutig zuweisbare Informationseinheit. Jede Reihe und jede Spalte einer `MatrixQuestion` wird mit einem `Item` belegt. Durch diese Zuweisungen lassen sich die Ergebnisse in mehreren `ResultMatrixItem`-Elementen festhalten, die zur Referenzierung sowohl die Spalte, als auch die Zeile als Schlüssel verwenden.

In Abbildung A.4 sind konkrete `QuestionElements` zu sehen, welche die Eigenschaften des `AbstractItem`-Elementes übernehmen. Zur genauen Erläuterung des Aufbaus eins `AbstractItem`-Elementes wird exemplarisch das `DropdownQuestion`-Element herangezogen. Die `DropdownQuestion` besitzt neben der Typisierung eine Liste von `Items`, welche zuvor erläutert wurden. Diese Liste spiegelt die Auswahlmöglichkeiten wieder, die dem Patienten zur Verfügung stehen und durch den Benutzer oder behandelnden Arzt erzeugt werden können. Ein `placeholder`-Attribut nimmt vor der Auswahl den Platz der Anzeige ein. Das Ergebnis wird in einer weiteren Liste von `Items` festgehalten. Durch die Verwendung einer Liste können mehrerer Antwortmöglichkeiten zugelassen werden.

Die in Abbildung 6.9 gezeigte Kategorie wird bestimmt durch die Oberklasse `AbstractSliderQuestion`. Jedes der zwei `Elements`, die von der Klasse `AbstractSliderQuestion` erben, besitzt die Attribute `startNumber`, `endNumber`, `stepSize`, `unit`, `type` und `result`. Die `startNumber` und `endNumber` bestimmen ein Intervall in welchem mit dem Abstand, der durch `stepSize` gegeben ist, traversiert werden kann. Die Eigenschaft `unit` definiert dabei die Einheit um die es sich handelt. Das `SliderRangeQuestion`-Element unterscheidet sich von dem `SliderSingleQuestion` durch die Angabe des Ergebnisses. Während ein `SliderSingleQuestion` als Ergebnis nur eine Wert innerhalb des angegebenen Bereichs wiedergeben kann, wir bei der `SliderRangeQuestion` eine Liste von `Items` als Ergebnis erwartet. Diese Liste kann durch

eine beschränkende Validierungsfunktion nur aus zwei Einträgen bestehen. Dadurch lässt sich das Ergebnis als Sektor in einem Intervall angeben.

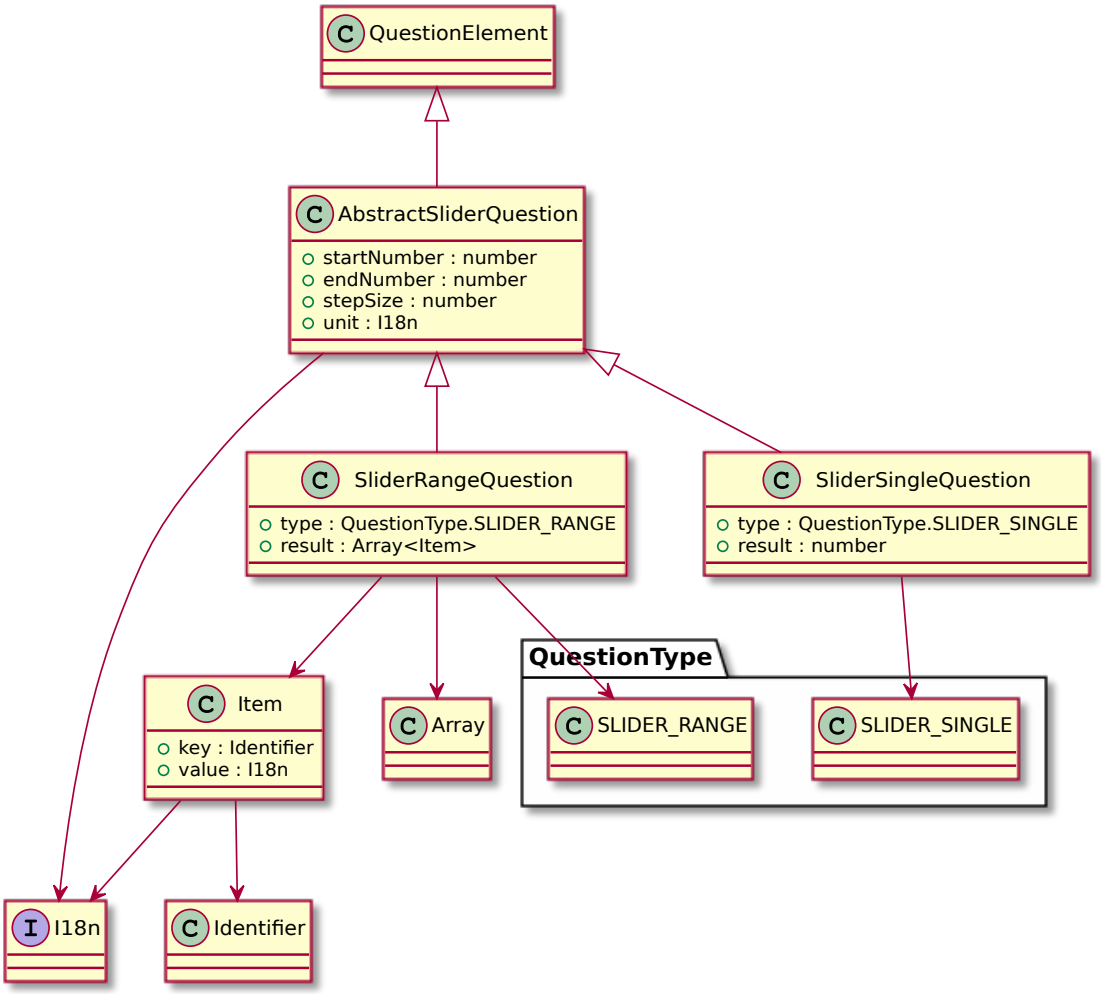

Abbildung 6.9: Slider-Question-Elements Pseudo-UML-Klassendiagramm

Die Gruppe der `AbstractImageQuestion` bildet das Schlusslicht der `Question-Elements`. Die abstrakte Klasse `AbstractImageQuestion` stellt als einziges Attribut eine `mediaId` zur Verfügung. Die `mediaId` enthält eine Referenz auf das `Media`-Objekt, welches in Abbildung 6.5 gezeigt wurde. Mithilfe der Referenz lassen sich aus dem `Media`-Objekt die hinterlegen Medieninhalte herausziehen und können bei Bedarf angezeigt werden. Durch die `instruction`-Eigenschaft, die jedes `QuestionElement`

besitzt, können Fragen bezüglich des Medieninhaltes gestellt werden. Eine plausible Anweisung könnte die Frage „Wo ist Walter?“ sein, wie es in einer Kundebuchreihe von Martin Handford heißt. Die Wahl des Elementes fällt in diesem Fall auf das `ImagePointAndClickQuestion`-Element, welches als Ergebnis einen `Vector` erwartet. Ein `Vector` spiegelt genau einen Punkt in einem rasterförmigen Medieninhalt wider. Um einen Bereich in einem Bild zu beschreiben wird ein `ImagePointAndDrawQuestion`-Element verwendet. Über eine Liste von vier `Vectors` kann ein rechteckiger Bereich in einem Bild wiedergegeben werden. Dieser Struktur ist in Abbildung 6.10 zu sehen und wird durch die Typisierung des `result` angegeben. Für dei `ImagePointAndClickQuestion` besteht das `result` nur aus einem `Vector`, bei der `ImagePointAndDrawQuestion` hingegen können über einen Array mehrere `Vector`-Elemente angegeben werden.

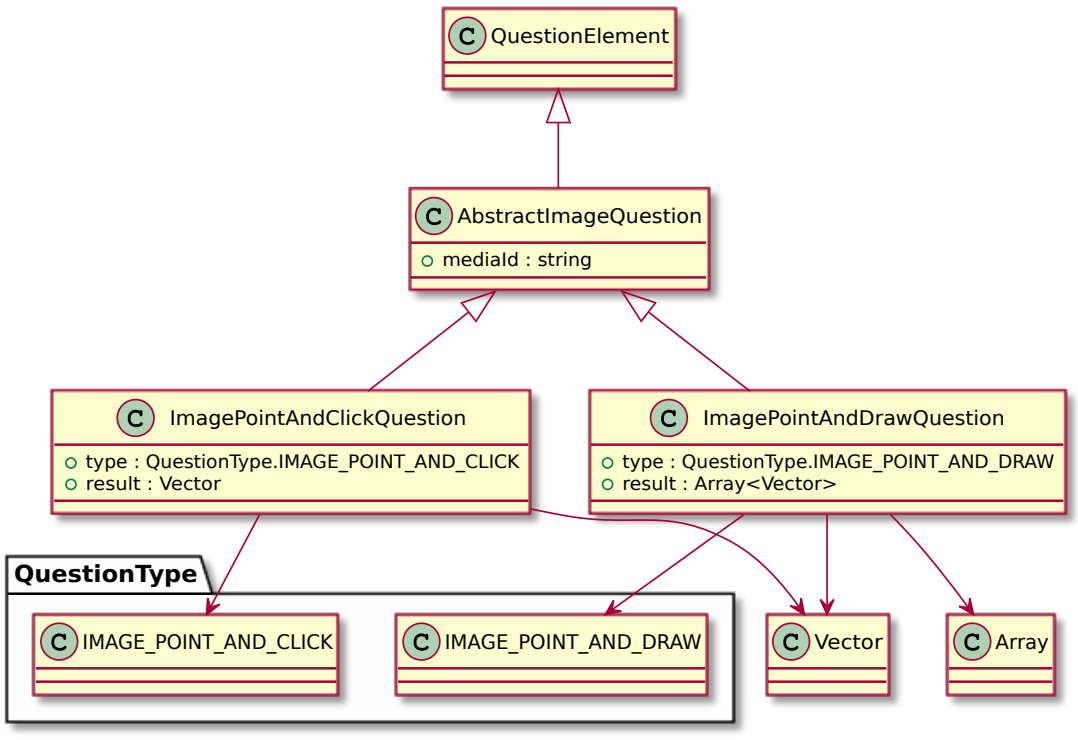

Abbildung 6.10: Image-Question-Elements Pseudo-UML-Klassendiagramm

6.2 Konfigurator

In diesem Abschnitt geht es um den grafischen und strukturellen Aufbau des Konfigurators, welcher bei der Entwicklung der Struktur für medizinische Instrumente entstanden ist. Er dient neben der grafischen Darstellung eines medizinischen Instrumentes auch der Beweisführung für eine erfolgreich entwickelte Datenstruktur.

Das Konzept hinter dem Konfigurator ist in großen Teilen den Arbeiten [38, 39] entnommen. Die dort durchgeführte Studie bieten eine gute Grundlage für den Aufbau und die Benutzbarkeit eines Konfigurators zur Erstellung medizinischer Instrumente.

Die Abbildung 6.11 zeigt die drei Bereiche in die der Konfigurator aufgeteilt ist. Der linke Bereich bietet dem Benutzer die Möglichkeit die Elemente, die in Tabelle 2.1 vorgestellt wurden, über eine Drag-and-Drop-Aktion in den mittleren Bereich zu ziehen. Der mittlere Bereich des Konfigurators ermöglicht es einzelne Elemente zu kombinieren und zeigt den gesamten Graphen, der eine medizinischen Instrument abbildet, an. Durch das Auswählen eines Elementes im mittleren Bereich, kann das Element im rechten Teil des Konfigurators bearbeitet und gespeichert werden. Es handelt sich bei dem rechten Teil um ein Maske in Form eines HTML-Formulars, welches eine Manipulation der vorgestellten Elemente in Tabelle 2.1 zulässt.

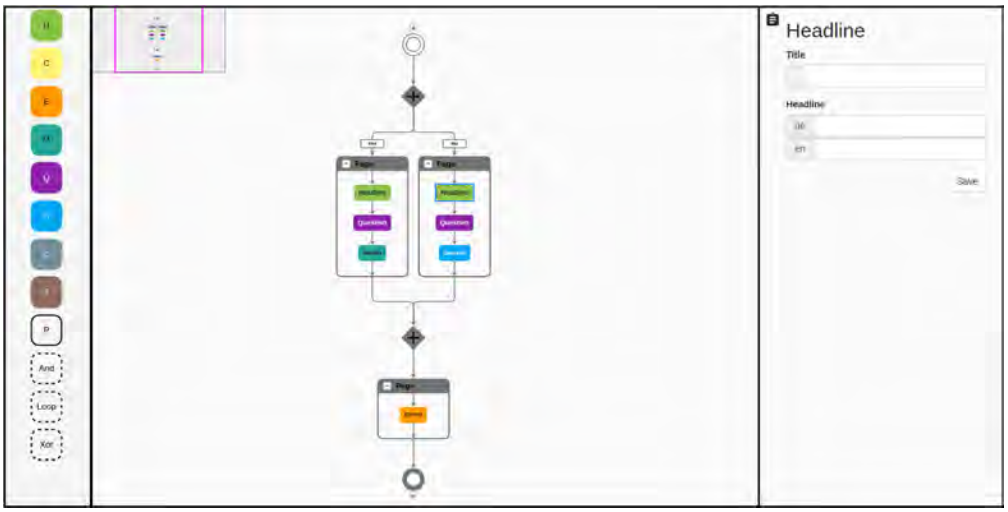

Abbildung 6.11: Grafische Oberfläche des Konfigurators

Der Fluss eines medizinischen Instrumentes verläuft im Konfigurator vertikal. Eine Ausnahme bilden die Gateways, eine Untergruppe der Strukturelemente. Ihre Einfluss ist über eine horizontale Ausprägung des Graphen gezeichnet und unterstreicht so, deren Eigenschaften den Verlauf des Pfades zu ändern.

Validierung

Für die erste Phase der Validierung wurde ein „Correctness by Construction“-Programmieransatz gewählt, welcher in der Arbeit [16] ausführlich beschrieben ist. Dieser Ansatz ermöglicht es schon bei der Erstellung eines neuen medizinischen Instrumentes mithilfe des Konfigurators Fehler zu vermeiden. Durch die Typisierung der einzelnen Elemente, wie zum Beispiel die Abbildung 6.6 zeigt, können Kombinationsmöglichkeiten der Elemente bei der Konstruktion eines medizinischen Instrumentes ausgeschlossen, bzw. erlaubt werden. Wie die Abbildung 6.4 zeigt muss jedes Instrument ein `Start`- und `End`-Element enthalten, die durch einen `Link` verbunden sind. Um den Ansatz „Correctness by Construction“ verfolgen zu können sind alle Aktion, die das Einfügen eines neuen Elementes zulassen nur auf `Links` erlaubt. Damit ist die Validierung auf Seiten des Konfigurators grundsätzlich abgeschlossen. Eine Kombinationsmöglichkeit von einem Element und einem `Link` wird über den jeweiligen Typ bestimmt. Für eine bessere Anschauung kann exemplarisch das Einfügen eines `Text`-Elementes, welches den Typ `NodeType.TEXT` besitzt herangezogen werden. Ein solches Element lässt sich durch eine Regel nur in eine `Page` einfügen, welche per Definition nur `Links` des Typs `LinkType.DATA` enthalten kann. Folglich ist ein Einfügen eines `Text`-Elementes nur dann möglich, wenn es sich bei dem `Link` um einen `DataLink` mit dem Typ `LinkType.DATA` handelt.

7

Ausgewählte Implementierungsaspekte

In diesem Kapitel wird anhand der technischen Umsetzung des Konfigurators auf die entwickelte Datenstruktur eingegangen. Nach der Vorstellung einzelner verwendeter Technologien wird in Abschnitt 7.2 die Umsetzung des Struktur in Form von Klassen erläutert. In Abschnitt 7.3 wird auf die Validierung einzelner Elemente mithilfe der in Abschnitt 7.1.3 erklärten `Decorators` eingegangen. Abschließend werden bestimmte Aspekte der Anwendungslogik erörtert.

7.1 Verwendete Technologien

Aufgrund der Verteilung und Einsatzmöglichkeiten der Struktur fällt die Wahl des Datenformats auf JSON. Diese Wahl ergab sich durch die Untersuchung in Kapitel 4. Die verarbeitende Webanwendung wird wegen ihrer Plattformunabhängigkeit und des objektorientierten Ansatzes in TypeScript verfasst. In diesem Abschnitt werden die wesentlichen Technologien und Frameworks beschrieben, die zum Bau der Datenstruktur und des Konfigurators eingesetzt wurden.

7.1.1 GoJS

GoJS ist eine Bibliothek für JavaScrip und Typescript des Northwoods Software® Unternehmens. Diese Bibliothek erlaubt es in wenigen Schritten interaktive Diagramme und Graphen zu zeichnen. Die GoJS Bibliothek verwendet hierfür ein HTML Canvas Element, das es erlaubt Vektorgrafiken zu zeichnen. Es werden außerdem einige vorde-

finierte Layouts bereitgestellt, die es ermöglichen in nur wenigen Schritten einfachste Diagramme oder Graphen zu skizzieren [9].

In Bezug auf diese Arbeit sind drei Typen der `GraphObjects` von besonderem Interesse. Die `Node`, welche zur grafischen Abbildung von Ereigniselemente, Beschreibungselemente und Eingabeelemente zum Einsatz kommt. Die `Group`, die als Grundlage der Strukturelemente dient. Der `Link`, welcher den Datenfluss und die Validität der Struktur bei der Makroprogrammierung durch den Benutzer sicherstellt.

7.1.2 Angular

Angular ist ein auf TypeScript basiertes Font-End-Webanwendungsframework. Durch die Verwendung von Komponenten als Architekturkonzept ermöglicht Angular eine individuelle Strukturanpassung für jedes Projekt. Der in dieser Arbeit entstandene Konfigurator profitiert von dieser Eigenschaft. für jede Gruppe von Elementen konnten eigene Module, und für jedes Element eigenen Komponenten mit spezialisierten Templates angelegt werden [20].

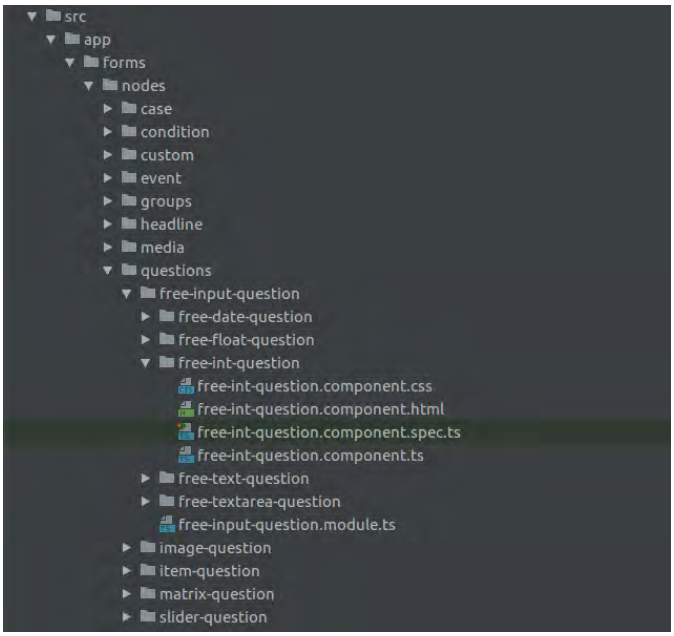

Abbildung 7.1: Projektstruktur bei einem komponentenbasierten Ansatz

Die Abbildung 7.1 zeigt einen kleinen Ausschnitt der Struktur des Konfigurators und soll deutlich machen, wie der Einsatz der Komponenten entscheidend für die Architektur des Projektes war. Die entstandene Anwendungsstruktur ermöglicht ein einfache Erweiterung des Systems. Um eine neue `free-input-question` hinzuzufügen, muss lediglich eine neue Komponente erstellt und an dem Modul (`free-input-question-module.ts`) angemeldet werden.

7.1.3 class-validator

„class-validator“ ist eine JavaScript-Bibliothek, die es erlaubt mithilfe von Decorators Instanzen von Klassen zu validieren. Dieses Verfahren ermöglicht es Model und Logik voneinander zu trennen. Die Klassifizierung eines Attributs findet durch die Kennzeichnung mit den Decorators statt.

Vordefinierte Funktion dienen zur Validierung primitiver und einfacher Datentypen. Über benutzerdefinierte Funktion lassen sich komplexere Vergleichsoperationen abbilden [4].

Decorators

Die Sprache TypeScript erlaubt es mehrere Arten von `Decorators` zu verwenden. In dieser Arbeit geht es insbesondere um die „Property Decorators“. Ein „Property Decorator“ wird vor einem Klassen-Attribut deklariert. Zur Laufzeit einer Anwendung wird der Decorator als eine Funktion mit zwei Parametern aufgerufen. Der erste Parameter ist der Prototyp der Klasse für eine Instanz und der zweite der Name des Attributs. Auf diese Weise lassen sich zum Beispiel Information über das Attribut aufzeichnen. Diese Art von `Decorators` wird auch für eine Typ-Validierung zur Laufzeit verwendet. Das Listing 7.1 zeigt eine Beispielklasse mit einem Attribut „property“ [7].

```
class Example {

  @IsString()
  property: string;

}
```

Listing 7.1: Property Decorators

Während der Laufzeit lässt sich eine Instanz der Klasse `Example`, die in Listing 7.1 abgebildet ist, über die Funktion `validate()` validieren. Bei der Validierung wird die Decorator-Funktion `IsString()` ausgeführt, welche die Referenz dieser Instanz und den Namen des Attributs übergeben bekommt. Im Anschluss kann überprüft werden, ob es sich bei dem Attribut „property“ um einen String handelt.

7.2 Modelle

Die in dieser Arbeit entstandene Datenstruktur wird in diesem Abschnitt in Klassen abgebildet, welche die Grundlage für sowohl die Makroprogramming-Anwendung, als auch die clientseitige Anwendung sind.

7.2.1 Questionnaire

Die in Listing 7.2 gezeigte Klasse bildet das Grundgerüst der Struktur. Das `meta` Attribute hält neben den Informationen des Autors auch die verfügbaren Sprachen des `Questionnaires`, bzw. des Instruments bereit.

Die Klasse `Questionnair` erbt von der Klasse `go.Diagram`, welche die `Nodes` und die `Links` in zwei Arrays mitführt.

```
class Meta {

  @IsString()
  contact: string;

  @IsEmail()
  email: string;

  @IsI18n()
  name: I18n;

  @IsI18n()
  description: I18n;

  @IsI18n()
  languages: AvailableLanguages;

  @IsString()
  version: string;
}

export class Questionnaire extends go.Diagram {

  @ValidateNested()
  meta: Meta;

  @IsChangelog()
  changelog: Changelog;

  @IsMedia()
  media: Array<Media>;

  @ValidateNested({each:true})
  nodeDataArray: Array<BaseNode>

  @ValidateNested({each:true})
  linkDataArray: Array<BaseLink>

}
```

Listing 7.2: `Questionnaire`-Klasse

7.2.2 Nodes

Für die Darstellung eines Elements innerhalb des Konfigurators werden die Elemente an eine `Node`, wie sie durch das Framework GoJS definiert ist, geheftet. Zum besseren Verständnis werden `Groups` bei der weiteren Erklärung nicht betrachtet. Im Grunde handelte es sich bei `Groups` um `Nodes`, welche die Eigenschaft besitzen weitere GoJS-Elemente, inklusive Nodes, zu umschließen.

Das Listing 7.3 zeigt die Eigenschaften einer `Node`, die zu jedem Zeitpunkt existieren müssen. Das Attribut `key` stellt sicher, dass jedes Element einzigartig ist. Über das `group`-Attribut kann festgestellt werden, ob sich eine `Node` innerhalb einer Gruppe befindet. Die wichtigste Eigenschaft `element` der `BaseNode`-Klasse bindet ein Element von einem medizinischen Instrument. Jede `Node` muss außerdem eine Kategorie (`category`) besitzen. Dieses Attribut dient der Identifizierung nach einer Serialisierung und der Klassifizierung bei der Darstellung durch den Konfigurator.

```
1  export abstract class BaseNode {
2
3    @IsIdentifier()
4    key: Identifier;
5
6    @IsOptional()
7    @IsNumberString()
8    group: Identifier;
9
10   @ValidateNested()
11   abstract element: BaseElement;
12
13   @IsEnum(NodeType)
14   abstract category: NodeType;
15
16 }
```

Listing 7.3: `BaseNode`-Klasse

Die in Listing 7.4 abgebildete Klasse vertieft die abstrakte Klasse `BasNode` durch die eindeutige Kategorisierung als `NodeType.QUESTION`. Es wird ebenfalls festgelegt, dass eine `QuestionNode` nur Elemente vom Type `QuestionElement` binden darf.

```
1 export class QuestionNode extends BaseNode {
2
3   @IsEnum(NodeType)
4   category = NodeType.QUESTION;
5
6   @ValidateNested()
7   element: QuestionElement;
8 }
```

Listing 7.4: `QuestionNode`-Klasse

7.2.3 Links

Ein `Link` hat die Fähigkeit zwei `Nodes` oder `Groups` zu verbinden.

Des weiteren besteht durch die Kategorisierung der `Links` die Möglichkeit die Validität eines erstellten Diagramms zu überprüfen. Wie in Abschnitt 6.2 gezeigt wurde, können nur bestimmte `Nodes` Start- oder Endpunkte eines `Links` sein. Bevor eine `Node` in die Diagrammstruktur des Konfigurators eingefügt werden kann, wird der Type des `Node` auf Kompatibilität mit dem Typ des `Links` getestet.

Listing 7.5 zeigt die Beschaffenheit eines `BaseLink`. Die beiden Attribute `from` und `to` definieren die Start- und Endpunkte mithilfe des `key`-Attributs einer `Node`. Ein `Link` besitzt selber keinen `key`, da er durch Start- und Endpunkt eindeutig identifiziert werden kann.

```
1  export abstract class BaseLink implements go.ObjectData {
2
3    @IsIdentifier()
4    from: Identifier;
5
6    @IsIdentifier()
7    to: Identifier;
8
9    @IsEnum(LinkType)
10   abstract category: LinkType;
11
12 }
```

Listing 7.5: `BaseLink`-Klasse

Eine exemplarische Spezifizierung eines `Links` ist in Listing 7.6 zu sehen. Das Attribut `category` identifiziert jede Instanz dieser Klasse eindeutig mit einem `Link` vom Typen `LinkType.CONDITIONAL`.

Die Klasse `ConditionalLink` besitzt außerdem noch ein Besonderheit, das `element` Attribut. Diese Besonderheit dient der Fallunterscheidung, wie es in der Programmierung eingesetzt wird (Switch-Case-Anweisung). Ein `ConditionalLink` kommt immer dann zum Einsatz, wenn ein XOR-oder LOOP-Gatway aus den Strukturelementen in Tabelle 2.1 verwendet wird. Im Falle eines XOR-Gateways sind alle ausgehenden `Links` der Transmitternode von der Klasse `ConditionalLink` und besitzen ein `CaseElement`. Jeder dieser `ConditionalLinks` hält einen Fall bereit für die Bedingung, die in der `TransmitterNode` definiert ist. Dieses Vorgehen hat den Vorteil bei der Pfadüberprüfung nicht jede `Node`, auf die ein `Link` zeigt, aufrufen zu müssen.

```
export class ConditionalLink extends BaseLink {

  @IsEnum(LinkType)
  category = LinkType.CONDITIONAL;

  @ValidateNested()
  element: CaseElement;

}
```

Listing 7.6: `ConditionalLink`-Klasse

7.2.4 Elements

Jedes Objekt der in Auflistung 7.3 gezeigten Klasse besitzt ein standardisiertes Element, welches in Listing 7.7 zu sehen ist. Ein Basiselement besitzt nur ein einziges Attribut, den Titel. Der Title dient zur besseren Differenzierbarkeit für den Anwender.

```
export abstract class BaseElement {

  @IsString()
  title: string;

}
```

Listing 7.7: `BaseElement`-Klasse

Ein `QuestionElement`, dessen Klasse in Auflistung 7.8 zu sehen ist, erweitert das Basiselement um vier weitere Attribute. Das Attribut `type` definiert die Art des Elementes. Diese redundante Art der Typisierung dient der eindeutigen Zuordnung des Elementes bei einem Serialisierungsvorgang. Das Kennzeichen `required` stell fest, ob es sich bei diesem Element um ein Pflichtelement handelt. Die `instruction` enthält mögliche Anweisung die zur Beantwortung der Frage nötig sind. Durch die Verwendung des I18n-Typs kann die Anweisung in beliebig vielen Sprachen verfasst werden. Das letzte Merkmal `result` hält das typspezifische Ergebnis fest.

```
export abstract class QuestionElement extends BaseElement {

  @IsEnum(QuestionType)
  abstract type: QuestionType;

  @IsBoolean()
  required = false;

  @IsI18n()
  instruction: I18n = {};

  abstract result;

}
```

Listing 7.8: `QuestionElement`-Klasse

Listing 7.9 zeigt eine spezifizierte `QuestionElement` Klasse. Die `FreeIntQuestion` wird zusätzlich zum Typ der Klasse durch das `type` Attribut definiert. Für die Datenerhebung über ein Endgerät eines Benutzers besitz eine Instanz der Klasse `FreeIntQuestion` die Attribute: `unit`, `placeholder`, `min` und `max`. Das Attribut `unit` bestimmt die Einheit, in der ein Benutzereingabe gemacht werden soll. Ein Platzhalter wird über das Attribut `placeholder` bestimmt. Sowohl `unit`, als auch `placeholder` ist vom Typ I18n. Für das Anzeigen eines Platzhalters in der deutschen Sprache lautet des Aufruf **`new`** `FreeIntQuestion.placeholder['de']`. Voraussetzung hierfür ist, dass das benötigte Sprachpaket vorhanden ist.

```
export class FreeIntQuestion extends QuestionElement {

  @IsEnum(QuestionType)
  type = QuestionType.FREE_INT;

  @IsI18n()
  unit: I18n = {};

  @IsI18n()
  placeholder: I18n = {};

  @IsNumber()
  min: number;

  @IsGreaterOrEqual('min')
  @IsNumber()
  max: number;

  @IsGreaterOrEqual('min')
  @IsLessOrEqual('max')
  @IsNumber()
  @IsResult()
  result: number;

}
```

Listing 7.9: `FreeIntQuestion`-Klasse

7.3 Validierung

In diesem Abschnitt soll der technische Aspekt der Validierung genauer erläutert werden. Es wird gezeigt, wie die Logik der Validierung von den einzelnen Klassen getrennt werden kann. Im Anschluss daran wir anhand eines Beispiels eine solche Validierungsfunktion näher erklärt.

Das in Abschnitt 7.2.4 gezeigte Listing 7.9 besitzt für jedes Attribut der Klasse `FreeIntQuestion` mindestens einen `Decorator`, der in Abschnitt 7.1.3 näher erläutert wurden. Auf diese Weise ist es möglich die Validierungslogik von den Eigenschaften einer Klasse

zu trennen. Der `Decorator IsGreaterOrEqual` in Zeile 15 besitzt die Fähigkeit das Attribut `max` in Zeile 17 einer Instanz der Klasse `FreeIntQuestion` zur Laufzeit zu überprüfen. In diesem Fall ist es möglich zu validieren, ob das dekorierte Attribut gleichgroß oder größer als das übergebene Attribut `min` in Zeile 13 ist.

In Listing 7.10 ist die Logik eines `Decorators` in Form einer Funktion abgebildet. Die Validierungsfunktion befindet sich in Zeile 5 und ist als Funktion an das Attribut `validator` des `Decorators` gebunden. Über den Aufruf in Zeile 7 wird der Wert des zu vergleichenden Attributs referenziert. Im Anschluss daran werden beide Werte des Attributes auf Vergleichskompatibilität geprüft und bei einem Treffer verglichen. Sollte es sich um einen `string` handeln, so werden beider Werte in Zeile 9 hinsichtlich auf die Länge des `strings` geprüft. Handel es sich um eine Zahl oder ein Datum, lautet der Vergleichsoperator `>=`.

```
1  export function IsGreaterOrEqual(...) {
2    return (...) => {
3      registerDecorator({
4        ...
5        validator: {
6          validate(value: any, args: ValidationArguments) {
7            const ref = args.object[args.constraints[0]];
8            if (typeof value === 'string' && typeof ref === 'string') {
9              return value.length >= ref.length;
10           } else if (typeof value === 'number' && typeof ref === 'number' || value
                 instanceof Date && ref instanceof Date) {
11             return value >= ref;
12           } else {
13             return false;
14           }
15         },
16         ...
17       }
18     });
19   };
20 }
```

Listing 7.10: Größer-gleich Validierungsfunktion

7.4 Anwendungslogik

Die Brauchbarkeit der entworfenen Datenstruktur zur Abbildung medizinischer Instrumente lässt sich messen durch die Benutzbarkeit des Konfigurators. Dieser baut auf der entworfenen Datenstruktur auf und gibt dem Benutzer die Möglichkeit die definierten Elemente aus Tabelle 2.1 nach vorgegebenen Regeln zu kombinieren.

Um die Erweiterbarkeit der Datenstruktur in Bezug auf die Anwendung zu gewährleisten wurden `Operations` eingeführt.

Operations

Eine `Operation` definiert was bei einer bestimmten Aktion mit den Elementen passieren soll. Es kann zwischen vier Aktionen unterschieden werden, die auf ein Element angewandt werden können. Ein Element kann hinzugefügt werden (`add`), es kann kopiert werden (`copy`), es kann ausgeschnitten werden (`cut`) oder es kann gelöscht werden (`del`).

In Listing 7.11 wird gezeigt wie die `AddQuestion-Operation` für ein `Question-Element`-Objekt angemeldet werden kann. Wenn ein Benutzer ein Element auf einen `Link` bewegt wird die statische Methode `AddOperation.apply(e, obj)` aufgerufen. Die Variable `diagram` in Zeile 5 referenziert das GoJS-Diagram in dessen Kontext die Aktion ausgeführt wird. In Zeile 6 wird das Objekt bestimmt, in diesem Fall der `Link`, auf den das Element, das in Zeile 7 bestimmt wird, geworfen wird. Im Anschluss daran wird anhand des Typs des Elementes entschieden welche Datentyp spezifische `Operation` ausgeführt werden soll.

```
1  export class AddOperation {
2
3    public static apply(e, obj): void {
4
5      const diagram = e.diagram;
6      const link = obj.part;
7      const node = diagram.selection.first();
8
9      switch (node.category) {
10
11       case NodeType.QUESTION:
12         if (AddQuestion.canPerform(link)) {
13           AddQuestion.perform(diagram, link, node);
14         } else {
15           OperationException.cantPerfom();
16         }
17         break;
18         ...
19     }
20   }
21 }
```

Listing 7.11: `AddOperation`-Klasse

Die `Operations`, welche im Kontext dieser Arbeit in einer Klasse abgebildet wurden, besitzt zwei statische Methoden. Die `canPerform`-Methode und die `perform`-Methode. In Listing 7.12 soll exemplarisch erklärt werden, wie eine solche `Operation` aufgebaut ist.

Die statische `canPerform`-Methode der Klasse `AddQuestion` in Zeile 3 bekommt als Übergabeparameter den `Link` gegeben, auf den die `QuestionNode` geworfen werden soll. Zur Überprüfung ob eine Aktion ausgeführt werden kann oder nicht, wird mithilfe eines Switch-Case-Anweisung die Kategorie des `Links` überprüft. Im Falle des Listings 7.12 kann eine Aktion nur ausgeführt werden, wenn die Kategorie des `Links` vom Typ `LinkType.DATA` ist. Diese Art von `Link` kann nur inerhalb einer `Page` vorkommen und gewährleistet somit die Validität des medizinischen Instrumentes.

Nach der Kontrolle der `canPerfom`-Methode wird, bei eine erfolgreichen Prüfung, die `perform`-Methode aufgerufen. Diese Methode benötigt drei Übergabeparameter. Das

Diagramm, auf dem die Aktion ausgeführt werden soll, den Link, auf den das neue Objekt geworfen wird und die Node, welche das zu werfende Objekt enthält. Die Funktion die ein Einfügen der Node in den Link ermöglicht wird von einer Transaktion in Zeile 13 und 15 des Diagramms umschlossen. Eine solche Transaktion kann diesen Vorgang, durch die Speicherung des vorherigen Zustandes, wieder rückgängig zu machen. Mit dem Aufruf `insertNodeIntoLink(diagram, link, node, DataLink)` wird die Node als Endpunkt des Links auf den geworfen wird festgelegt. Der ehemalige Endpunkt wird ausgehend von der neuen Node mit einem Link der Klasse `DataLink` verbunden, welcher vom Typ `LinkType.DATA` ist.

```
1  export class AddQuestion {
2
3    static canPerform(link): boolean {
4      switch (link.data.category) {
5        case LinkType.DATA:
6          return true;
7        default:
8          return false;
9      }
10   }
11
12   static perform(diagram, link, node): boolean {
13     diagram.startTransaction(this.name);
14     insertNodeIntoLink(diagram, link, node, DataLink);
15     diagram.commitTransaction(this.name);
16     return true;
17   }
18 }
```

Listing 7.12: `AddQuestion`-Klasse

8

Diskussion

In diesem Kapitel geht es darum die entwickelte Struktur auf ihre Benutzbarkeit und ihre Erweiterbarkeit zu testen. Auf diese Weise werden Vor- und Nachteile bei einem praktischen Einsatz der Struktur deutlich.

Einen Beweis für die Erweiterbarkeit der Struktur wird anhand einer Erweiterung um das `FreeTextareaQuestion`-Element geführt, das sowohl in die Struktur, als auch in den Konfigurator eingebunden wird. Das Listing 8.1 zeigt die Struktur des neuen `FreeTextareaQuestion`-Elementes. Da es sich bei diesem Element um eine konkretisiertes `QuestionElement` handelt, muss die Klasse von `QuestionElement` erben. Dies geschieht in Zeile 1 durch den Befehl **`extends`** `QuestionElement`. Es wird außerdem ein neuer Typ erzeugt, der dieses Element eindeutig bestimmt. In Zeile 4 wird der Typ der Klasse und jeder daraus erzeugten Instanz zugewiesen. Alle weiteren Attribute beziehen sich auf die Eigenschaften des neuen Elementes. Zeile 7 weist dem `FreeTextaraeQuestion`-Element einen Platzhalter zu und gibt dem Benutzer die Möglichkeit die Länge des Textes mit `minLength` und `maxLenght` einzuschränken. In Zeile 22 kann das Ergebnis des Patienten festgehalten werden. Weitere Schritte sind bei der Erweiterung der Datenstruktur nicht notwendig.

```
export class FreeTextareaQuestion extends QuestionElement {

  @IsEnum(QuestionType)
  type = QuestionType.FREE_TEXT_AREA;

  @IsI18n()
  placeholder: I18n = {};

  @IsPositive()
  @IsNumber()
  minLength: number;

  @IsGreaterOrEqual('minLength')
  @IsPositive()
  @IsNumber()
  maxLength: number;

  @IsGreaterOrEqual('minLength')
  @IsLessOrEqual('maxLength')
  @IsString()
  @IsResult()
  result: string;

}
```

Listing 8.1: `FreeTextareaQuestion`-Klasse

Um die Auswirkungen einer Erweiterung des Struktur auf mögliche Anwendungssysteme zu untersuchen, wird der in dieser Arbeit entstandene Konfigurator ebenfalls um das `FreeTextareaQuestion`-Element erweitert. Die Abbildung 8.1 zeigt, dass eine Anpassung der Anzeige der erweiterten Struktur in der Leiste der möglichen Elemente und in der Graphenübersicht nicht notwendig ist. Bei der `FreeTextareaQuestion` handelt es sich um eine Unterklasse der `QuestionElements`, somit ist die grafische Repräsentation bereits gegeben. Für eine Anpassung der grafischen Darstellung bedarf es lediglich der Angabe eines neuen Typs und die Zuweisung einer neuen Farbe für eine bessere Differenzierung.

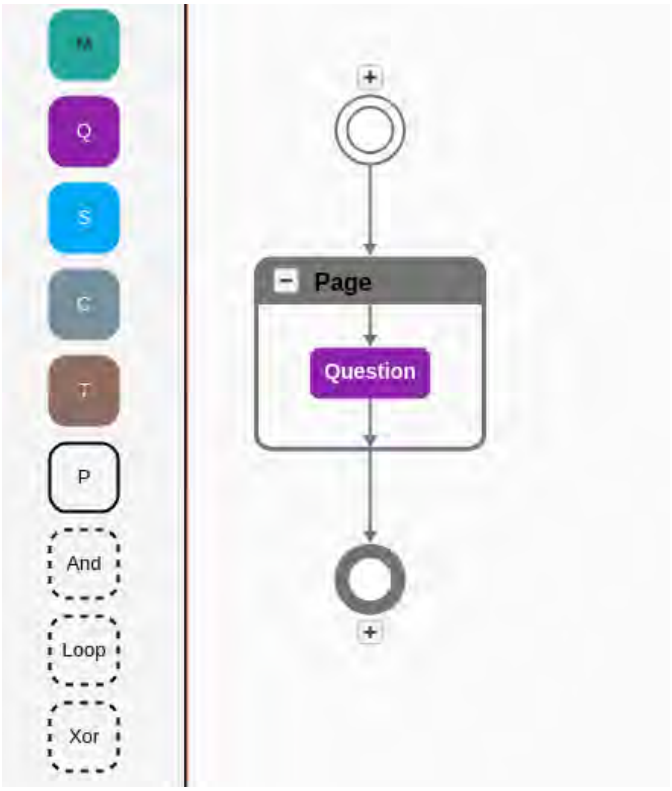

Abbildung 8.1: Grafische Darstellung einer `FreeTextareaQuestion`

Damit dem Benutzer die Bearbeitung des Elementes ermöglicht werden kann muss ein neue Formularmaske erstellt werden. Diese kann aus einigen bereits vordefinierten Elementen zusammengebaut werden, wie Schieberegler, Eingabefelder für Daten, Zahlen und Wörter. Nach dem Erzeugen einer Formularmaske muss diese an der Anwendung angemeldet werden. Dafür wird zuerst eine Zuweisung an das neu Datenerfassungselement `FreeTextareaQuestion` vorgenommen und anschließend wird eine Liste für alle Auswahlmöglichkeiten um die neue Formularmaske erweitert. Über ein Drop-Down-Menü kann im Konfigurator das neu erstelle Element ausgewählt werden, wie das Schaubild 8.2 zeigt.

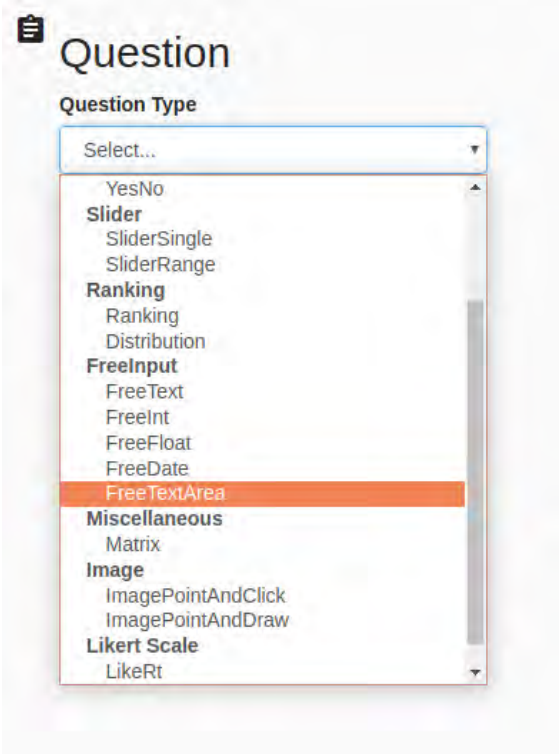

Abbildung 8.2: Einbindung der `FreeTextareaQuestion`

In Abbildung 8.3 ist die Formularmaske für die `FreeTextareaQuestion` zu sehen. Die drei ersten Eigenschaften `Title`, `Required` und `Instruction` sind Basiseigenschaften eines jeden Elementes, das von der Klasse `QuestionElement` erbt. Die Eigenschaften `Placeholder`, `Min Length` und `Max Length` konnten ebenfalls von der schon bestehenden Klasse `FreeTextQuestion` übernommen werden.

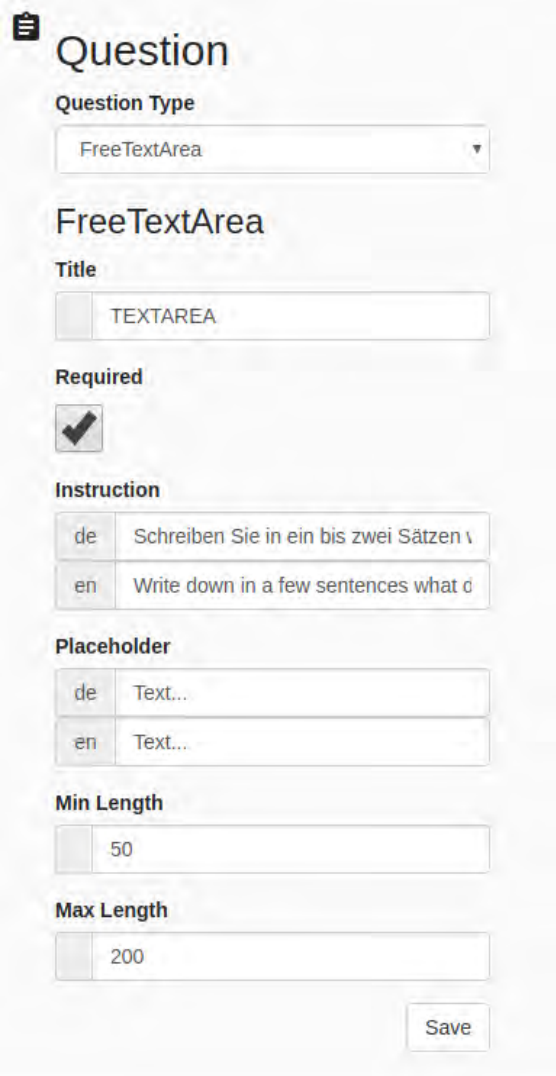

Abbildung 8.3: `FreeTextareaQuestion` Formularmaske

Dieser Vorgang zur Erweiterung der Elemente eines medizinischen Instrumentes hat durch seine Einfachheit gezeigt, dass die Erweiterbarkeit der Struktur gegeben ist. Jedoch benötigt es einen Fachexperten oder einen sehr technisch affinen Benutzer um eine solche Anpassung vorzunehmen.

9

Zusammenfassung und Ausblick

In diesem Kapitel wird die Arbeit zusammenfassend noch einmal aufgeführt. Es werden die wesentlichen Erkenntnisse und die Zielsetzung dieser Arbeit aufgegriffenen, um einen Ausblick auf zukünftige Verwendungsmöglichkeiten zu geben.

Diese Arbeit hat sich mit der Entwicklung einer Struktur zur Abbildung von medizinischen Instrumenten auseinandergesetzt. Dafür wurden zuerst die Probleme, die in vielen anderen Arbeiten aufgetreten sind, analysiert und relevante verwandte Arbeiten näher betrachtet. Ziel dieser Arbeit ist es gewesen aus den gesammelten Erkenntnissen eine Struktur abzuleiten, die es ermöglicht eine medizinisches Instrument in seiner Gesamtheit zu erfassen. In Folge dessen wurden Anforderungen definiert und der Aufbau eines medizinischen Instrumentes systematisch Kategorisiert, um eine digital Abbildung möglich zu machen. Es wurden relevante Stellen des Entwicklungsprozesses eines medizinischen Instrumentes betrachtet, um eine konkrete Vorstellung des Entwicklungsprozesses zu erlangen. Das Resultat dieser Arbeit ist eine Struktur zur Abbildung von medizinischen Instrumenten. Es entstand darüber hinaus eine Konfigurator, welcher es einem Benutzer ermöglicht ein solches Instrument im Sinne der Endbenutzer-Entwicklung zu programmieren.

9.1 Ausblick

Wie aus dem dem Kapitel 8 hervorgeht sind die meisten Anforderungen in der Umsetzung vollständig abgebildet worden. Jedoch stellte sich heraus, dass die Validierung des Datenobjekts in den Bereichen der Sensoren einen größeren Umfang hat, als ursprünglich angenommen wurde. Es war daher nicht möglich eine Schnittstelle zu definieren, die

generisch einen Sensor für eine Datenerfassung einbindet. Es muss außerdem geklärt werden wie Brauchbar die entwickelte Struktur und der daraus resultierende Konfigurator ist. Hierfür könnte eine Studie, wie sie in der Arbeit [38] durchgeführt worden ist, von großem Nutzen sein.

Die Konzeption und Entwicklung einer solchen Struktur lässt sich weiter mit einer automatisierten Auswertung der erfassten Daten koppeln, die der nächste Schritt bei der Weiterführung des Projektes ist. Für eine große Flexibilität muss das System, welches nach Vorgaben des Anwenders die Resultate einer Umfrage selektiert und grafisch darstellt, als unabhängige Erweiterung entwickelt werden. Diese Erweiterung stellt nur eine Möglichkeit dar, wie auf die entwickelte Struktur aufgebaut werden könnte. Als eine weiter Anpassung der Struktur, die im Rahmen dieser Arbeit entstanden ist, könnte die Einbindung und Analyse von sprach basierten Daten folgen. Aufgrund der typsicheren Serialisierung der Struktur und der einfach gehaltenen Architektur bietet die entstandenen Implementierung eine solide Grundlage für weitere Projekte.

Literaturverzeichnis

[1] *CDA-Body – Hl7wiki*. http://wiki.hl7.de/index.php?title=CDA-Body, Abruf: 10.03.2020

[2] *CDA-Header – Hl7wiki*. http://wiki.hl7.de/index.php?title=CDA-Header, Abruf: 10.03.2020

[3] *CDA-Level – Hl7wiki*. http://wiki.hl7.de/index.php?title=CDA-Level, Abruf: 10.03.2020

[4] *class-validator*. https://github.com/typestack/class-validator, Abruf: 29.02.2020

[5] *Data Structures Information Model*. https://specifications.openehr.org/releases/RM/latest/data_structures.html, Abruf: 09.03.2020

[6] *Datatypes-examples - FHIR v4.0.1*. http://hl7.org/fhir/datatypes-examples.html#Attachment, Abruf: 09.03.2020

[7] *Decorators*. https://www.typescriptlang.org/docs/handbook/decorators.html, Abruf: 09.01.2020

[8] *Good Health Clinic Consultation Note*. https://www.w3.org/TR/grddl-primer/hl7-sample.xml, Abruf: 10.03.2020

[9] *Introduction to GoJS Diagramming Components*. https://gojs.net/latest/intro/index.html, Abruf: 29.02.2020

[10] *Ionic Framework 5*. https://ionicframework.com/, Abruf: 10.02.2020

[11] *openEHR Architecture Overview*. https://specifications.openehr.org/, Abruf: 09.03.2020

[12] *Questionnaire - FHIR v4.0.1*. https://www.hl7.org/fhir/questionnaire.html, Abruf: 01.05.2020

[13] *QuestionSys - A Generic and Flexible Questionnaire System Enabling Process-Driven Mobile Data Collection*. https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/questionsys/, Abruf: 02.03.2020

[14] ANHØJ, J. ; MØLDRUP, C. : Feasibility of Collecting Diary Data From Asthma Patients Through Mobile Phones and SMS (Short Message Service): Response Rate Analysis and Focus Group Evaluation From a Pilot Study. In: *J Med Internet Res* 6 (2004), Dec, Nr. 4, e42. http://dx.doi.org/10.2196/jmir.6.4.e42. – DOI 10.2196/jmir.6.4.e42. – ISSN 1438–8871

[15] BENDER, D. ; SARTIPI, K. : HL7 FHIR: An Agile and RESTful approach to healthcare information exchange. In: *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, 2013. – ISSN 1063–7125, S. 326–331

[16] BORDIN, M. ; VARDANEGA, T. : Correctness by Construction for High-Integrity Real-Time Systems: A Metamodel-Driven Approach. In: ABDENNADHER, N. (Hrsg.) ; KORDON, F. (Hrsg.): *Reliable Software Technologies – Ada Europe 2007*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007. – ISBN 978–3–540–73230–3, S. 114–127

[17] DOLIN, R. H. ; ALSCHULER, L. ; BOYER, S. ; BEEBE, C. ; BEHLEN, F. M. ; BIRON, P. V. ; SHABO (SHVO), A. : HL7 Clinical Document Architecture, Release 2. In: *Journal of the American Medical Informatics Association* 13 (2006), 01, Nr. 1, 30-39. http://dx.doi.org/10.1197/jamia.M1888. – DOI 10.1197/jamia.M1888. – ISSN 1067–5027

[18] FISCHER, G. ; GIACCARDI, E. ; YE, Y. ; SUTCLIFFE, A. G. ; MEHANDJIEV, N. : Meta-Design: A Manifesto for End-User Development. In: *Commun. ACM* 47 (2004), Sept., Nr. 9, 33–37. http://dx.doi.org/10.1145/1015864.1015884. – DOI 10.1145/1015864.1015884. – ISSN 0001–0782

[19] FORSTER, D. ; BEHRENS, R. H. ; CAMPBELL, H. ; BYASS, P. : Evaluation of a computerized field data collection system for health surveys. In: *Bulletin of the World Health Organization* 69 (1991), Nr. 1, 107-111. https://pubmed.ncbi.nlm.nih.gov/2054915. – ISSN 0042–9686. – 2054915[pmid]

[20] In: FREEMAN, A. : *JavaScript and TypeScript: Part 1*. Apress. – ISBN 978–1–4842–3649–9, 63–85

[21] GAGGIOLI, A. ; PIOGGIA, G. ; TARTARISCO, G. ; BALDUS, G. ; CORDA, D. ; CIPRESSO, P. ; RIVA, G. : A mobile data collection platform for mental health research. In: *Personal and Ubiquitous Computing* 17 (2013), Nr. 2, 241-251. http://dx.doi.org/10.1007/s00779-011-0465-2. – DOI 10.1007/s00779–011–0465–2. – ISSN 1617–4917

[22] GUNDLACH, H. : What is a psychological instrument? In: *Psychology's Territories: Historical and Contemporary Perspectives from Different Disciplines* (2012), 01, S. 195–224. http://dx.doi.org/10.4324/9780203936658. – DOI 10.4324/9780203936658

[23] ISHIDA, R. ; W3C ; MILLER, S. K. ; BOEING: *Lokalisierung vs. Internationalisierung*. https://www.w3.org/International/questions/qa-i18n.de.html, Abruf: 13.02.2020

[24] KEEDLE, H. ; SCHMIED, V. ; BURNS, E. ; DAHLEN, H. : The Design, Development, and Evaluation of a Qualitative Data Collection Application for Pregnant Women. In: *Journal of Nursing Scholarship* 50 (2018), Nr. 1, 47-55. http://dx.doi.org/10.1111/jnu.12344. – DOI 10.1111/jnu.12344

[25] KRAFT, R. ; SCHLEE, W. ; STACH, M. ; REICHERT, M. ; LANGGUTH, B. ; BAUMEISTER, H. ; PROBST, T. ; HANNEMANN, R. ; PRYSS, R. : Combining Mobile Crowdsensing and Ecological Momentary Assessments in the Healthcare Domain. In: *Frontiers in Neuroscience* 14 (2020), February, 164. http://dbis.eprints.uni-ulm.de/1879/

[26] LESLIE, H. : International Developments in OpenEHR Archetypes and Templates. In: *Health Information Management Journal* 37 (2008), Nr. 1, 38-39. http://dx.doi.org/10.1177/183335830803700104. – DOI 10.1177/183335830803700104. – PMID: 18245863

[27] LI, Y. ; MANOHARAN, S. : A performance comparison of SQL and NoSQL databases. In: *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 2013. – ISSN 1555–5798, S. 15–19

[28] MYERS, B. A. ; KO, A. J. ; BURNETT, M. M.: Invited Research Overview: End-User Programming. In: *CHI '06 Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery (CHI EA '06). – ISBN 1595932984, 75–80

[29] NURSEITOV, N. ; PAULSON, M. ; REYNOLDS, R. ; IZURIETA, C. : Comparison of JSON and XML Data Interchange Formats: A Case Study. In: *Caine* 9 (2009), S. 157–162

[30] O'GRADY, S. : *The RedMonk Programming Language Rankings: January 2020*. https://redmonk.com/sogrady/2020/02/28/language-rankings-1-20/?utm_source=rss&utm_medium=rss&utm_campaign=language-rankings-1-20, Abruf: 09.02.2020

[31] PAVLOVIĆ, I. ; KERN, T. ; MIKLAVČIČ, D. : Comparison of paper-based and electronic data collection process in clinical trials: Costs simulation study. In: *Contemporary Clinical Trials* 30 (2009), Nr. 4, 300 - 316. http://dx.doi.org/https://doi.org/10.1016/j.cct.2009.03.008. – DOI https://doi.org/10.1016/j.cct.2009.03.008. – ISSN 1551–7144

[32] PROBST, T. ; PRYSS, R. ; LANGGUTH, B. ; SPILIOPOULOU, M. ; LANDGREBE, M. ; VESALA, M. ; HARRISON, S. ; SCHOBEL, J. ; REICHERT, M. ; STACH, M. ; SCHLEE, W. : Outpatient Tinnitus Clinic, Self-Help Web Platform, or Mobile Application to Recruit Tinnitus Study Samples? In: *Frontiers in Aging Neuroscience* 9 (2017), April, S. 113–113

[33] SCAFFIDI, C. ; SHAW, M. ; MYERS, B. : Estimating the Numbers of End Users and End User Programmers. In: *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, 2005, S. 207–214

[34] SCHICKLER, M. ; SCHOBEL, J. ; PRYSS, R. ; REICHERT, M. : Mobile Crowd Sensing ? A New way of collecting data from trauma samples? In: *XIV Conference of European Society for Traumatic Stress Studies (ESTSS) Conference*, 244

[35] SCHOBEL, J. ; PRYSS, R. ; REICHERT, M. : Using Smart Mobile Devices for Collecting Structured Data in Clinical Trials: Results from a Large-Scale Case Study. In: *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*, 2015. – ISSN 1063–7125, S. 13–18

[36] SCHOBEL, J. ; PRYSS, R. ; SCHICKLER, M. ; RUF-LEUSCHNER, M. ; ELBERT, T. ; REICHERT, M. : End-User Programming of Mobile Services: Empowering Domain Experts to Implement Mobile Data Collection Applications. In: *2016 IEEE International Conference on Mobile Services (MS)*, 2016. – ISSN 2329–6453, S. 1–8

[37] SCHOBEL, J. ; REICHERT, M. (Hrsg.) ; SCHLEE, W. (Hrsg.) ; PRYSS, R. (Hrsg.): *A Model-Driven Framework for Enabling Flexible and Robust Mobile Data Collection Applications*. http://dbis.eprints.uni-ulm.de/1695/. Version: September 2018

[38] SCHOBEL, J. ; PRYSS, R. ; PROBST, T. ; SCHLEE, W. ; SCHICKLER, M. ; REICHERT, M. : Learnability of a Configurator Empowering End Users to Create Mobile Data Collection Instruments: Usability Study. In: *JMIR Mhealth Uhealth* 6 (2018), Jun, Nr. 6, e148. http://dx.doi.org/10.2196/mhealth.9826. – DOI 10.2196/mhealth.9826. – ISSN 2291–5222

[39] SCHOBEL, J. ; PRYSS, R. ; SCHICKLER, M. ; REICHERT, M. : A Configurator Component for End-User Defined Mobile Data Collection Processes. In: DRIRA, K. (Hrsg.) ; WANG, H. (Hrsg.) ; YU, Q. (Hrsg.) ; WANG, Y. (Hrsg.) ; YAN, Y. (Hrsg.) ; CHAROY, F. (Hrsg.) ; MENDLING, J. (Hrsg.) ; MOHAMED, M. (Hrsg.) ; WANG, Z. (Hrsg.) ; BHIRI, S. (Hrsg.): *Service-Oriented Computing – ICSOC 2016 Workshops*. Cham : Springer International Publishing, 2017. – ISBN 978–3–319–68136–8, S. 216–219

[40] SCHOBEL, J. ; PRYSS, R. ; SCHICKLER, M. ; REICHERT, M. : Process-Driven Mobile Data Collection (Extended Abstract). In: *8th International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA 2017)*

[41] SCHOBEL, J. ; PRYSS, R. ; SCHLEE, W. ; PROBST, T. ; GEBHARDT, D. ; SCHICKLER, M. ; REICHERT, M. : Development of Mobile Data Collection Applications by Domain Experts: Experimental Results from a Usability Study. In: DUBOIS, E. (Hrsg.) ; POHL, K. (Hrsg.): *Advanced Information Systems Engineering*. Cham : Springer International Publishing, 2017. – ISBN 978–3–319–59536–8, S. 60–75

[42] SCHOBEL, J. ; PRYSS, R. ; WIPP, W. ; SCHICKLER, M. ; REICHERT, M. : A Mobile Service Engine Enabling Complex Data Collection Applications. In: SHENG, Q. Z. (Hrsg.) ; STROULIA, E. (Hrsg.) ; TATA, S. (Hrsg.) ; BHIRI, S. (Hrsg.): *Service-Oriented Computing*. Cham : Springer International Publishing, 2016. – ISBN 978–3–319–46295–0, S. 626–633

[43] SCHOBEL, J. ; SCHICKLER, M. ; PRYSS, R. ; MAIER, F. ; REICHERT, M. : Towards Process-Driven Mobile Data Collection Applications: Requirements, Challenges, Lessons Learned. In: *10th Int'l Conference on Web Information Systems and Technologies (WEBIST 2014), Special Session on Business Apps*, 371–382

[44] SCHOBEL, J. ; SCHICKLER, M. ; PRYSS, R. ; NIENHAUS, H. ; REICHERT, M. : Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned. In: *9th Int'l Conference on Web Information Systems and Technologies (WEBIST 2013), Special Session on Business Apps*, 509–518

[45] SCHOBEL, J. ; SCHICKLER, M. ; PRYSS, R. ; REICHERT, M. : Process-Driven Data Collection with Smart Mobile Devices. In: MONFORT, V. (Hrsg.) ; KREMPELS, K.-H. (Hrsg.): *Web Information Systems and Technologies*. Cham : Springer International Publishing, 2015. – ISBN 978–3–319–27030–2, S. 347–362

[46] SCHOBEL, J. ; SCHICKLER, M. ; PRYSS, R. ; REICHERT, M. ; ELBERT, T. : A Domain-Specific Framework for Collecting Data in Trials with Smart Mobile Devices. In: *XIV Conference of European Society for Traumatic Stress Studies (ESTSS) Conference*

[47] TAYLOR, K. ; SILVER, L. : Smartphone Ownership Is Growing Rapidly Around the World, but Not Always Equally. In: *Pew Research Center* (2019)

[48] WATERS, K. : Prioritization using moscow. In: *Agile Planning* 12 (2009), S. 31

A

Anhang

```
<component>
  <section>
    <code code="11384-5" codeSystem="2.16.840.1.113883.6.1"
        codeSystemName="LOINC"/>
    <title>Physical Examination</title>
    <component>
      <section>
        <code code="8716-3" codeSystem="2.16.840.1.113883.6.1"
            codeSystemName="LOINC"/>
        <title>Vital Signs</title>
        <text>
          <table>
            <tr>
              <th>Date / Time</th>
              <th>April 7, 2000 14:30</th>
              <th>April 7, 2000 15:30</th>
            </tr>
            <tr>
              <th>Height</th>
              <td>177 cm (69.7 in)</td>
          </table>
        </text>
        <entry>
          <Observation>
            <code code="50373000" codeSystem="2.16.840.1.113883.6.96"
                codeSystemName="SNOMED CT" displayName="Body height
                measure"/>
            <effectiveTime value="200004071430"/>
            <value xsi:type="PQ" value="1.77" unit="m">
              <translation value="69.7" code="[in_I]" codeSystem="
                  2.16.840.1.113883.6.8" codeSystemName="UCUM"/>
            </value>
          </Observation>
        </entry>
      </section>
    </component>
  </section>
</component>
```

Listing A.1: Ausschnitt CDA Patientendaten [8]

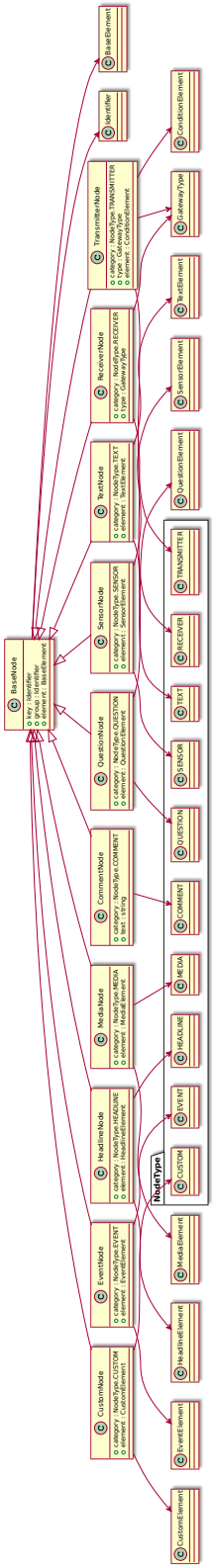

Abbildung A.1: Nodes Pseudo-UML-Klassendiagramm

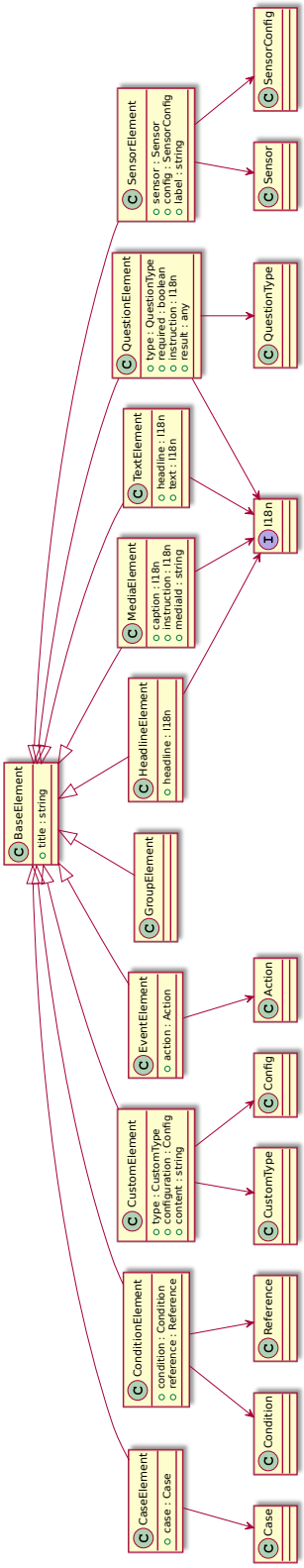

Abbildung A.2: Elements Pseudo-UML-Klassendiagramm

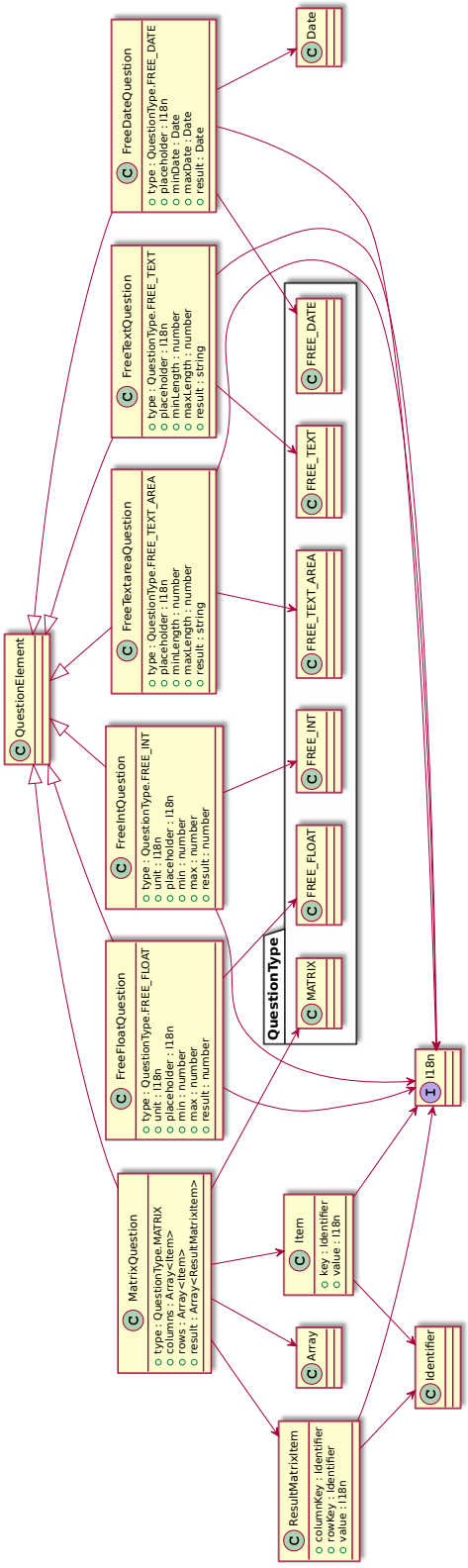

Abbildung A.3: Question-Elements Pseudo-UML-Klassendiagramm

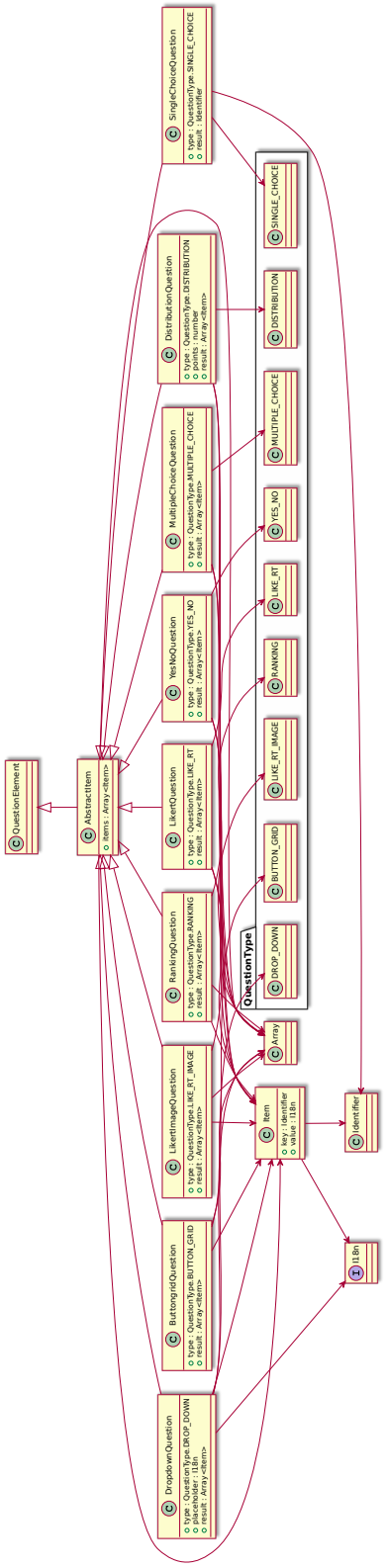

Abbildung A.4: Item-Question-Elements Pseudo-UML-Klassendiagramm

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Maximilian Scheurich Matrikelnummer: 857517

Erklärung

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

Ulm, den 21.05.2020 M. Scheurich.

Maximilian Scheurich