ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

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

Entwicklung einer Webapplikation zur Verwaltung von Ergebnissen digitaler Fragebögen

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Benedikt Fels
benedikt.fels@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Dr. Johannes Schobel

2018

Fassung 3. Dezember 2018

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Fragebögen werden häufig in den Bereichen der Psychologie und dem Gesundheitswesen eingesetzt, um gezielt und effizient große Mengen an personenbezogenen Daten zu erheben. Diese werden bis heute größtenteils papierbasiert verwendet, obwohl dies mehrere Nachteile mit sich bringt. Das QuestionSys Framework, das an der Universität Ulm entwickelt wurde, versucht die Nachteile der Erhebung von Daten durch papierbasierte Fragebögen zu eliminieren, indem der gesamte Prozess digitalisiert wird. Das Framework bietet dem Nutzer die Möglichkeit digitale Fragebögen mit einer Konfigurator Anwendung zu erstellen. Die Fragebögen können im Anschluss daran mithilfe von mobilen Endgeräten ausgefüllt und die entstehenden Ergebnisse auf einen Server übertragen werden. Die erfassten Ergebnisdaten können bisher jedoch nicht zentral verwaltet werden. Dies führt dazu, dass ein Nutzer des QuestionSys Frameworks bislang nicht in der Lage ist, mittels eines Softwaresystems sämtliche ihm zugeordneten Fragebögen und deren Ergebnisse verwalten zu können.

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung einer Webapplikation, welche die Verwaltung der erfassten Daten digitaler Fragebögen innerhalb des QuestionSys Frameworks ermöglicht. Die Webanwendung zeigt hierfür dem Benutzer sämtliche ihm zugewiesenen Fragebögen und deren Versionen an. Zudem werden in der Anwendung alle bisher erhobenen Resultate einer Fragebogenversion aufgelistet. Einzelne Ergebnisdatensätze können durch die Anwendung editiert und gelöscht werden. In der Arbeit wird hierfür anhand einer Anforderungsanalyse das Konzept der Webanwendung vorgestellt und die resultierende Architektur beschrieben. Des Weiteren werden Implementierungsaspekte im Hinblick auf die verwendeten Vorlagen sowie ausgewählter Bereiche der Anwendung erläutert. Abschließend wird die grafische Benutzeroberfläche der entwickelten Anwendung vorgestellt.

Danksagung

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

Besonders möchte ich mich bei meinem Betreuer Dr. Johannes Schobel bedanken, der mich stets mit hilfreichen Anregungen sowie konstruktiver Kritik unterstützt und betreut hat.

Inhaltsverzeichnis

1

Einleitung

Fragebögen werden häufig in Anwendungsbereichen der Psychologie oder dem Gesundheitswesen verwendet, um große Mengen an personenbezogenen Daten zu erheben. Diese Art der Untersuchungsverfahren wird heutzutage meist noch papierbasiert durchgeführt, was in einem deutlich erhöhtem Arbeitsaufwand in der Verarbeitung und Analyse der Daten resultiert [1]. Um die Datenerfassung durch Fragebögen zu digitalisieren, wurde an der Universität Ulm das *QuestionSys Framework* entwickelt [2]. Das Framework ermöglicht es Nutzern ohne ausgeprägte IT Kenntnisse, digitale Fragebögen mithilfe einer benutzerfreundlichen Konfigurator Anwendung zu erstellen. Die erstellten Fragebögen können daraufhin auf unterschiedlichsten mobilen Endgeräten ausgefüllt und die Ergebnisse gespeichert werden. Durch die Bearbeitung der Fragebögen auf den mobilen Endgeräten kann die Fehlerrate verringert und die Qualität der erhobenen Daten verbessert werden [3]. Dies folgt unter anderem daraus, dass Fehlerquellen wie etwa die Digitalisierung der papierbasierten Fragebögen oder das fehlerhafte Ausfüllen eines Fragebogens verhindert werden können. Zusätzlich können die Ergebnisse der beantworteten Fragebögen von den mobilen Endgeräten auf einen Datenbankserver übertragen und dort gespeichert werden. Ein Konzept für die Verwaltung der erfassten Daten ist bisher jedoch noch nicht vorhanden. Der Nutzer besitzt dadurch nicht die Möglichkeit, die erfassten Ergebnisse der Fragebögen zu verwalten. Um eine statistisch aussagekräftige Analyse der erhobenen Ergebnisse durchführen zu können, werden sämtliche Resultate der ausgefüllten Fragebögen benötigt. Derzeit existiert noch keine Softwarelösung, die es dem Nutzer des QuestionSys Frameworks erlaubt, die gesammelten Daten von dem Server abzurufen und zu verwalten, obwohl diese bereits online abgespeichert werden können.

Aus diesem Grund soll in der vorliegen Arbeit eine Webanwendung entwickelt werden, die dem Nutzer gestattet, sämtliche durch das QuestionSys Framework erhobenen Ergebnisse der ihm zugeordneten Fragebögen verwalten zu können. Die Webapplikation muss dabei mit dem Server des QuestionSys Frameworks kommunizieren, um die benötigten Daten abzurufen. Dem Nutzer soll eine benutzerfreundliche Oberfläche zur Verfügung gestellt werden, um mit den gesammelten Daten interagieren zu können. Insbesondere sollen die Fragebögen und deren existierende Versionen mithilfe von informativen Dashboard Übersichten dargestellt werden. Alle erfassten Resultate einer ausgewählten Fragebogenversion sollen aufgelistet und bestimmte Metadaten einzelner Ergebnisse durch das System editiert werden können.

Struktur der Arbeit

Die Arbeit gliedert sich in folgende Kapitel. Zunächst wird in Kapitel 2 das zugrunde liegende Umfeld beschrieben. In Abschnitt 2.1 wird das Konzept des QuestionSys Projekts erläutert. Dabei wird auch festgelegt, an welcher Stelle innerhalb des Projekts die entwickelte Webanwendung zum Einsatz kommt. Danach wird in Abschnitt 2.2 das JavaScript Framework Angular vorgestellt, welches für die Entwicklung der Webapplikation verwendet wurde. Abschnitt 2.3 beinhaltet eine Übersicht über die JSON:API Spezifikation, welche für den Datenaustausch zwischen dem Server des QuestionSys Frameworks und der Webanwendung verwendet wird. In Kapitel 3 wird das Konzept vorgestellt. Zunächst wird hierfür in Abschnitt 3.1 eine Anforderungsanalyse durchgeführt. Diese besteht aus einem Anwendungsszenario und den funktionalen, technischen sowie nicht funktionalen Anforderungen an die Anwendung. Anhand dessen wird in Abschnitt 3.2 das Konzept der Webapplikation dargelegt. Ferner wird in Abschnitt 3.3 die daraus resultierende Architektur der Webanwendung beschrieben. Das Kapitel 4 beschäftigt sich mit der Implementierung der Anwendung. Zunächst werden in Abschnitt 4.1 die verwendeten Vorlagen, INSPINIA und Angular Material, vorgestellt. Im Anschluss daran werden ausgewählte Bereiche der Implementierung in Abschnitt 4.2 beschrieben. Kapitel 5 beinhaltet eine Übersicht über die erstellte Anwendung, indem die einzelnen Komponenten der Anwendung durch Screenshots vorgestellt und beschrieben werden.

Abschließend sollen in Kapitel 6 verwandte Systeme vorgestellt und auf die Ähnlichkeiten bezüglich der grafischen Benutzeroberfläche zwischen der entwickelten Anwendung und Webanwendung von Dropbox eingegangen werden.

2

Grundlagen

Dieses Kapitel gibt zunächst einen Überblick über das Konzept des QuestionSys Frameworks und beschreibt an welcher Stelle das zu entwickelnde System hierbei eingesetzt werden kann. Im Anschluss daran werden Grundlagen des Angular Frameworks beschrieben, welches zur Erstellung der Webanwendung genutzt wird. Abschließend wird die JSON:API Spezifikation, welche bei der Datenübertragung zwischen dem Server und Anwendung eingesetzt wird, vorgestellt.

2.1 QuestionSys Framework

Das QuestionSys Framework ermöglicht es, die Datenerhebung mittels papierbasierter Fragebögen zu digitalisieren, sodass diese auf mobilen Endgeräten ausgefüllt werden können. Dabei sollen vor allem Experten IT fremder Bereiche (beispielsweise der Psychologie oder dem Gesundheitswesen) die Möglichkeit bekommen, ihre eigenen mobilen Anwendungen erstellen können. Diese enthalten die digitalen Fragebögen und das Erstellen der Anwendungen soll ohne Kenntnisse in der Softwareentwicklung möglich sein. Abbildung 2.1 beschreibt den Ansatz der Erstellung eines Fragebogens mit der Konfigurator Anwendung **1)**, das daraus resultierende Prozess Modell des Fragebogens **2)** sowie die Anwendung zur prozessgesteuerten mobilen Datenerhebung **3)**.

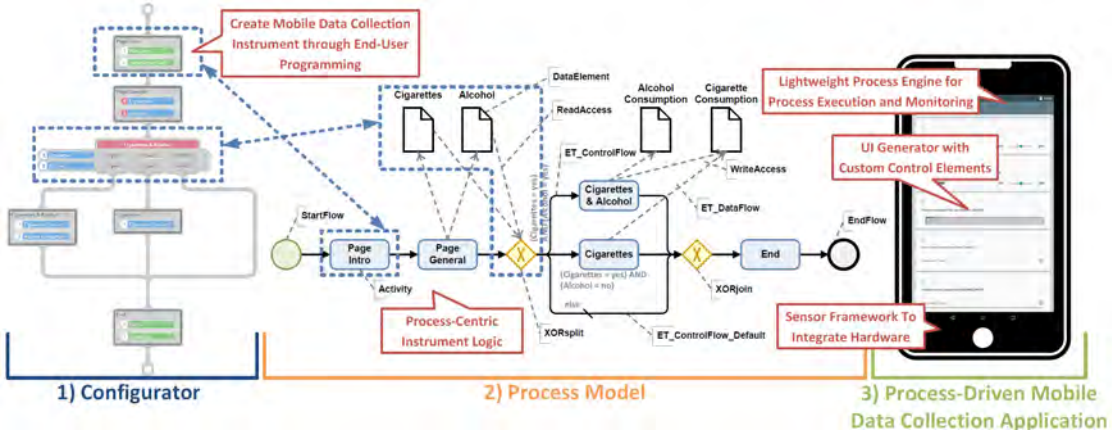

Abbildung 2.1: Prozessbasierte Entwicklung von Anwendungen zur mobilen Datenerhebung [2]

Die Konfigurator Anwendung zur Fragebogenerstellung wird in [4] detailliert beschrieben. Der Konfigurator bietet den Domänen-Experten eine abstrakte, nachvollziehbare grafische Oberfläche zur Modellierung der Ablauflogik des digitalen Fragebogens. Die Logik und der Inhalt der Fragebögen werden dabei prozessorientiert modelliert. Das entstehende Prozess Modell kann daraufhin auf ein mobiles Endgerät übertragen und mithilfe eine Prozess-Engine gerätespezifisch interpretiert werden [5]. Die Benutzeroberfläche der mobilen Anwendungen wird anhand der Modellierung generiert und der Benutzer ist in der Lage den Fragebogen auszufüllen. Die dabei entstehenden Ergebnisse können an einen Server übertragen und dort gesammelt werden.

Die Architektur des QuestionSys Frameworks ist in Abbildung 2.2 abgebildet. Hierbei beschreibt die linke Spalte die Konfigurator Komponente, die den Nutzern (beispielsweise ein Studienleiter oder ein Analyst) die prozessbasierte Erstellung der Fragebögen (a) sowie die Definition von Regeln für die automatisierte Auswertung (b) der Fragebögen ermöglicht. Die mittlere Spalte repräsentiert die Server Komponente des QuestionSys Frameworks, die für sämtlichen Datenaustausch (1) - (5) zwischen den Komponenten zuständig ist. Die rechte Spalte skizziert die Datenbeschaffung durch den Gebrauch von mobilen Endgeräten [6].

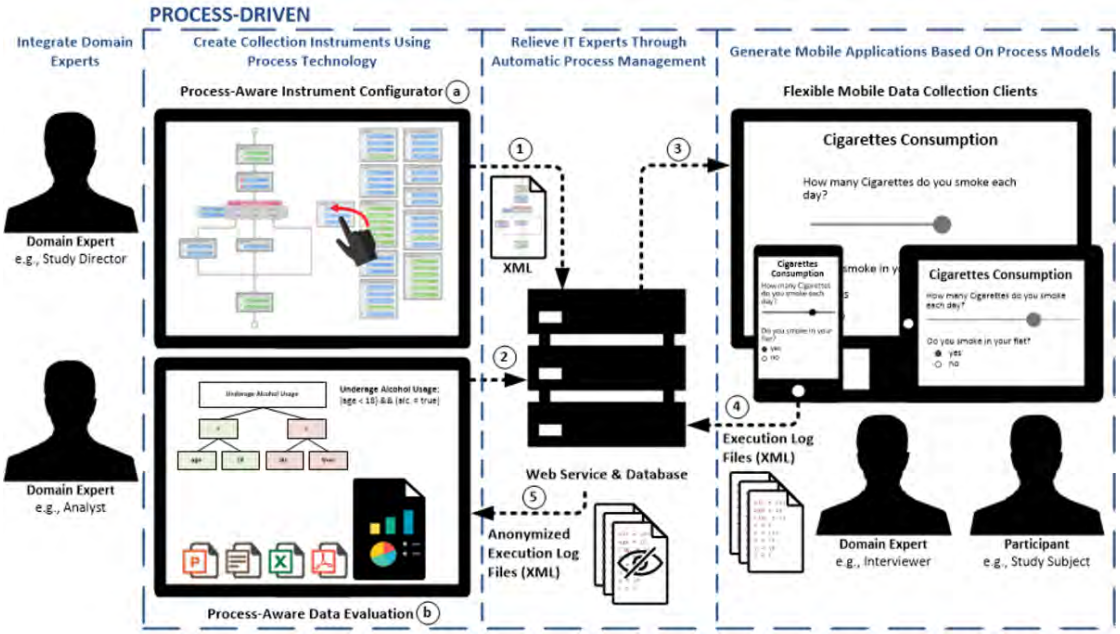

Abbildung 2.2: Architektur des QuestionSys Frameworks [7]

Bei der Durchführung einer Studie wird dem zuständigen QuestionSys Framework Nutzer der gewünschte Fragebogen zugewiesen. Die Zuordnung der gesammelten Ergebnisse ist dadurch im Anschluss an die Studie möglich. Fragebögen können zudem in unterschiedlichen Versionen verfügbar sein. Beispielsweise existieren individuelle Versionen für verschiedene Altersgruppen. Die mit den mobilen Endgeräten erhobenen Daten werden während der Durchführung der Studie an den QuestionSys Server übertragen und dort gespeichert. Das zu entwickelnde System erweitert demnach das QuestionSys Framework um eine Komponente, die für die Verwaltung der erhobenen Ergebnisse zuständig ist. Abbildung 2.3 stellt schematisch die Interaktionen der jeweiligen Komponente mit dem Server des QuestionSys Frameworks dar.

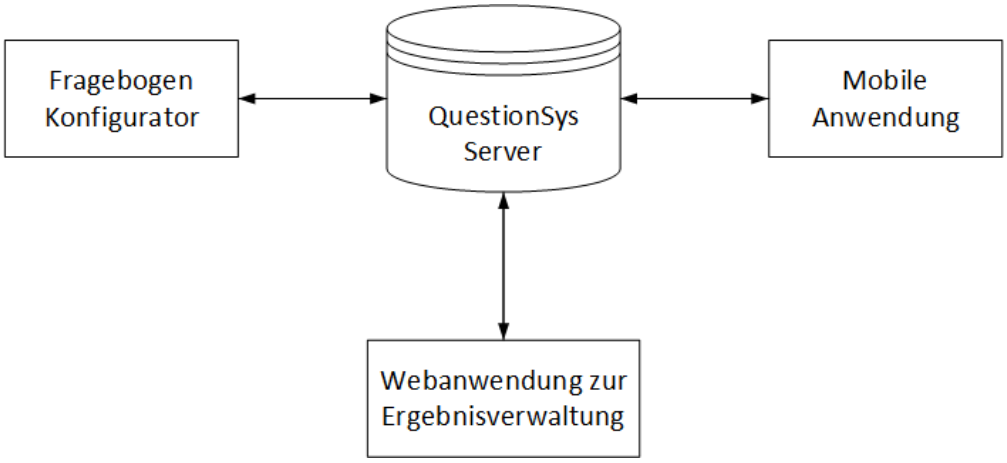

Abbildung 2.3: Schematisch dargestellte Interaktionen mit dem QuestionSys Server

2.2 Angular

JavaScript ist die am weitesten verbreitete Programmiersprache [8] und wird in fast allen aktuellen Webseiten verwendet. Für die Entwicklung einer Webanwendung mit JavaScript gibt es eine große Anzahl an Front-end Frameworks.
Nach Derek Schlesselman [9] zählen zu den wichtigsten Vorteilen von Front-end Frameworks, dass der Entwickler schnell und einfach loslegen kann und die Benutzeroberfläche bereits ansprechend aussieht. Der enthaltene Code ist zudem verlässlich sowie gut getestet und der Entwickler findet jederzeit, aufgrund der weiten Verbreitung der Frameworks, Hilfe bei auftretenden Problemen. Für die Entwicklung der Webanwendung zur Verwaltung der erhobenen Ergebnisse wird das Framework Angular verwendet. Angular beschreibt dabei eine Plattform und ein Framework zur Erstellung von Clientseitigen Webanwendungen in HTML (Hypertext Markup Languague) und TypeScript [10]. TypeScript ist eine Erweiterung von JavaScript, die von Mark Clow so zusammengefasst wurde [11]:

> TypeScript = JavaScript + Types + Classes + Modules + More

TypeScript kann bisher nicht von Webbrowsern direkt verarbeitet werden und wird zu reinem JavaScript kompiliert [11].

Die wichtigsten Bestandteile einer Angular Anwendung sollen nun nach [10] zusammengefasst werden.

Module

Angular Module deklarierten einen Kompilierungskontext für zusammenhängende Angular Komponenten einer Webanwendung. Ein Modul verknüpft dabei die enthaltenen Komponenten mit zugehörigem Code, um funktionsfähige Einheiten zu bilden. Jede Angular Anwendung enthält dabei mindestens ein Root-Modul, welches das Starten der Webapplikation ermöglicht. Weitere Module können für einzelne Bestandteile der Webanwendung erstellt werden. Durch die Aufteilung in separate Module wird die Übersichtlichkeit der Anwendung erhöht. Zusätzlich werden die einzelnen Module ausschließlich bei Bedarf geladen, wodurch die Anwendung schneller startet. Angular Module können dabei selbst Module importieren oder ihr Funktionalitäten exportieren, um diese anderen Modulen bereitzustellen.

Komponenten

Angular Komponenten definieren und steuern eine Ansicht innerhalb der Webanwendung. In jeder Angular Anwendung ist eine Root-Komponente vorhanden. Diese verbindet die hierarchisch angeordneten Komponenten der Webanwendung mit dem DOM (Dokument Objekt Model) der Webseite. Jede Komponente definiert eine Klasse, die sowohl Anwendungsdaten als auch Programmlogik beinhaltet und dabei mit einer HTML Datei verknüpft ist. Die HTML Datei beschreibt dabei wie Angular die Webseite darstellen soll.

Templates, Directives und Datenbindung

Ein Angular Template beschreibt eine HTML Datei mit angularspezifischen Ausdrücken. Diese werden vor dem Darstellen der Webseite ausgewertet und können den HTML Code verändern. Die Ausdrücke unterteilen sich in sogenannte *Template Directives*, die programmlogische Ausdrücke enthalten und in Ausdrücke, die für die Datenbindung

zwischen dem DOM und den Anwendungsdaten der zugeordneten Angular Komponente zuständig sind. Angular interpretiert vor dem Anzeigen der Webseite sämtliche angular-spezifischen Bezeichnungen und modifiziert anhand derer die HTML Struktur und das Dokument Objekt Modell der Webseite. Ein Ausschnitt eines solchen Angular Templates ist in Listing 4.1 abgebildet.

Services und Dependency Injection

Service Klassen können in Angular dafür verwendet werden, um Daten zwischen mehreren Komponenten auszutauschen. Die Klasse stellt dabei einen wiederverwendbaren Datenzugriffspunkt dar [12]. Services können innerhalb von Angular Komponenten verwendet werden, indem sie durch die sogenannte Dependency Injection (Einbringen von Abhängigkeiten) eingebunden werden. Die Services können dabei verwendet werden, um Daten von einem Server abzufragen und diese den Komponenten zur Verfügung zu stellen. Durch die Auslagerung solcher Tätigkeiten wird die Modularität der Anwendung sowie die Wiederverwertbarkeit der Bauteile erhöht. Die Verwendung von Services in der Webapplikation wird in Abschnitt 4.2 beschrieben.

Routing

Angular stellt dem Entwickler ein vordefiniertes Modul zur Verfügung, das zur Navigation zwischen den einzelnen Komponenten genutzt werden kann. Dieses Router Modul beinhaltet einen Dienst, der die Zuordnung von bestimmten Navigationspfaden zu deren zugehören Ansichten der Applikation ermöglicht. Die allgemeinen Navigationskonventionen eines Browsers bleiben dabei erhalten. Diese beinhalten, dass der Nutzer durch die Eingabe einer URL (Uniform Resource Locator) in die Adresszeile des Browsers, durch Interaktionen mit der grafischen Benutzeroberfläche der Webseite sowie durch die „Zurück“ und „Vorwärts“ Schaltflächen des Browsers navigieren kann. Dabei werden URL basierte Pfade nicht wie üblich Seiten, sondern bestimmte Ansichten der Anwendung zugewiesen. Bei Benutzereingaben des Nutzers, die eine neue Seite in dem Webbrowser laden würden, unterbindet der Router das Verhalten des Browsers und weist aufgrund

des angefragten Pfads die zugehörige Angular Ansicht zu. Zudem ist der Router auch dafür zuständig, benötigte Module einer Ansicht während der Verwendung der Webapplikation zu laden. Die Regeln zur Navigation in der Anwendung beinhalten eine Verknüpfung der Navigationspfade mit den korrespondierenden Angular Komponenten. Die Verwendung des Routers in der Webanwendung wird in Abschnitt 4.2 beschrieben.

Abbildung 2.4 beschreibt die Zusammenhänge der zuvor beschriebenen Grundbaustücke einer Angular Anwendung. Die Service Klassen der Anwendung können per Dependency Injection in die Angular Komponenten eingefügt werden. Ein Template ist bidirektional mit einer Komponente verknüpft. Angular Directives können Templates verändern und Module deklarieren einen Kompilierungskontext für zusammengehörige Komponenten.

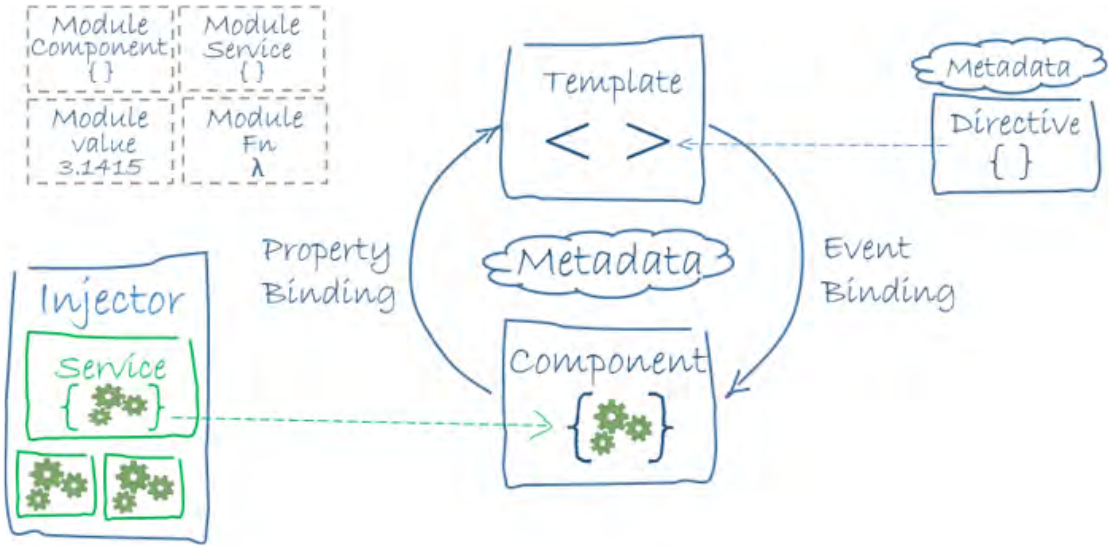

Abbildung 2.4: Zusammenhänge der Grundbaustücke von Angular [10]

2.3 JSON:API

Für den Datenaustausch innerhalb des QuestionSys Framework zwischen dem Server und den Anwendungen wird das JSON (JavaScript Object Notation) Format verwendet. JSON beschreibt dabei ein kompaktes Format für den Datenaustausch. Es kann vom

Mensch leicht gelesen und geschrieben werden und Maschinen können es problemlos interpretieren und erzeugen. Da JSON ein textbasiertes Format ist, kann es plattformunabhängig für den Datenaustausch eingesetzt werden [13]. Abbildung 2.5 beschreibt die verfügbaren Datenstrukturen des JSON Formats.

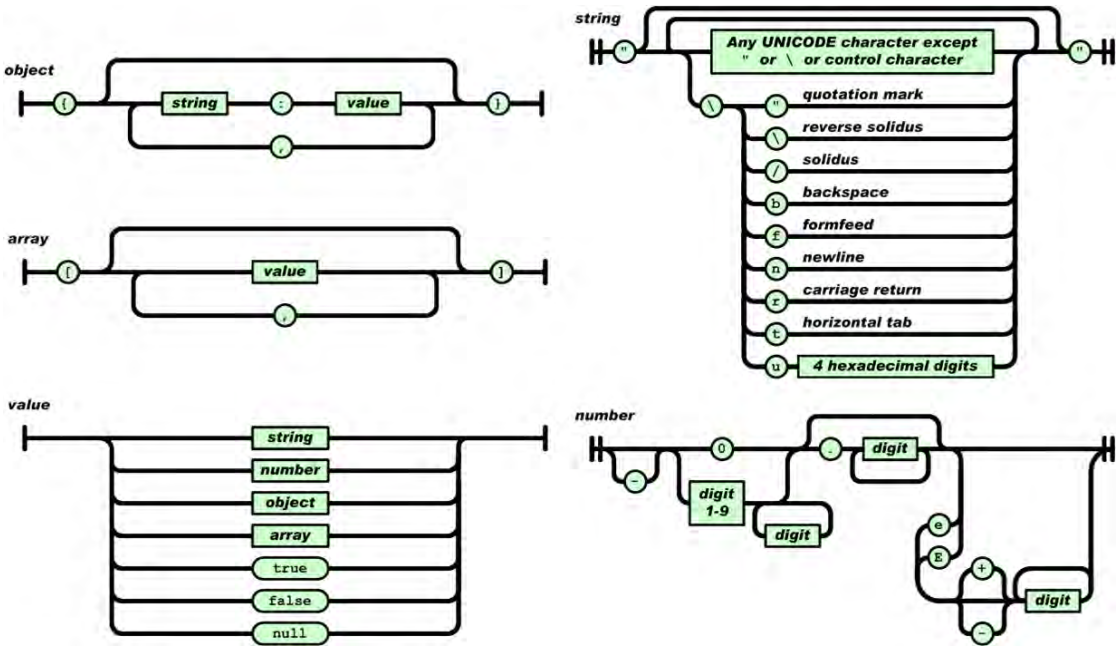

Abbildung 2.5: JSON Datenstrukturen [13]

Der Datenaustausch zwischen dem Server und den Anwendungen des QuestionSys Frameworks erfolgt dabei nach der JSON:API Spezifikation [14]. Diese definiert, wie die JSON-formatierten Daten bei Anfragen an den Server und dessen Antworten formatiert sein sollen. Die Spezifikation sorgt dafür, dass Anfragen an die API (Application Programming Interface) effizienter gestaltet werden können. Dadurch müssen insgesamt weniger Anfragen getätigt werden [15]. Die JSON:API Spezifikation soll im Folgenden anhand von [14, 15] genauer betrachtet werden.

Struktur des Dokuments

Bei der JSON:API Spezifikation müssen sämtliche Dokumente als valide JSON Objekte formatiert sein. Abbildung 2.6 zeigt hierbei beispielhaft, wie ein Artikel innerhalb eines JSON:API konformen Dokuments aussehen würde. Dabei enthält eine Ressource (hier ein Artikel) immer einen *type* sowie eine *id*. Die *attributes* und *relationsships* sind optional. In den *attributes* werden die Attribute der Ressource aufgelistet und die *relationships* beschreiben bestehende Verbindungen zu anderen Ressourcen.

```
// ...
{
  "type": "articles",
  "id": "1",
  "attributes": {
    "title": "Rails is Omakase"
  },
  "relationships": {
    "author": {
      "links": {
        "self": "/articles/1/relationships/author",
        "related": "/articles/1/author"
      },
      "data": { "type": "people", "id": "9" }
    }
  }
}
// ...
```

Abbildung 2.6: Repräsentation eines Artikels im JSON:API konformen Format [16]

Zusammengehörige Dokumente

Die JSON:API Spezifikation kann die Möglichkeit bieten, Dokumente zurückzugeben, welche bereits mit einer Anfrage alle, der Ressource in Form von Beziehungen zugeordneten, Objekte einschließt. Bei dem in Abbildung 2.6 abgebildeten Dokument würde dabei aufgrund der *relationships* die *people* Ressource des Autors innerhalb eines *included* Arrays angehängt werden.

Einschließen verknüpfter Ressourcen

Eine JSON:API konforme API kann zudem die Möglichkeit bieten, ausgewählte Verknüpfungen einer Ressource innerhalb eines Dokuments zurückzugeben. Dies hat gegenüber des zuvor beschriebenen Ansatzes den Vorteil, dass ausschließlich die selektierten Ressourcen vom Server übertragen werden. Dadurch kann die Übertragung ungenutzter Informationen unterbunden werden. Um dem Dokument bestimmte Ressourcen anzuhängen, muss die Anfrage an den Server um einen Parameter erweitert werden. Es können mehrere verknüpfte Ressourcen angefragt werden, indem man die Parameter, wie in Abbildung 2.7 abgebildet, durch ein Komma trennt. Die dargestellte Anfrage würde somit den Artikel mit „*id=1*“ zurückgeben und dabei sowohl den Autor des Artikels als auch die Autoren der Kommentare an das Dokument anhängen.

```
GET /articles/1?include=author,comments.author HTTP/1.1
Accept: application/vnd.api+json
```

Abbildung 2.7: Anfrage mit verknüpften Ressourcen [16]

Sortierung

Ein weiteres Feature, das von der JSON:API Spezifikation unterstützt werden kann, ist die Sortierung der zurückgegeben Daten. Hierfür wird der Anfrage analog zu dem *Einschließen verknüpfter Ressourcen* ein Parameter übergeben. Abbildung 2.8 zeigt eine beispielhafte Anfrage, die alle Personen vom Server abfragt. Diese werden aufgrund der Parameterübergabe aufsteigend nach dem Alter sortiert zurückgegeben.

```
GET /people?sort=age HTTP/1.1
Accept: application/vnd.api+json
```

Abbildung 2.8: Anfrage mit aufsteigender Sortierung [16]

3

Konzept

In diesem Kapitel wird zunächst eine Anforderungsanalyse durchgeführt, um die geforderte Funktionalität der Applikation sowie die Qualitätsansprüche an das System zu definieren. Daraufhin wird die architektonische Struktur des Softwaresystems abgebildet und die einzelnen Komponenten vorgestellt.

3.1 Anforderungsanalyse

Im folgenden Abschnitt wird zunächst ein Szenario beschrieben, welches einen möglichen Ablauf einer Studiendurchführung zusammen mit dem Zugriff auf die gesammelten Ergebnisse beschreibt. Hierbei werden unter anderem die Aufgaben genannt, welche die zu entwickelnde Applikation bewältigen können soll. Anhand dieses Szenarios werden im Anschluss die funktionalen, die technischen sowie die nicht funktionalen Anforderungen an das Systems spezifiziert.

3.1.1 Anwendungsszenario

Das folgende Szenario soll beschreiben, wie eine Studie mithilfe des QuestionSys Frameworks durchgeführt werden kann und insbesondere wie ein Studienleiter vorgehen würde, um die erhobenen Daten der Studie zu verwalten.
An einer Universität soll das Konsumverhalten der Studenten analysiert werden. Dazu wird im Vorfeld ein digitaler Fragebogen mit der Konfigurator Anwendung des QustionSys Frameworks erstellt. Dieser Fragebogen wird dem Studienleiter im Anschluss daran zugewiesen. Ein Fragebogen kann in mehreren Versionen vorliegen. Der Studienleiter

erhält durch die Zuordnung eines Fragebogens automatisch alle verfügbaren Versionen. Nachdem die teilnehmenden Studenten den Fragebogen mittels der mobilen Anwendungen ausgefüllt haben, werden die erhobenen Daten an einen Server übermittelt und dort gespeichert. Daraufhin meldet sich der Studienleiter mit seinem bereits bestehenden QuestionSys Account an dem zu entwickelnden System an. Nach der Anmeldung werden dem Nutzer zunächst alle ihm zugewiesenen Fragebögen angezeigt. Diese Anzeige beinhaltet auch den Fragebogen zu der aktuellen Studie zum Konsumverhalten der Studenten. Nach der Auswahl des entsprechenden Fragebogens werden dem Benutzer des Systems alle verfügbaren Versionen des Fragebogens aufgeführt, woraus er die benötigte Version auswählen muss. Schließlich werden alle erhobenen Ergebnisdatensätze der ausgewählten Fragebogenversion angezeigt. Der Studienleiter hat nun die Möglichkeit einzelne Datensätze auszuwählen und bei diesen diesen die Beschreibung, die Flags sowie die Farben anzupassen oder den Datensatz zu löschen. Der Studienleiter kann zudem jederzeit seine Profildaten anzeigen lassen.

3.1.2 Anforderungen

Anhand des zuvor beschrieben Szenarios 3.1.1 und allgemeingültiger Anforderungen an Softwaresysteme können im Folgenden die Anforderungen an das System beschrieben werden. Hierzu werden zunächst die funktionalen Anforderungen aufgelistet. Diese beschreiben konkrete Aufgaben, die das System bewerkstelligen können soll. Des Weiteren sollen technische Anforderungen an das System aufgeführt werden, welche die benötigten Interaktionen des Systems mit dem Server beschreiben. Die nicht-funktionalen Anforderungen, welche die Qualitätsanforderungen an das System definieren, welchen im Anschluss daran dargelegt.

Funktionale Anforderungen

FA1 Profil: Die Anwendung muss die persönlichen Profildaten des Benutzers anzeigen können.

FA2 Navigation: Der Nutzer der Webanwendung muss zwischen den einzelnen Ansichten navigieren können.

FA3 Purchase Dashboard: Eine Dashboard Übersicht mit sämtlichen dem Benutzer zugewiesenen Fragebögen muss angezeigt werden können.

FA4 Version Dashboard: Ein Dashboard mit allen verfügbaren Versionen eines Fragebogens muss angezeigt werden können.

FA5 Name der Ressource: Der Name eines Fragebogens oder einer Version muss dargestellt werden können.

FA6 Gesamtanzahl der Ergebnisse: Die Gesamtanzahl der gesammelten Ergebnisse eines Fragebogens beziehungsweise der Version eines Fragebogens muss angezeigt werden können.

FA7 Anzahl an Ergebnissen der letzten 30 Tage: Die Anzahl der innerhalb der letzten 30 Tage gesammelten Ergebnisse eines Fragebogens sowie einer Version muss angezeigt werden können.

FA8 Diagramm: Ein Diagramm der in den letzten 30 Tagen erhobenen Ergebnisse eines Fragebogens sowie der Anzahl der Ergebnisse einer Version muss angezeigt werden können.

FA9 Widget: Die in **FA5-FA8** beschriebenen Informationen sollen in einem Widget zusammengefasst werden. Dieses soll in den Dashboard Übersichten **FA3,FA4** angezeigt werden.

FA10 Ergebnis Tabelle: Eine Tabelle mit den gesammelten Ergebnissen einer Fragebogenversion muss angezeigt werden können.

FA11 Ergebnisse pro Seite: Der Nutzer soll in der Lage sein, die Anzahl der angezeigten Ergebnisse pro Seite zu ändern.

FA12 Navigation innerhalb der Tabelle: Der Benutzer soll in der Lage sein, innerhalb der Tabelle durch die Seiten der gesammelten Ergebnisse zu navigieren.

FA13 Auswahl mehrerer Ergebnisse: Der Nutzer soll mehrere Ergebnisse gleichzeitig auswählen können.

FA14 Sortieren der Ergebnisse: Der Benutzer soll die Reihenfolge der angezeigten Ergebnisse ändern können.

FA15 Bearbeiten der Ergebnisse: Der Nutzer muss die Möglichkeit besitzen, einzelne Ergebnisse mithilfe einer Benutzeroberfläche zu bearbeiten. Hierbei sollen die *Beschreibung*, die *Flags* sowie die fragebogenspezifischen *Farben* des Datensatzes verändert werden können.

FA16 Löschen von Ergebnissen: Gesammelte Ergebnisse sollen durch den Benutzer gelöscht werden können. Hierfür muss eine geeignete Benutzeroberfläche generiert werden.

Technische Anforderungen

TA1 Fehlerfreie Anfragen: Das System muss fehlerfreie, JSON:API konforme Anfragen an die QuestionSys API stellen können.

TA2 Zuweisung der Modelle: Das System muss die JSON-formatierten Daten des Servers korrekt auf Datenmodelle in der Anwendung abbilden können.

TA3 Login: Das System muss den Benutzer erfolgreich am Server des QuestionSys Frameworks einloggen können.

TA4 Logout: Das System muss den Nutzer vollständig vom Server des QuestionSys Frameworks abmelden können.

TA5 Profildaten: Das System muss die Profildaten des eingeloggten Nutzers vom Server abrufen können.

TA6 Zugeordnete Fragebögen: Das System muss sämtliche dem Nutzer zugeordneten Fragebögen vom Server abrufen können.

TA7 Verfügbare Versionen: Das System muss alle verfügbaren Versionen eines Fragebogens vom Server abrufen können.

TA8 Erhobene Ergebnisse der letzten 30 Tagen: Das System muss die Anzahl der erfassten Ergebnisse der letzten 30 Tage von dem Server abrufen können. Diese Information muss sowohl für die Fragebögen als auch für deren Versionen abgerufen werden können.

TA9 Gesammelte Ergebnisse: Das System muss alle erhobenen Resultate einer bestimmten Fragebogenversion vom Server abrufen können.

TA10 Ändern der Beschreibung: Das System muss die *Beschreibung* eines erhobenen Ergebnisdatensatzes ändern können. Dafür muss eine Anfrage an den Server gestellt werden können.

TA11 Ändern der Flags: Das System muss die *Flags* eines Ergebnisdatensatzes mithilfe einer Serveranfrage ändern können.

TA12 Ändern der Colors: Das System muss in der Lage sein, die *Colors* eines Ergebnisdatensatzes mittels einer Anfrage an den Server zu ändern.

TA13 Löschen eines Ergebnisses: Das System muss eine Anfrage an den Server stellen können, um ein gespeichertes Ergebnis zu löschen.

Nicht-funktionale Anforderungen

NFA1 Benutzbarkeit: Die Webanwendung soll ohne Vorkenntnisse leicht und intuitiv zu bedienen sein.

NFA2 Zuverlässigkeit: Die Webapplikation soll stets zuverlässig und ohne Aussetzer arbeiten.

NFA3 Verfügbarkeit: Das System sollte jederzeit zur Verfügung stehen.

NFA4 Dokumentation: Der Programmcode der Webapplikation sollte stets nachvollziehbar dokumentiert sein.

NFA5 Portierbarkeit: Die Webanwendung sollte plattformunabhängig mit jedem aktuellen Browser fehlerfrei funktionieren.

NFA6 Sprache: Die Webapplikation muss in englischer Sprache verfügbar sein.

NFA7 Effizienz: Die Webanwendung sollte so aufgebaut sein, dass der Benutzer mit möglichst wenig Interaktionen sämtliche Funktionen ausführen kann.

NFA8 Erweiterbarkeit: Die Anwendung sollte ohne viel Aufwand um weitere Funktionen erweiterbar sein.

3.2 Konzept der Webanwendung

Mithilfe der zuvor durchgeführten Anforderungsanalyse in Abschnitt 3.1 lässt sich nun ein Konzept erstellen, um alle benötigten Funktionen sowie die Qualitätsansprüche an das Systems umzusetzen. Die Konfigurator Komponente des QuestionSys Frameworks sowie die mobile Applikation zur Beantwortung der Fragebögen wurden bereits entwickelt (vgl. [4]). Die Möglichkeit, einen Account für das QuestionSys System zu erstellen, ist ebenfalls gegeben und nicht Teil der zu entwickelnden Webapplikation. Es ist zwingend notwendig, die personenspezifischen Informationen nur autorisierten Benutzern zur Verfügung zu stellen. Dies hat zur Folge, dass für die Nutzung der Applikation eine Authentifizierung durch einen Login vorausgesetzt wird. Der Benutzer soll seine persönlichen Profildaten anzeigen zu können. Eine schlichte Anzeige soll hierfür generiert werden. Um dem Nutzer der Webanwendung eine Übersicht seiner Fragebögen anzuzeigen, soll hierfür eine Dashboard Übersicht erstellt werden. Abbildung 3.1 zeigt den Entwurf eines solchen Dashboards mit der Darstellung der Fragebögen in Form von Widgets. Diese enthalten den Namen des Fragebogens sowie nützliche Informationen zu der Datenerhebung. Die Anzahl der insgesamt erhobenen Datensätze sowie die Anzahl der innerhalb der letzten 30 Tagen gesammelten Ergebnisse werden hierbei abgebildet. In dem unteren Bereich eines Widgets wird zusätzlich ein Diagramm angezeigt, welches die Anzahl der in den letzten 30 Tagen erhobenen Ergebnisse zudem grafisch darstellt. Das Anklicken eines Widgets sorgt dafür, dass der Benutzer auf die Übersicht der verfügbaren Versionen des ausgewählten Fragebogens gelangt.

Fragebögen können in mehreren Versionen vorliegen. Diese können sich untereinander stark unterscheiden, weshalb die Ergebnisse einzelner Versionen getrennt betrachtet

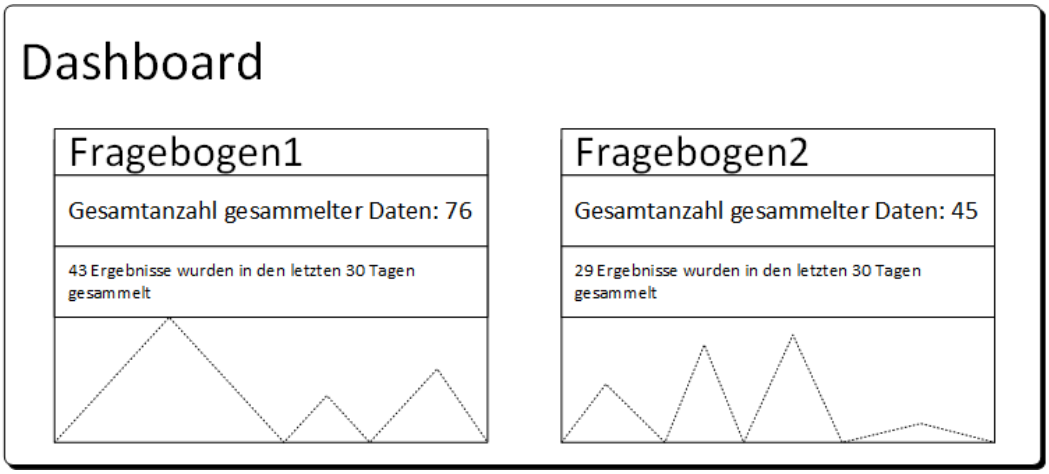

Abbildung 3.1: Entwurf einer Dashboard Übersicht mit Widgets

werden müssen. Um die verfügbaren Versionen eines Fragebogens anzuzeigen, wird analog zu dem zuvor beschriebenen Dashboard eine weitere Übersicht generiert. Diese enthält dabei für jede Version ein Widget, welches den Namen der Version, die Gesamtanzahl der Ergebnisse sowie die Anzahl der in den letzten 30 Tagen erhobenen Daten darstellt. Letztere werden wiederum grafisch mit einem Diagramm in dem unteren Bereich des Widgets angezeigt. Mit einem Klick auf das Widget einer Fragebogenversion gelangt der Nutzer auf die Übersicht der gesammelten Ergebnisse dieser Version. Alle erhobenen Ergebnisse einer Fragebogenversion werden in einer Tabelle aufgelistet. Ein Entwurf der Tabelle ist in Abbildung 3.2 abgebildet. Diese beinhaltet mehrere Informationen der Ergebnisse. Angezeigt werden hierbei *Instance*, *Identifier*, *Description*, *Locale*, *Created*, *Collected*, *Updated*, *Flags* und *Colors*. Die zwei Spalten, *Edit* und *Delete*, sollen für die Benutzerinteraktionen genutzt werden.

Ergebnis Tabelle

Instance	Identifier	Description	Locale	Created	Collected	Updated	Flags	Colors	Edit	Delete
1	1	Desc1	de	Okt	Okt	Nov	123			
2	2	Desc2	en	Aug	Aug	Okt	321			
3	3	Desc3	de	Jun	Jul	Sep	132			

Abbildung 3.2: Entwurf der Ergebnis Tabelle

Der Nutzer der Webanwendung muss in der Lage sein, die *Description*, die *Flags* sowie die *Colors* eines einzelnen Fragebogens zu ändern. Ein Entwurf der Oberfläche für das Editieren eines Ergebnisdatensatzes ist in Abbildung 3.3 dargestellt. Der Benutzer gelangt zu dieser Ansicht, indem er in der Ergebnis Tabelle die „Bearbeiten“ Schaltfläche der *Edit* Spalte betätigt. Die grafische Benutzeroberfläche beinhaltet dabei jeweils ein Textfeld für die *Beschreibung* sowie die *Flags* eines Ergebnisses. Die gesammelten Resultate können bis zu fünf verschiedene *Colors* besitzen. Diese sollen in der Oberfläche dargestellt und durch eine Farbauswahl Oberfläche gesetzt werden können. Die Farben des ausgewählten Ergebnisses können durch Betätigen der „Zurücksetzen“ Schaltfläche zurückgesetzt werden. Die „Speichern“ Taste kann dazu verwendet werden Änderungen zu speichern. Ähnlich zu der Oberfläche für das Bearbeiten eines Ergebnisses soll durch die „Löschen“ Schaltfläche der *Delete* Spalte eine Oberfläche angezeigt werden. Diese zeigt die selben Informationen wie die Oberfläche zum Editieren an und bietet dem Nutzer die Möglichkeit, das Ergebnis zu löschen.

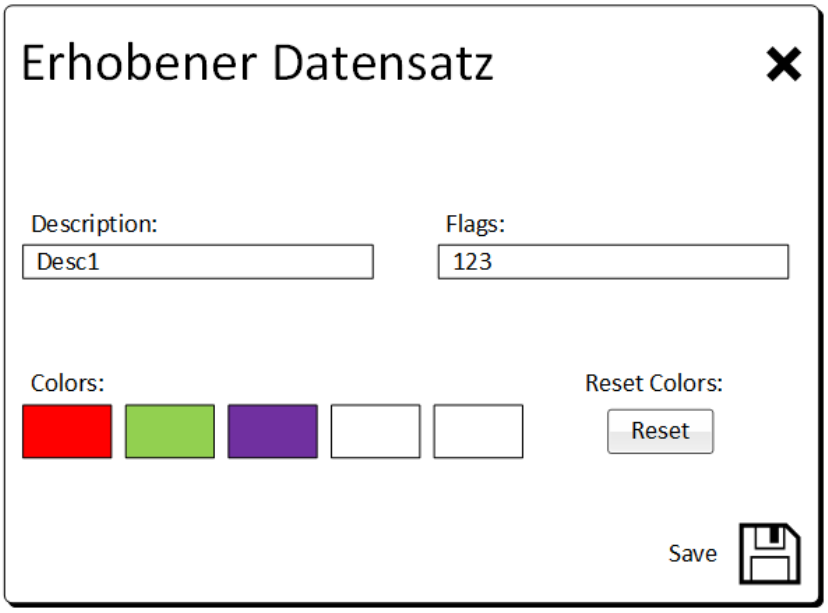

Abbildung 3.3: Entwurf der Oberfläche zum Editieren von Datensätzen

3.3 Architektur

Anhand der zuvor in Abschnitt 3.1 herausgearbeiteten Anforderungen an das zu entwickelnde System und dem vorgestellten Konzept der Anwendung in Abschnitt 3.2 wird in diesem Abschnitt die grundlegende Architektur der entwickelten Webanwendung vorgestellt. Abbildung 3.4 zeigt ein Diagramm, welches abstrakt alle Komponenten der Applikation sowie mögliche Interaktionen eines Benutzers mit dem System grafisch darstellt.

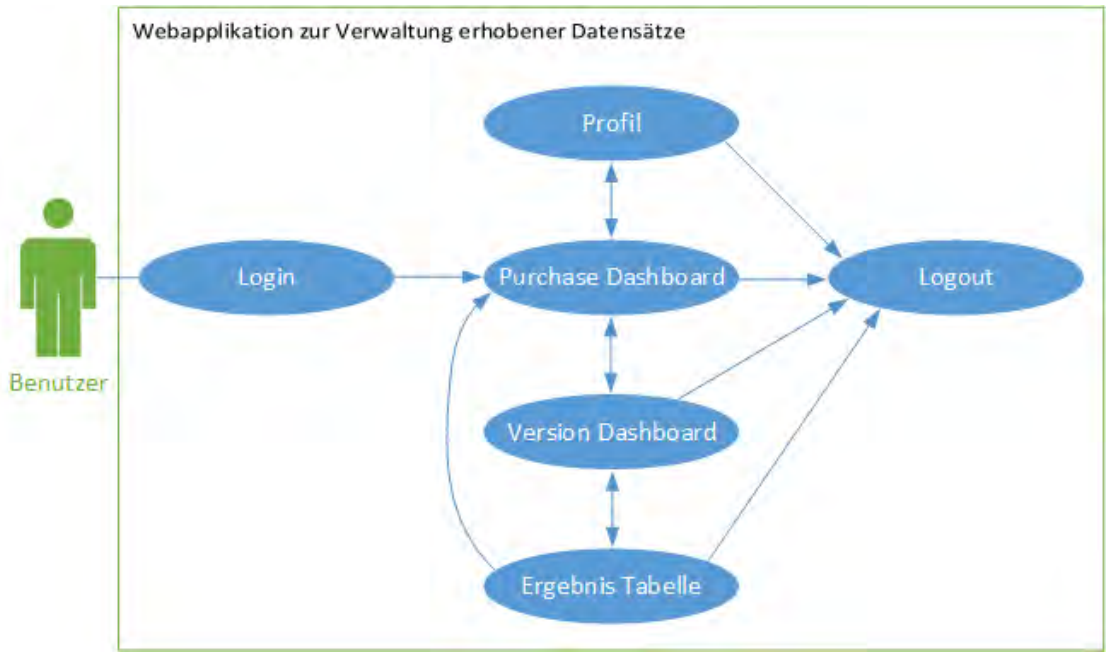

Abbildung 3.4: Architektur der Webapplikation

Jeder unauthentifizierte Benutzer gelangt zunächst zu der *Login Komponente* des Systems. Die *Login Komponente* ermöglicht dem Nutzer sich an dem System anzumelden. Autorisierte Benutzer sind in der Lage die *Profil Komponente* aufzurufen, welche die im System hinterlegten Informationen des Benutzers bereitstellt.
Die *Purchase Dashboard* Komponente listet für eingeloggte Benutzer alle ihm zugewiesenen Fragebögen auf. Von dieser Komponente aus kann das *Version Dashboard* erreicht werden.
In dem *Version Dashboard* werden alle verfügbaren Versionen eines ausgewählten

Fragebogens dargestellt. Durch die Selektion einer Version gelangt man schließlich zu einer Übersicht, welche alle gesammelten Ergebnisse anzeigt. Hierfür ist die *Ergebnis Tabelle* Komponente zuständig.
Die *Logout* Funktion steht jedem Benutzer zur Verfügung, der erfolgreich an dem System angemeldet ist und sorgt dafür, dass der Benutzer sich vollständig von dem System abmelden kann.

4

Implementierung

Dieses Kapitel führt zunächst die verwendeten Templates auf, die für die Entwicklung der Webapplikation verwendet wurden. Im Anschluss daran werden einzelne Bereiche der Implementierung genauer betrachtet und deren Umsetzung beschrieben.

4.1 Vorlagen

Für die Entwicklung der Webanwendung wurden bestimmte Templates, Inspinia und Angular Material, verwendet. Das Inspinia Template gewährleistet unter anderem, dass die Benutzeroberfläche der Webapplikation Ähnlichkeiten zu bereits bestehenden Anwendungen des QuestionSys Projekts aufweist. Ein Grund hierfür ist, dass die Anwendungen aufgrund des Templates die selben CSS (Cascading Style Sheets) Dateien zur Gestaltung der Weboberfläche nutzen. Dadurch wird dem Benutzer direkt ein bekanntes und vertrautes Gefühl übermittelt. Dies ist förderlich für die Zufriedenheit des Nutzers, eines der drei Ziele der Gebrauchstauglichkeit [17]. Angular Material bietet Templates für viele Benutzeroberflächen-Komponenten an. Die Verwendung dieser Komponenten erspart dem Entwickler Zeit und Implementierungsaufwand.

4.1.1 Inspinia

Das Inspinia Template [18] wurde als Grundgerüst, insbesondere für die grafische Gestaltung der Webanwendung, verwendet. Eine beispielhafte Oberfläche der Vorlage ist in Abbildung 4.1 abgebildet. Das Inspinia Template wurde bereits in mehreren Anwendungen des QuestionSys Frameworks verwendet und beinhaltet viele verschiedene

Komponenten. Diese können dafür genutzt werden, um den Implementierungsaufwand einer ansprechenden und funktionalen Benutzeroberfläche zu minimieren. Das Template beinhaltet dabei die passenden Quelldateien für verschiedenste Frameworks.

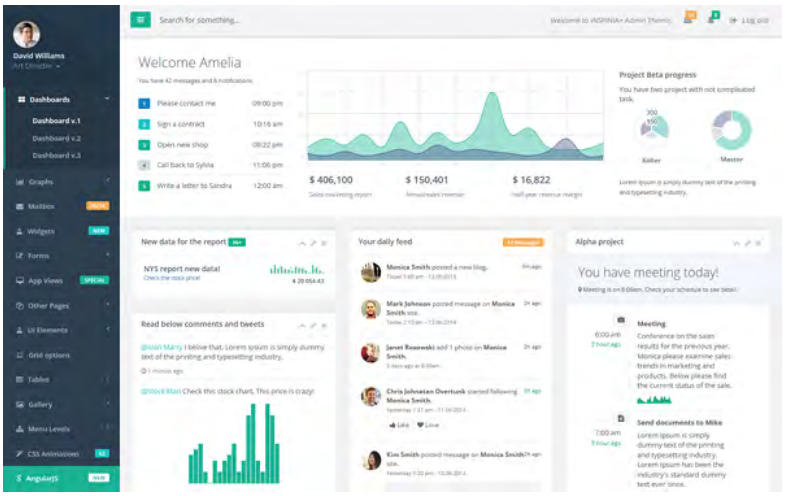

Abbildung 4.1: INSPINIA - Responsive Admin Theme [18]

Unter den verfügbaren Komponenten ist auch ein Widget. Dieses wird in den Dashboard-Übersichten sowohl für die grafische Darstellung einzelner Fragebögen als auch deren Versionen genutzt. Zur Generierung der Graphen innerhalb des Widgets wird hierfür die JavaScript Bibliothek *Flot* [19] verwendet. Für die Umsetzung des Widgets wird eine Angular Komponente erstellt und in dieser Komponente mit der *Flot* Bibliothek der Graph für die Anzahl erhobener Datensätze generiert. Listing 4.1 zeigt den benötigten HTML Codeabschnitt der Dashboard Komponente, der für die Generierung der Widgets genutzt wird. Die resultierende Oberfläche ist in Abbildung 5.3 zu sehen.

```
1  <div class="col-md-6 col-lg-3" *ngFor="let widget of widgets">
2          <app-widget
3                  [name]="widget.name"
4                  [inputDataValues]="widget.values"
5                  [bgcolorIndex]="(widget.name.charCodeAt(0))%5"
6                  [purchaseID]="widget.purchaseID"
7                  [productID]="widget.productID"
8                  [routeString]="routeString"
9                  [total]="widget.total"
10         ></app-widget>
11 </div>
```

Listing 4.1: HTML Code zur Generierung der Widgets

Die Wiederverwertbarkeit von Angular Komponenten wird hierbei deutlich sichtbar. Die Widget Komponente wird aufgrund der strukturellen Angular Direktive *ngFor* in Zeile 1 für jedes Element des *widgets* Arrays generiert. Die Verwendung der strukturellen Direktive verändert dabei das DOM Layout, indem diesem Elemente hinzugefügt werden [11]. Die bei Angular verwendete Daten-Bindung zwischen der Angular Komponente und dem Dokumenten-Objekt-Modell (DOM) wird in Zeile 3-9 sichtbar. Der Widget Komponente werden durch das sogenannte *property binding* (Eigenschaftsbindung) die benötigten Werte von der Dashboard Übersicht übergeben [20].

4.1.2 Angular Material

Angular selbst bietet Benutzeroberflächen in Form von Angular Material Komponenten zur freien Verfügung an [21]. Diese Komponenten lassen sich leicht in das Projekt importieren und beinhalten bereits viele Funktionalitäten, sowie ansprechende Animationen. Für die Entwicklung der Webanwendung wurde Angular Material genutzt, um eine interaktive Tabelle für die erhobenen Ergebnisse zu erzeugen. Die Dialogfenster zum Editieren und Löschen einzelner Datensätze werden auch durch Angular Material Komponenten generiert.

```
import {
        MatTableModule,MatPaginatorModule,
        MatSortModule,MatCheckboxModule
} from '@angular/material';
import {MatTooltipModule} from '@angular/material/tooltip';
```

Listing 4.2: Benötigte Imports für die Generierung der Tabelle mit Angular Material

Zusätzlich zu den in Listing 4.2 genannten Imports muss noch das *BrowserAnimations-Module* importiert werden, damit die Angular Material Komponenten verwendet werden können. Für die Komponenten stehen vier verschiedene Leitmotive zur Verfügung, welche für die Farbwahl der generierten Oberfläche ausschlaggebend sind und woraus ein Motiv gewählt werden muss. Hierfür muss der in 4.3 abgebildete Codeabschnitt innerhalb der *styles.css* eingebunden werden [22].

```
@import "~@angular/material/prebuilt-themes/indigo-pink.css";
```

Listing 4.3: Auswahl des Leitmotivs

4.2 Ausgewählte Bereiche der Implementierung

Im folgenden Abschnitt sollen nun einzelne Bereiche der Implementierung genauer betrachtet werden, die für die Umsetzung des Konzepts der Webanwendung von Bedeutung sind. Dabei soll insbesondere die Auflistung und Darstellung der Fragebögen und deren Versionen betrachtet werden.

Dashboard Übersicht

Für eine möglichst informative und intuitive Darstellung der Fragebögen, wurde im Zuge dieser Arbeit das Konzept eines Dashboards gewählt. Für die Anzeige des Dashboards wird hierfür eine Angular Komponente erstellt. Das Dashboard der verfügbaren Versionen eines Fragebogens wird hierzu analog generiert. In der Dashboard Komponte

wurde das Widget Konstrukt aus der Inspinia Vorlage benutzt. Da in der vorliegenden Version 2.7.1 des Inspinia Templates lediglich ein Angular 4 Seed Projekt enthalten ist, musste der Quellcode zunächst angepasst werden, um in der Anwendung verwendet werden zu können. Dies hat den Grund, dass Angular einen Veröffentlichungszyklus von circa einem halben Jahr verfolgt [23] und die einzelnen Versionen teilweise nicht abwärtskompatibel sind. Die entwickelte Webanwendung läuft zum Zeitpunkt der Bachelorarbeit in Angular Version 6.0.0 . Für die dynamische Anordnung der Widgets wurde das Tabellen System von Bootstrap [24] genutzt. Die resultierende Benutzeroberfläche ist für die Übersicht der erworbenen Fragebögen in Abbildung 5.3 sowie für die Versions Übersicht in Abbildung 5.4 zu sehen.

Datenbeschaffung

Um dem Benutzer die benötigten Daten anzeigen zu können, müssen diese zuvor vom Server abgefragt werden. Angular Komponenten sollten selbst keine Daten von einem Server beschaffen [25]. Es gibt in Angular die Konvention, diese Datenbeschaffung von dafür erstellten *Services* erledigen zu lassen. Die Verwendung von Services sorgt dafür, dass die Komponenten so wenig Codezeilen wie möglich enthalten. Die Wiederverwertbarkeit innerhalb einer Anwendung wird gesteigert, indem Komponenten nicht für die Datenbeschaffung genutzt werden [26]. Für die Beschaffung der Daten werden daher mehrere Service Klassen erstellt. Diese werden per *Dependency Injection* in die Komponenten der Anwendung eingefügt. Die Komponenten beschaffen dadurch die Daten daher nicht selbst sondern lassen die Service Klassen dies erledigen [27].
Listing 4.4 zeigt hierbei die Injektion des *DashboardService* in die Dashboard Komponente der Anwendung. Dieser Service wird bei der Initialisierung der Dashboard Komponente verwendet, um alle benötigten Daten für das Dashboard zu laden.

```
1 constructor(
2         private dashboardService: DashboardService
3 ){}
```

Listing 4.4: Dependency Injection des *DashboardService* in die Dashboard Komponente

In den *DashboardService* werden zwei weitere Services injiziert. Die Generierung der URL (Uniform Ressource Locator) wird von dem *UriService* übernommen. Die URL wird dazu genutzt, die benötigten Ressourcen auf dem Server zu adressieren. Der *ApiService* ist (stark vereinfacht) hauptsächlich dafür zuständig, die HTTP (Hypertext Transfer Protocol) Anfragen an die QuestionSys API zu tätigen und die vom Server übertragenen JSON Objekte auf die Modelle abzubilden. Abbildung 4.2 zeigt eine vereinfachte Darstellung der Service-Injektionen, die benötigt werden, um sämtliche Daten für die Dashboard Komponente zu erhalten.

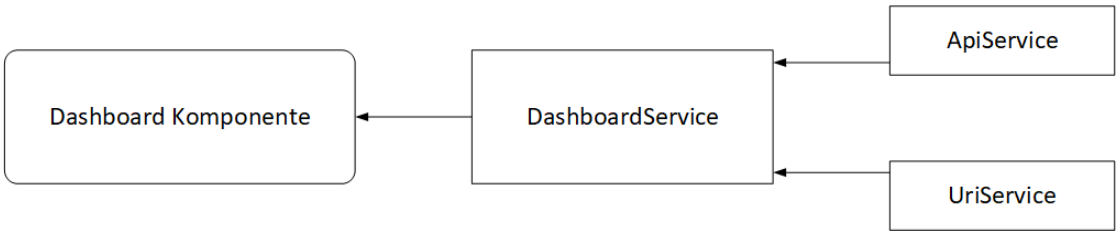

Abbildung 4.2: Schema der Service-Injektionen für die Dashboard Komponente

Modellierung der Daten

Die von dem Server erhaltenen Daten werden mithilfe des *ApiService* den entsprechenden Modellen zugewiesen. Die Modelle werden unter Zuhilfenahme der Pakete *QuestionSys-Model* und *questionsys-client-api-connector* erstellt. Die Pakete wurden für die Modellierung sowie den Datenaustausch mit der QuestionSys API erstellt und für die Entwicklung der Webanwendung zur Verfügung gestellt. Mit Modellen lassen sich die erhaltenen Daten abbilden, um mit diesen zu interagieren. Listing 4.5 zeigt die Modellierung eines Fragebogens. Die Attribute des Modells sind hierbei mit dem *@Attribute* Dekorierer versehen.
Der *object* Wert ist in diesem Fall „Product“. Der Name des Fragebogens wird in *name* gespeichert. *Description* enthält die Beschreibung eines Fragebogens. Die Schlüsselwörter eines Fragebogens werden in *keywords* angezeigt. Ob der Fragebogen veröffentlicht und aktiv ist, wird in *is_featured* und *is_active* festgehalten. Die Kontaktdaten der zuständigen Person sind in *contact* abgebildet. Um Fragebögen des QuestionSys Projekts nutzen zu können, müssen diese erworben werden. Der Preis eines Fragebogens

wird in *price* dargestellt. Die Attribute *created_at* und *updated_at* besagen, wann der Fragebogen erstellt und wann dieser zuletzt verändert wurde.

```
import {ApiModel, Attribute, Model}
        from 'questionsys-client-api-connector';

/**
 * Product Model
 */
@Model({
    type: 'products'
})

export class ProductModel extends ApiModel {

    @Attribute()
    object: string;

    @Attribute()
    name: string;

    @Attribute()
    description: string;

    @Attribute()
    keywords: string[];

    @Attribute()
    is_featured: boolean;

    @Attribute()
    is_active: boolean;

    @Attribute()
    contact: {
        name: string,
        email: string,
        uri: string
    };

    @Attribute()
    price: {
        model: {
            amount: string,
            currency: string
        },
        instance: {
            amount: string,
```

```
46              currency: string
47          }
48      };
49
50      @Attribute()
51      created_at: {
52          date: string,
53          timezone_type: number,
54          timezone: string
55      };
56
57      @Attribute()
58      updated_at: {
59          date: string,
60          timezone_type: number,
61          timezone: string
62      };
63  }
```

Listing 4.5: Modell eines Fragebogens

Das Modell des Fragebogens ist dabei durch eine Beziehung an das *PurchaseModel* gebunden. Das *PurchaseModel* beschreibt die Zusammengehörigkeit zwischen einem Fragebogen (*ProductModel*) und dessen Versionen (*ProductVersionModel*) mitsamt der jeweiligen Anzahl an Ergebnissen (*ProductResultAmountModel*, *ProductVersionResultAmountModel*). Die Beziehungen werden durch den *@Relationship* Dekorierer modelliert. Das *PurchaseModell* steht wiederum in einer Beziehung zu einem *DashboarddetailModel*. Abbildung 4.3 zeigt die hierarchische Struktur des *DashboarddetailModels* in Hinblick auf die beziehungsbedingten Verbindungen zu anderen Modellen. Das *DashboarddetailModel* kann hierbei eine Verbindung zu mehreren *PurchaseModels* besitzen. Das *PurchaseModel* besitzt Beziehungen zu dem *ProductModel*, dem *ProductVersionModel*, dem *ProductResultAmountModel* und dem *ProductVersionResultAmountModel*. Diese Art der Modellierung ermöglicht eine effiziente Interaktion mit den empfangenen Daten des Servers.

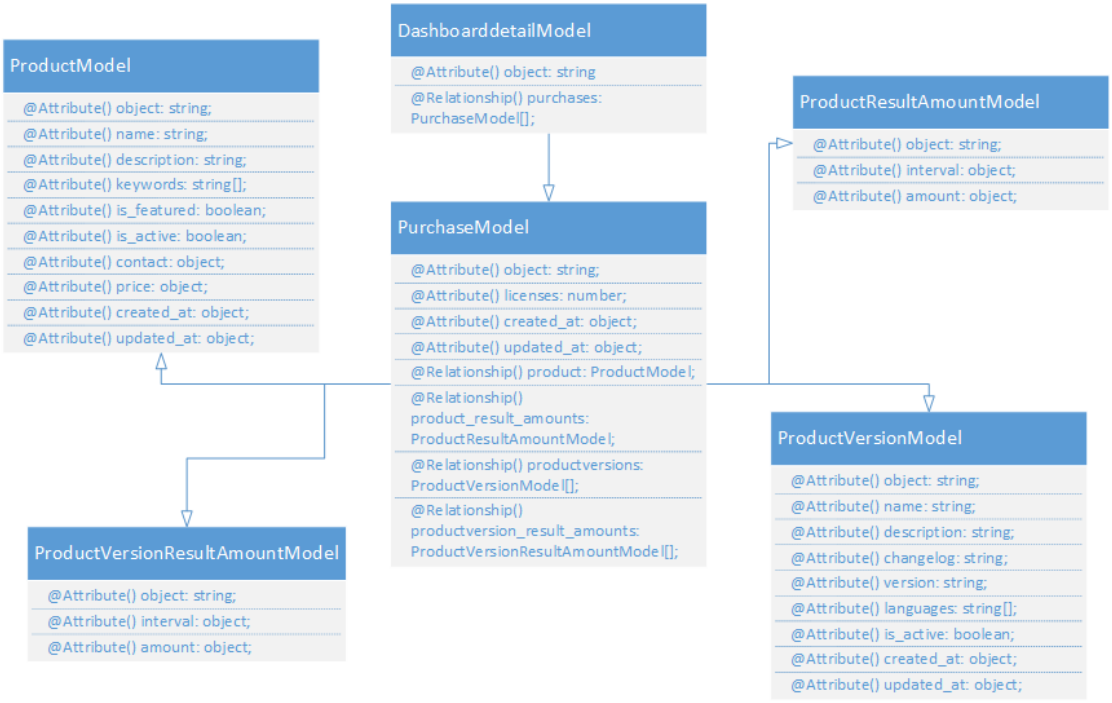

Abbildung 4.3: Hierarchische Struktur des DashboarddetailModels

Navigation zwischen Angular Komponenten

Für die Navigation zwischen den einzelnen Komponenten der Webanwendung wird der Router von Angular verwendet. Die Verwendung des Angular Routers wird von Victor Savkin in [28] detailliert beschrieben. Um in der Anwendung zwischen den verschiedenen Ansichten zu navigieren, wird hierfür ein *AppRoutingModule* erstellt. Das Modul beinhaltet alle verfügbaren Routen der Webanwendung. Listing 4.6 zeigt hierbei einen Ausschnitt aus dem *AppRoutingModule* mit den vorhandenen Routen der Applikation. Die Routen sind dabei als JSON Objekt gespeichert und beinhalten den Pfad, sowie die dazugehörige Angular Komponente. Der verwendete *AuthGuard* Service sorgt dafür, dass bestimmte Pfade der Anwendung ausschließlich von autorisierten Benutzern aufgerufen werden können. Um die Autorisierung eines Nutzers zu verifizieren, wird überprüft, ob das beim Login erhaltene Token vorhanden ist. Der *Router* wird per Dependency Injection in die Angular Komponenten eingefügt. Danach kann die *navigate()*

Funktion mit den gewünschten Parametern aufgerufen werden. Listing 4.7 zeigt den benötigten Codeabschnitt, um von dem Dashboard der aufgelisteten Fragebögen zu dem Dashboard mit den dafür existieren Versionen zu navigieren.

```
const appRoutes: Routes = [

    // Login Komponente
    {path: 'login', component: LoginComponent},

    // Komponente zum Anzeigen des Profils
    {path: 'editUserData', component: DisplayUserDataComponent,
        canActivate: [AuthGuard]},

    // Dashboard Komponente mit den zugewiesenen Frageboegen
    {path: 'dashboard', component: DashboardComponent,
        canActivate: [AuthGuard]},

    // Dashboard Komponente mit den Versionen eines Fragebogens
    {path: 'products/:productid/versions',
        component: PurchaseoverviewComponent,
        canActivate: [AuthGuard]},

    // Ergebnistabellen Komponente
    {path: 'analysis/purchases/:purchaseid/products/:productid/
        productversions/:productversionid/results',
        component: ResultTableComponent,
        canActivate: [AuthGuard]},

    // Main redirect
    {path: '', redirectTo: 'login', pathMatch: 'full'}
];
```

Listing 4.6: Vorhandene Routen der Webanwendung

```
// DashboardComponent zu PurchaseoverviewComponent
this.router.navigate(
        ['products/' + this.productID + '/versions']
);
```

Listing 4.7: Codeabschnitt für die Navigation zwischen den Dashboards

5

Vorstellung der Anwendung

In diesem Kapitel soll die innerhalb der Arbeit entwickelte Webanwendung vorgestellt werden. Hierzu werden Screenshot der Anwendungskomponenten abgebildet und beschrieben. Die dargestellten Daten gehören hierbei zu einem Testaccount des QuestionSys Frameworks, welcher für die Entwicklung des Systems genutzt wurde. Zusätzlich werden die umgesetzten funktionalen Anwendungen an der entsprechenden Stelle genannt.

5.1 Allgemein

Der Benutzer soll in der Lage sein, seine Profildaten anzeigen zu lassen. Abbildung 5.1 zeigt hierbei einen Screenshot der Komponente, die für die Anzeige der Profildaten zuständig ist. Der Nutzer ist jedoch nicht in der Lage, diese Daten zu ändern. Eine Angular Anwendung, die für die Verwaltung von Benutzer Accounts zuständig ist, existiert bereits. Die Schaltfläche *Change Data* soll demnach dafür verwendet werden können, um den Benutzer zu der Account Verwaltungs Anwendung des QuestionSys Projekts weiterzuleiten. Durch die Ansicht werden **FA1, FA2** abgedeckt.
Abbildung 5.2 zeigt eine einen Screenshot der Login Komponente. Beim Login versendet die Anwendung die Benutzeranmeldeinformationen an der Server und erhält daraufhin ein *token*, welches bei jeder darauffolgenden Anfrage an den Server zur Authentifikation des Nutzers mitgeschickt wird.

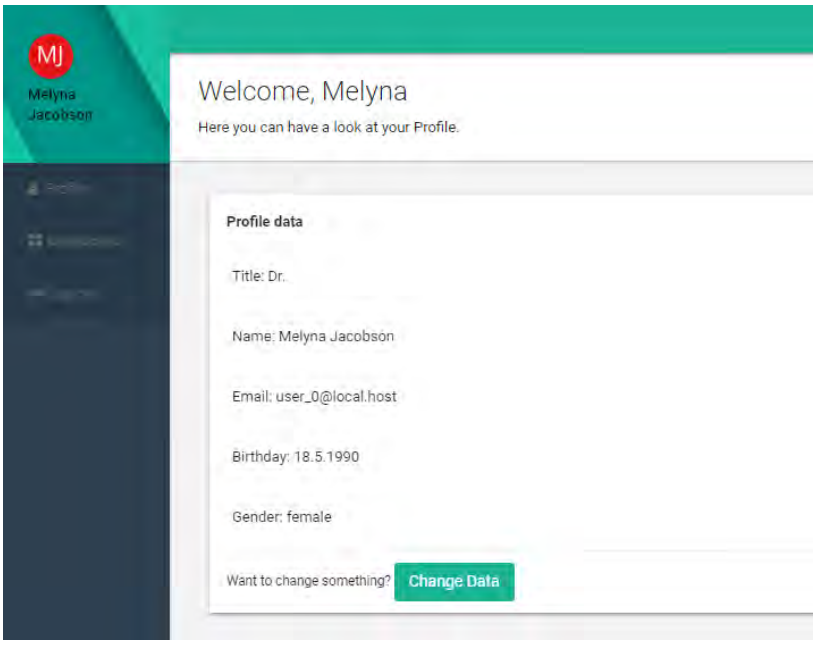

Abbildung 5.1: Übersicht über die Profildaten

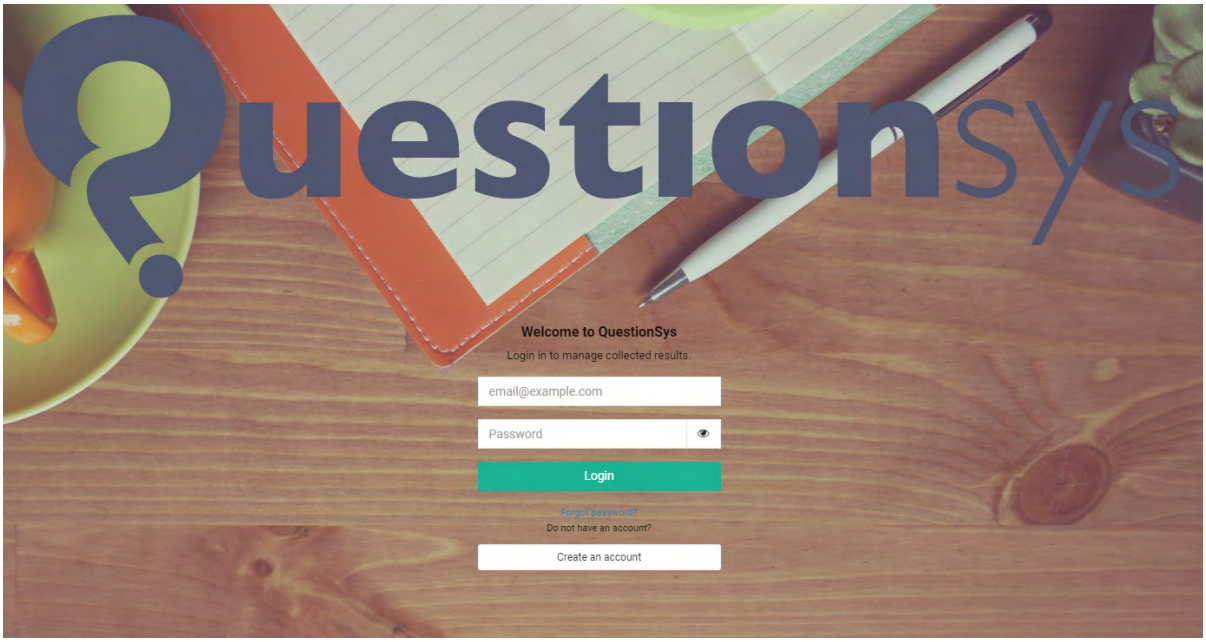

Abbildung 5.2: Login-Bildschirm der Webanwendung

5.2 Dashboard

Abbildung 5.3 zeigt die Oberfläche des Dashboards mit den dem Nutzer zugeordneten Fragebögen. Dieses Dashboard dient als Startseite der Applikation nach dem Einloggen des Benutzers. Unabhängig von der aktuellen Komponente wird zu jederzeit während der Verwendung der Webapplikation auf der linken Seite eine Leiste angezeigt, die zum einen benutzerspezifische Informationen wie den Namen des eingeloggten Benutzers und das dazugehörige Icon anzeigt **(1)**, sowie ein kleines Navigationsmenü beinhaltet. Innerhalb des Menüs kann der Benutzer zu seinen Profilinformationen navigieren **(2)** (dargestellt in Abbildung 5.1), zurück auf das Dashboard der Fragebögen gelangen **(3)** oder sich vom System abmelden **(4)**. Innerhalb des Dashboards werden die Fragebögen durch die Widgets **(5)** repräsentiert. Diese enthalten den Namen des Fragebogens **(6)**, die Anzahl an erhobenen Daten **(7)** und ein Diagramm mit den innerhalb der letzten 30 Tagen gesammelten Ergebnissen **(8)**. Die für den Screenshot dargestellten Diagramme beinhalten hierbei zufällig generierte Daten, da zur Zeit der Aufnahme keine Ergebnisse innerhalb der letzten 30 Tagen erhoben wurden. Zudem ist bereits der Anfang einer Brotkrumen-Navigation erkennbar **(9)**. Innerhalb der abgebildeten Ansicht werden **FA2,FA3,FA5-FA9** abgedeckt.

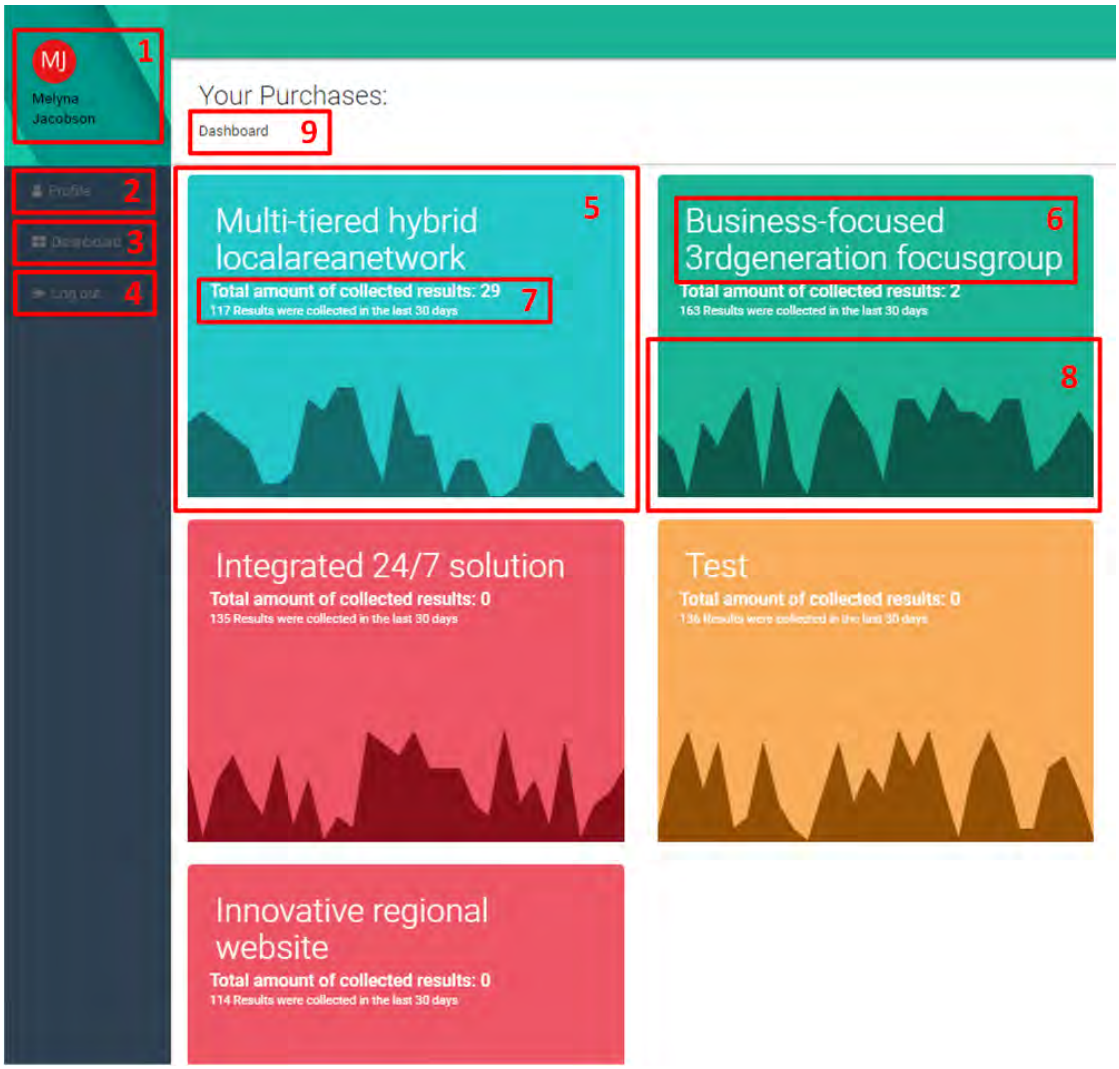

Abbildung 5.3: Dashboard mit den Fragebögen des Benutzers

5.3 Versionsübersicht

Das in Abbildung 5.4 dargestellte Dashboard mit den Versionen eines Fragebogens ist sehr ähnlich zu dem zuvor in Abschnitt 5.2 beschriebenen Dashboard aufgebaut. Man gelangt auf diese Übersicht, indem man in dem zuvor vorgestellten Dashboard auf eines der abgebildeten Widgets klickt. Die Übersicht unterscheidet sich lediglich darin, dass die abgebildeten Widgets nun die Versionen eines spezifischen Fragebogens repräsentieren anstatt die Fragebögen. Die Brotkrumen-Navigation bietet nun die Möglichkeit, auf die Übersicht der zugewiesenen Fragebögen zu gelangen. Durch das Klicken auf ein Widget einer bestimmten Version gelangt der Benutzer auf die Übersicht der erhobenen Ergebnisse. Die Ansicht beinhaltet zeigt die Umsetzung von **FA2,FA4-FA9**.

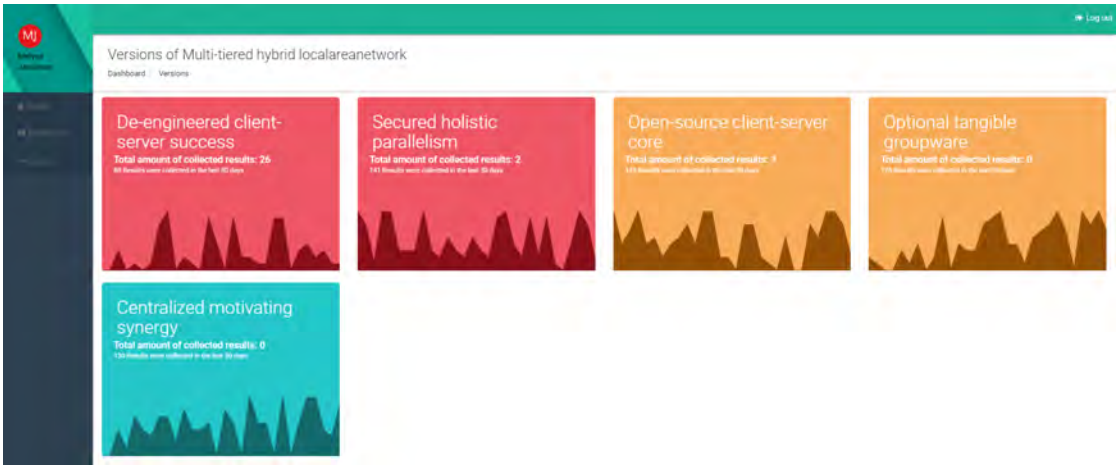

Abbildung 5.4: Dashboard der verfügbaren Versionen eines Fragebogens

5.4 Auflistung der Ergebnisse

Nachdem der Benutzer eine Version ausgewählt hat, gelangt er auf die Übersicht mit den erhobenen Daten eines Fragebogens in der selektierten Version. Diese Übersicht ist in Abbildung 5.5 dargestellt. Dem Benutzer wird eine interaktive Tabelle angezeigt. Die Brotkrumen-Navigation **(1)** kann dazu genutzt werden kann, um zurück zu den verfügbaren Versionen oder auf die Übersicht der Fragebögen zu gelangen. In der Tabelle

entspricht jede Zeile einem gesammeltem Ergebnis eines Fragebogens. Bereich **(2)** zeigt dem Benutzer alle verfügbaren Informationen zu den Ergebnissen an. Hierbei werden die gesetzten Farben eines Fragebogens durch einen farbigen Kreis und nicht textuell dargestellt. Durch Klicken auf die Einträge in der Kopfzeile der Tabelle können die Ergebnisse auf und absteigend sortiert werden. Die Editieren Spalte **(3)** bietet dem Benutzer die Möglichkeit, einzelne Ergebnisse zu bearbeiten. Hierfür wird ein Dialogfenster geöffnet, welches in Abschnitt 5.5 beschrieben und abgebildet ist. Ähnlich hierzu können mithilfe der Löschen Spalte **(4)** einzelne Ergebnisse gelöscht werden. Auch für diese Aktion öffnet sich ein Dialogfenster, beschrieben und abgebildet in Abschnitt 5.5 . Bereich **(5)** beinhaltet für den Benutzer die Möglichkeit, die Anzahl an Ergebnissen pro Seite anzupassen und durch die Seiten zu navigieren. Die Gesamtanzahl der vorhandenen Ergebnisse wird ebenfalls dort abgebildet. Im Hinblick auf mögliche Erweiterungen des Systems wurde bereits implementiert, mehrere Ergebnisse selektieren zu können **(6)**. Nach der Auswahl mehrerer Ergebnisse wird bereits unterhalb der Tabelle ein weiterer Bereich mit Interaktionsmöglichkeiten des Benutzers eingeblendet. Dieser dient jedoch aktuell noch als Platzhalter und besitzt zum derzeitigen Standpunkt nur die Funktion die Auswahl der selektierten Ergebnisse zurückzusetzen **(7)**. Dies hat den Vorteil, dass der Benutzer nicht durch die Seiten der Ergebnisse blättern muss, falls die ausgewählten Elemente sich auf verschiedenen Seiten befinden, um die Auswahl zurücksetzen zu können. Durch den abgebildeten Bereich wird die Umsetzung der Anforderungen **FA2,FA10-FA14** dargestellt.

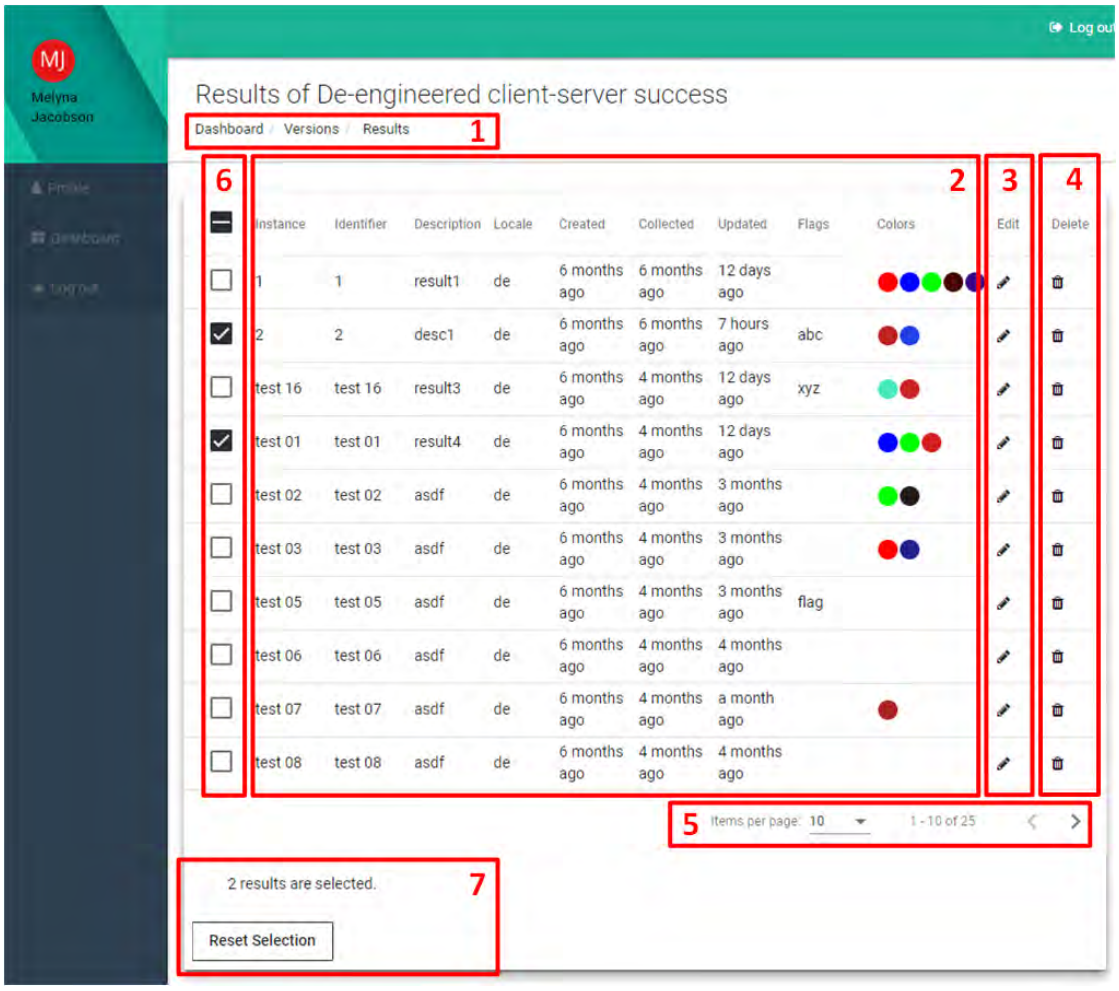

Abbildung 5.5: Tabelle der erhobenen Daten

5.5 Bearbeitung und Löschen der Ergebnisse

Um den Benutzer die Möglichkeit zu bieten, Ergebnisse bearbeiten und löschen zu können, wurden hierfür jeweils Dialogfenster erstellt. Abbildung 5.6 zeigt hierbei den Dialog, der anzeigt wird, wenn der Nutzer einen Ergebnisdatensatz bearbeiten möchte. Der Benutzer kann mithilfe der Ansicht die Daten eines Ergebnisdatensatzes ändern. Die Beschreibung sowie die *Flags* des Ergebnisses werden hierbei textuell in ein Textfeld dargestellt **(1,2)**. Es können bis zu fünf verschiedene Farben pro Ergebnis eingetragen werden. Dazu klickt der Benutzer in eines der Felder, wodurch sich eine weitere grafi-

sche Oberfläche zur Auswahl der Farbe öffnet **(3)**. Die Farben können durch Betätigen der *reset* Schaltfläche **(4)** zurückgesetzt werden. Änderungen werden mithilfe der *save* Schaltfläche **(5)** an den Server übertragen. Das Dialogfenster kann durch Betätigen des *Schließen* Symbols oder durch Klicken außerhalb des Dialogs geschlossen werden **(6)**. Sollten Änderungen eingetragen worden werden sein und der Nutzer will das Dialogfenster schließen, wird der Nutzer auf die nicht gespeicherten Änderungen hingewiesen. Dieses Dialogfenster beschreibt die Umsetzung von **FA20**.

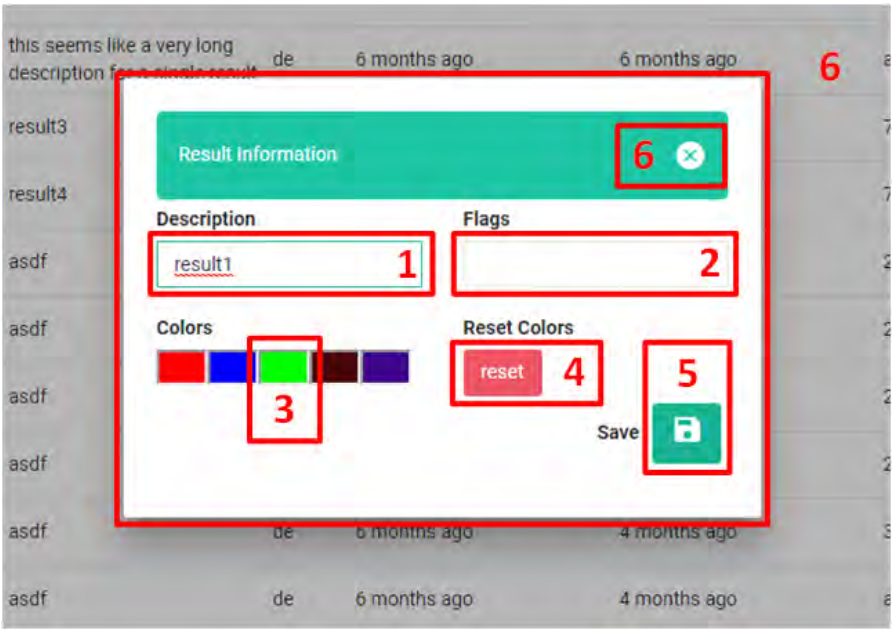

Abbildung 5.6: Dialogfenster zur Bearbeitung eines Ergebnisdatensatzes

Abbildung 5.7 zeigt das Dialogfenster, welches dem Nutzer zum Löschen von Ergebnissen angezeigt wird. Das Fenster ist dabei sehr ähnlich zu dem Dialogfenster zum Editieren von Ergebnissen aufgebaut. Es werden dem Benutzer die Beschreibung, die *Flags* sowie die Farben des Ergebnisdatensatzes angezeigt. Das Ergebnis kann durch Betätigen der *Delete* Schaltfläche vom Server gelöscht werden. Dieser Vorgang ist nicht rückgängig zu machen. Die Anzeige dieses Dialogfensters soll sicherstellen, dass der Nutzer nicht ungewollt Ergebnisse löscht und diesen Vorgang bestätigen muss. Durch diese Komponente wird **FA21** realisiert.

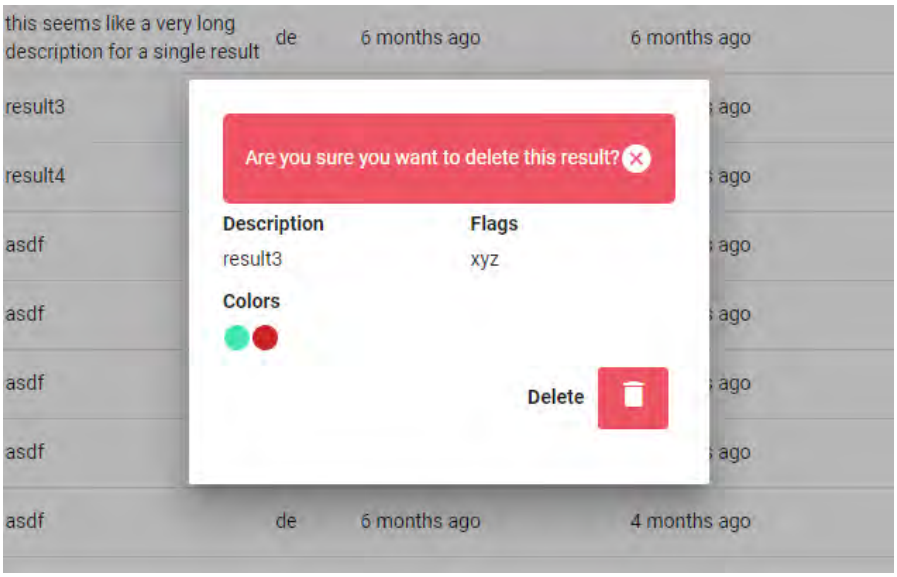

Abbildung 5.7: Dialogfenster zum Löschen eines Ergebnisdatensatzes

6

Verwandte Arbeiten

Softwareumsetzungen für das Erstellen von digitalen Fragebögen und Umfragen gibt es einige. Viele Softwarelösungen bieten dabei die Möglichkeit, die erhobenen Daten einsehen und analysieren zu können. Abschnitt 6.1 und Abschnitt 6.2 beschreiben dabei Systeme, welche den gesamten Ablauf der Datenerhebung durch digitale Fragebögen unterstützen. Wird der funktionale Hintergrund der Anwendung nicht betrachtet, lassen sich Ähnlichkeiten der entwickelten Webanwendung zu der Weboberfläche von File Hosting-Anbietern erkennen. Diese werden anhand eines Vergleichs mit der Weboberfläche von Dropbox in Abschnitt 6.3 dargestellt.

6.1 Survey Management App

Die *Survey Management App* ermöglicht einen systematischen und automatisierten Prozess, um Umfragen innerhalb eines Unternehmens durchzuführen [29]. Sie wird von der Firma MetricStream, die ihren Hauptsitz in Palo Alto, Kalifornien hat, vertrieben. Die Firma bietet unternehmensorientierte Softwarelösungen für die drei Handlungsebenen Governance (Unternehmensführung), Risk Management(Risikomanagement) und Compliance (Einhaltung gesetzlicher Vorschriften) an [30]. Die *Survey Management App* bietet dabei unter anderem die Möglichkeit, Fragebögen zu erstellen, diese zu beantworten sowie zu verwalten. Die Fragebögen können nach der Erstellung direkt an die Teilnehmer der Befragung geschickt werden. Zudem können mithilfe der Anwendung die erhobenen Ergebnisse der Fragebögen eingesehen, verwaltet und analysiert werden. Die *Survey Management App* bietet hierfür, ähnlich zu der entwickelten Anwendung, eine grafische Weboberfläche für die Verwaltung der Fragebögen sowie deren Ergeb-

nisse. Es werden ebenso verschiedene Übersichten und Tabellen verwendet, um dem eingeloggten Benutzer die Daten anzuzeigen [29]. Der Funktionsumfang der *Survey Management App* ist jedoch deutlich größer, da durch die Anwendung der gesamte Prozess der Datenerhebung unterstützt wird. Abbildung 6.1 zeigt hierbei einen Ausschnitt der Anwendung.

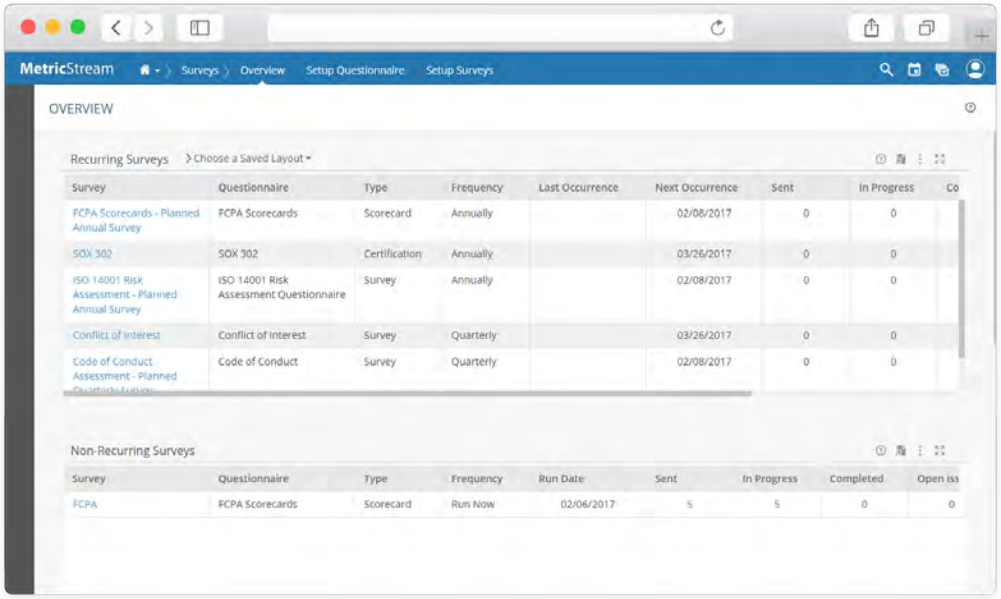

Abbildung 6.1: Oberfläche der Survey Management App [29]

6.2 SmartSurvey

SmartSurvey beschreibt eine Softwarelösung zur Erstellung von digitalen Umfragen und bietet zusätzlich die Möglichkeit, die gesammelten Ergebnisse zu analysieren [31]. *SmartSurvey* bietet für die Verwaltung von Fragebögen und deren Ergebnisse ein Management Tool an [32]. Das System ermöglicht seinen Benutzern, die Umfragen auf mobilen Endgeräten auszuführen und unterstützt mehrsprachige Umfragen [33]. Die Datenerhebung kann dabei in Echtzeit verfolgt werden und die Ergebnisse stehen direkt zur Verfügung, um diese beispielsweise in eine Excel Datei zu exportieren[34]. Die Oberfläche der Verwaltungskomponente von SmartSurvey ist in Abbildung 6.2 abgebildet. Dem Nutzer wird hierbei innerhalb einer Weboberfläche eine Tabelle angezeigt, um

die vorhandenen Umfragen zu verwalten. Von dieser Ansicht aus kann der Benutzer unter anderem die Umfragen bearbeiten, diese an Teilnehmer senden oder auch die erhobenen Ergebnisse einsehen.

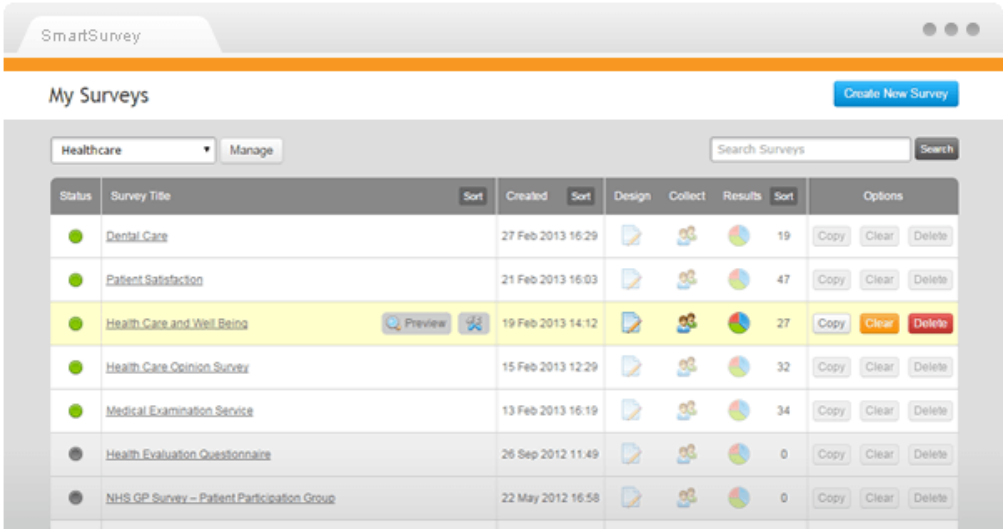

Abbildung 6.2: Oberfläche des Managementtools von SmartSurvey [32]

6.3 Dropbox

Dropbox ist ein Filehosting-Dienst. Dieser ermöglicht es, Daten online speichern und sie daraufhin Plattform unabhängig abrufen zu können. Zudem kann Dropbox dazu genutzt werden, um Daten zu zwischen verschiedenen Personen zu teilen [35].
Dropbox kann dabei sowohl durch ein heruntergeladenes Programm verwendet werden, als auch über eine Weboberfläche. Letztere besitzt bezüglich der Darstellung einzelner Elemente sowie der Navigation gewisse Ähnlichkeiten zu der entwickelten Webanwendung. Beide Webanwendungen beziehen ihre Daten plattformunabhängig von einem Server, um sie dem Nutzer darstellen zu können. Die Webanwendungen ermöglichen den Benutzern jeweils die Verwaltung von online-gespeicherten Daten. Diese können bei beiden Systemen editiert und gelöscht werden. Die abgebildeten Screenshots der Weboberfläche von Dropbox wurden durch einen privaten Dropbox Account angefertigt. Abbildung 6.3 bildet zwei Screenshots ab. Der obere Bereich **(1)** zeigt einen Ausschnitt

der Versionsübersicht der entwickelten Webapplikation, während der untere Bereich **(2)** einen Bereich der Übersicht über die vorhanden Ordner innerhalb der Dropbox Webanwendung darstellt. Hierbei entsprechen die Versionen eines Fragebogens in **(1)** den Ordnern in **(2)**.

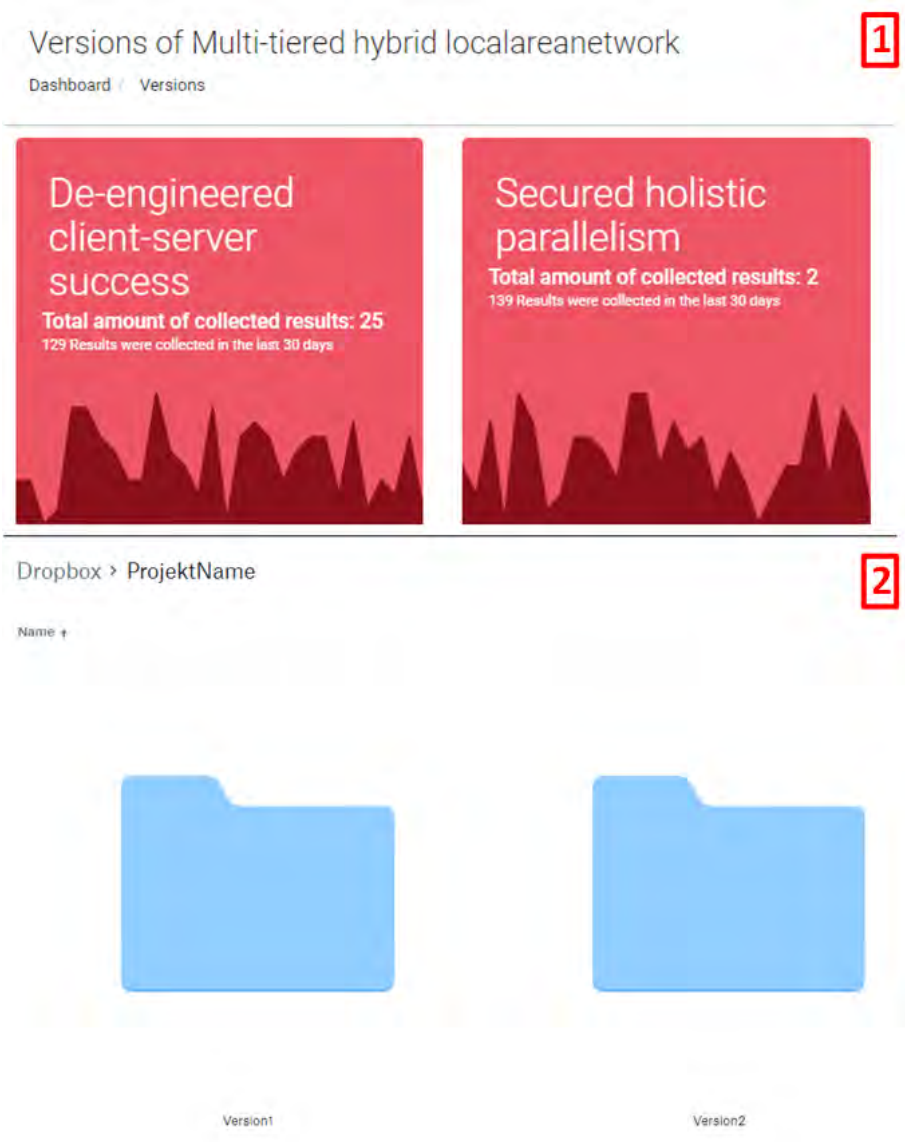

Abbildung 6.3: Screenshots von der Versionsübersicht der entwickelten Anwendung **(1)** und der Ordner Übersicht innerhalb der Dropbox-Webansicht **(2)** [36]

Abbildung 6.4 zeigt jeweils einen weiteren Ausschnitt der Webanwendungen. **(1)** entstammt wieder der entwickelten Webappliation und **(2)** der Webanwendung von Dropbox. Leicht erkennbar ist die ähnliche Anzeige der Auflistung einzelner Ressourcen. Bei beiden Anwendungen lassen sich mehrere Einträge selektieren, einzelne Ergebnisse beziehungsweise Dateien löschen oder auch editieren. Die Brotkrumen-Navigation wird in beiden Anwendungen unterstützt.

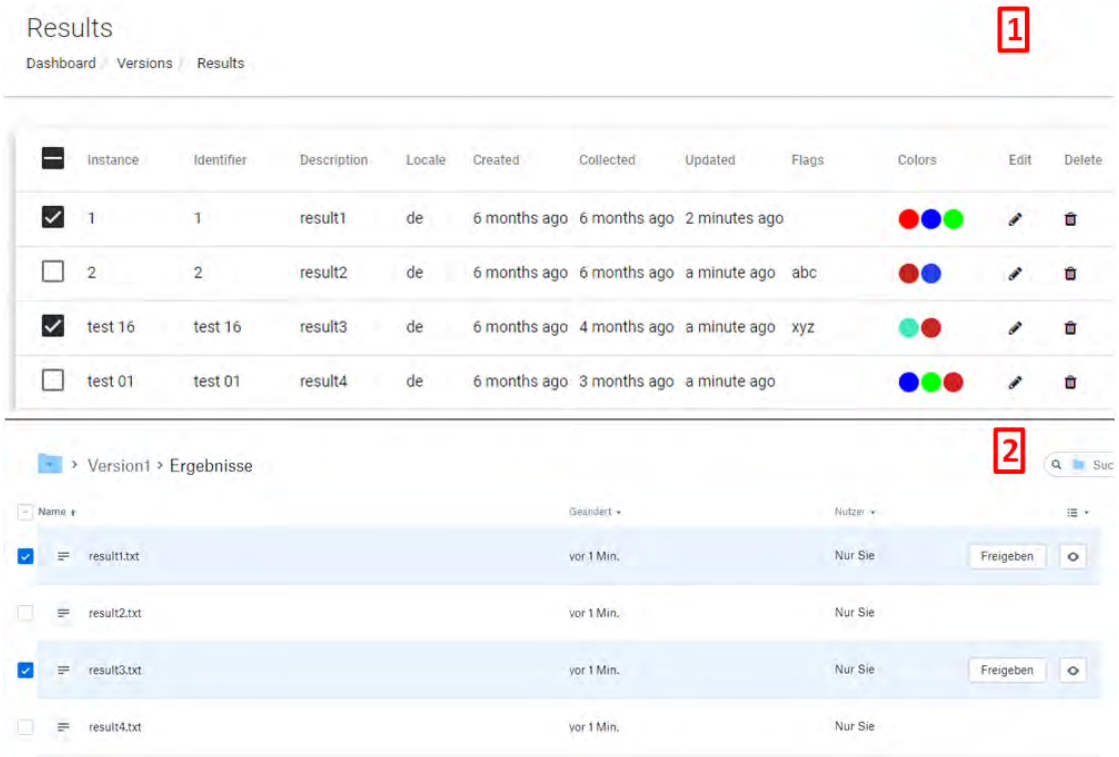

Abbildung 6.4: Screenshots von der Ergebnistabelle der entwickelten Anwendung **(1)** und der Dateiübersicht innerhalb der Dropbox-Webansicht**(2)** [36]

7

Zusammenfassung und Ausblick

In diesem Kapitel sollen abschließend die Ergebnisse der Arbeit zusammengefasst werden, um daraufhin einen Ausblick über mögliche zukünftige Erweiterungsmöglichkeiten der Webanwendung zu geben.

7.1 Zusammenfassung

Fragebögen werden vor allem in der Psychologie sowie im Gesundheitswesen verwendet, um personenbezogene Daten gezielt und effizient zu erheben. Dabei werden Fragebögen heutzutage noch vorwiegend papierbasiert verwendet, was einige Nachteile, wie etwa die erhöhte Fehlerrate aufgrund von Missachtung der Anweisungen oder aber auch Fehler beim Digitalisieren der erhobenen Daten, mit sich bringt. Das QuestionSys Framework versucht die Nachteile der papierbasierten Fragebogen Datenerhebung durch Digitalisierung dieses Prozesses zu minimieren und somit die Qualität der Ergebnisse zu optimieren. Dabei sollen auch die materiellen Kosten der Datenerhebung sowie die benötigte Zeit zur Analyse der Daten deutlich verringert werden.
Das Ziel der Arbeit war, eine Webanwendung zu entwickeln, die es einem Nutzer des QuestionSys Frameworks gestattet, sämtliche ihm zugewiesenen Fragebögen anzuzeigen und deren erfasste Ergebnisse verwalten zu können. Für die Entwicklung der Applikation wurde das Framework Angular verwendet. Während der Verwendung der Anwendung müssen Daten mit dem Server des QuestionSys Frameworks ausgetauscht werden. Die Kommunikation zwischen dem Server und der Anwendung erfolgt dabei nach den Spezifikationen der JSON:API. Die Webanwendung ist in der Lage, sämtliche dem Nutzer zugeordneten Fragebögen in einem Dashboard darzustellen. Ein Frage-

bogen wird dabei als Widget mit nützlichen Zusatzinformationen zu den gesammelten Ergebnissen dargestellt. Analog hierzu wird für die Versionen eines Fragebogens ein weiteres Dashboard generiert. Die erhobenen Ergebnisse eines Fragebogens werden in Form einer interaktiven Tabelle abgebildet. Einzelne Ergebnisse können dabei editiert oder vom Server gelöscht werden.

7.2 Ausblick

Die innerhalb dieser Arbeit erstellte Webapplikation zur Verwaltung von Ergebnissen digitaler Fragebögen ist aufgrund des verwendeten Angular Frameworks hervorragend erweiterbar. Zudem wurde während der Implementierung darauf geachtet, wiederverwertbaren Code zu schreiben, um möglichst viele Komponenten wiederverwenden zu können. Das QuestionSys Framework wird stetig weiterentwickelt. Daher ist es wahrscheinlich, dass die entwickelte Webanwendung um weitere Funktionalitäten ergänzt wird.
Möglich wäre beispielsweise, die bereits implementierten Selektionsfelder zu verwenden, um Interaktionen für mehrere Ergebnisse gleichzeitig vornehmen zu können.
Die gesammelten Ergebnisse exportieren und herunterladen zu können ist eine weitere Sinnvolle Erweiterung. Die Ergebnisse könnten daraufhin mit einer geeigneten Analysesoftware ausgewertet werden.
Ein Bezahlsystem, welches für den Fall genutzt werden könnte, dass erhobenen Daten vor dem exportieren bezahlt werden müssten, ist vorstellbar. Die Anwendung müsste dafür dem Benutzer die Möglichkeit bieten, die Datensätze in bezahlte sowie nicht bezahlte Resultate zu unterteilen und die Bezahlung der noch nicht erworbenen Ergebnisse durchzuführen.
Eine denkbare Erweiterung wäre unter anderem auch, dass der Nutzer der Webanwendung über eine Schaltfläche innerhalb der Ergebnistabelle eine Benachrichtigung an die befragten Personen verschicken könnte. Dies wäre unter anderem in der Medizin sehr sinnvoll, um beispielsweise Patienten an eine bevorstehende Untersuchung zu erinnern.
Dem Nutzer eine Vorschaukomponente der Ergebnisse zur Verfügung zu stellen ist eine weitere Erweiterungsmöglichkeit der Webanwendung. Die Entwicklung einer solchen

Komponente wurde in der Bachelorarbeit von Valentin Herter vorgestellt [37]. Diese könnte dafür genutzt werden, die Ergebnisse der selektierten Daten mithilfe von visuellen Graphen darstellen zu lassen, um vorab eine informative Übersicht über die Ergebnisse zu erhalten.

Literaturverzeichnis

[1] Schobel, J., Schickler, M., Pryss, R., Reichert, M.: Process-Driven Data Collection with Smart Mobile Devices. In: 10th International Conference on Web Information Systems and Technologies (Revised Selected Papers). Number 226 in LNBIP. Springer (2015) 347–362

[2] Schobel, J.: A Model-Driven Framework for Enabling Flexible and Robust Mobile Data Collection Applications. Dissertation an der Universität Ulm (2018)

[3] 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: 29th International Conference on Advanced Information Systems Engineering (CAiSE 2017). Number 10253 in LNCS, Springer (2017) 60–75

[4] Schobel, J., Pryss, R., Schickler, M., Reichert, M.: A Configurator Component for End-User Defined Mobile Data Collection Processes. In: Demo Track of the 14th International Conference on Service Oriented Computing (ICSOC 2016). (2016)

[5] Schobel, J., Pryss, R., Schickler, M., Reichert, M.: A Lightweight Process Engine for Enabling Advanced Mobile Applications. In: 24th International Conference on Cooperative Information Systems (CoopIS 2016). Number 10033 in LNCS, Springer (2016) 552–569

[6] 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: 5th IEEE International Conference on Mobile Services (MS 2016), IEEE Computer Society Press (2016) 1–8

[7] Andrews, K.: QuestionSys - Universität Ulm. `https://www.uni-ulm.de/in/iui-dbis/forschung/laufende-projekte/questionsys/` (2018) Besucht: 26.11.2018.

[8] stackoverflow.com: Developer Survey Results 2018 (https://insights.stackoverflow.com/survey/2018/#technology) Besucht: 10.11.2018.

[9] Schlesselman, D.: Pros & Cons of Front-End Frameworks. https://envisionitagency.com/blog/2018/04/pros-cons-front-end-web-frameworks/ (2018) Besucht: 27.11.2018.

[10] angular.io: Architecture overview (https://angular.io/guide/architecture) Besucht: 28.11.2018.

[11] Clow, M., ed.: Angular 5 projects: learn to build single page web applications using 70+ projects. Apress (2018)

[12] Kasagoni, S.K., ed.: Building modern web applications using Angular: Learn how to create rich and compelling web applications with Angular. Packt Publishing, Birmingham, UK (2017)

[13] json.org: Introducing JSON (https://www.json.org/) Besucht: 28.11.2018.

[14] jsonapi.org: JSON:API — A specification for building APIs in JSON (https://jsonapi.org/) Besucht: 28.11.2018.

[15] Brala, B.: Introduction to the JSON API. https://laravel-news.com/json-api-introduction (2018) Besucht: 28.11.2018.

[16] jsonapi.org: Latest Specification (v1.0) (https://jsonapi.org/format/) Besucht: 27.11.2018.

[17] iso.org: Ergonomics of human-system interaction - Part 11: Definitions and concepts (https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en) Besucht: 20.11.2018.

[18] webapplayers.com: Inspinia Admin v2.8 (http://webapplayers.com/inspinia_admin-v2.8/) Besucht: 20.11.2018.

[19] flotcharts.org: Attractive JavaScript plotting for jQuery (https://www.flotcharts.org/) Besucht: 20.11.2018.

[20] angular.io: Data binding (https://angular.io/guide/architecture-components#data-binding) Besucht: 20.11.2018.

[21] material.angular.io: Angular Material - Material Design components for Angular (https://material.angular.io/) Besucht: 20.11.2018.

[22] material.angular.io: Getting started (https://material.angular.io/guide/getting-started) Besucht: 20.11.2018.

[23] angular.io: Angular versioning and releases (https://angular.io/guide/releases) Besucht: 21.11.2018.

[24] getbootstrap.com: Grid system (https://getbootstrap.com/docs/4.1/layout/grid/) Besucht: 21.11.2018.

[25] angular.io: Services (https://angular.io/tutorial/toh-pt4) Besucht: 21.11.2018.

[26] Salehi, S., ed.: Angular Services: design state-of-the-art applications with customized Angular services. Packt Publishing, Birmingham, UK (2017)

[27] angular.io: Dependency Injection in Angular (https://angular.io/guide/dependency-injection) Besucht: 21.11.2018.

[28] Savkin, V., ed.: Angular router. Packt Publishing, Birmingham, UK (2017)

[29] metricsctream.com: Survey Management App. https://www.metricstream.com/apps/survey-management.htm (2018) Besucht: 28.11.2018.

[30] metricstream.com: About Us. https://www.metricstream.com/about-us (2018) Besucht: 28.11.2018.

[31] smartsurvey.co.uk: User-friendly survey software with advanced features. https://www.smartsurvey.co.uk/tour (2018) Besucht: 28.11.2018.

[32] smartsurvey.co.uk: Survey Management Tools. https://www.smartsurvey.co.uk/features/survey-management (2018) Besucht: 28.11.2018.

[33] smartsurvey.co.uk: Accessibility & Compatibility. https://www.smartsurvey.co.uk/features/survey-accessibility-and-compatibility (2018) Besucht: 28.11.2018.

[34] smartsurvey.co.uk: Reporting & Results Analysis. https://www.smartsurvey.co.uk/features/survey-reporting (2018) Besucht: 28.11.2018.

[35] dropbox.com: Was ist Dropbox? (https://www.dropbox.com/features) Besucht: 25.11.2018.

[36] dropbox.com: Dropbox. https://www.dropbox.com/h (2018) Besucht: 25.11.2018.

[37] Herter, V.: Entwicklung eines Dashboards zur Analyse digitaler Fragebögen. Bachelorarbeit an der Universität Ulm (2018)

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Benedikt Fels Matrikelnummer: 845691

Erklärung

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

Ulm, den .....

Benedikt Fels