ulm university universität
**u**ulm

**Universität Ulm** | 89069 Ulm | Germany

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

# Konzeption und Entwicklung einer Web-Plattform für die Definition, Durchführung und Auswertung von Studien im Kontext von Business Process Management

Masterarbeit an der Universität Ulm

**Vorgelegt von:**
Fabian Kreßmann
fabian.kressmann@uni-ulm.de

**Gutachter:**
Prof. Dr. Manfred Reichert
Dr. Rüdiger Pryss

**Betreuer:**
Michael Winter

2019

Fassung 24. Februar 2019



Satz: PDF-LaTeX 2$_\varepsilon$

# Kurzfassung

Um Vorgänge und Arbeitsabläufe besser zu verstehen und zu abstrahieren, werden sogenannte Geschäftsprozesse in Modellen dargestellt. Gute Prozessmodelle ermöglichen die Vereinfachung von komplexen Prozessen und erlauben eine effizientere Umsetzung von diesen Prozessabläufen. Deswegen wird ein gutes Verständnis des Modellierungsvorgangs und das Wissen der Kriterien zur Erstellung eines qualitativ hochwertigen Prozessmodells benötigt. Hierfür müssen empirische Daten ermittelt werden mit dem Ziel ein besseres Verständnis für die kognitiven und qualitativen Vorgänge bei der Betrachtung und Erstellung von Prozessmodellen zu entwickeln. Eine Möglichkeit zur Ermittlung dieser benötigten Daten ist die Befragung im Rahmen von Studien. Die in dieser Arbeit konzipierte Web-Plattform *ProMoEE* ermöglicht die Erstellung, Bearbeitung und Evaluation von Fragebögen im Kontext von Business Process Management (BPM). Diese Fragebögen sind über das Internet zugänglich und besitzen neben Fragen auch die Möglichkeit, Prozessmodelle anhand von Aufgabenstellungen im Fragebogen zu erstellen und zu bewerten. Die Plattform soll die Durchführung und Evaluation von empirischen Studien zum besseren Verständnis von Prozessmodellen unterstützen.

# Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Masterarbeit und allgemein bei meinem Studium unterstützt haben.

Zunächst möchte ich mich bei *Professor Dr. Manfred Reichert* für seine großartige Arbeit und Wirken im Institut Datenbank und Informationssysteme und an der Universität Ulm bedanken. Des Weiteren bedanke ich mich bei allen Mitarbeitern des Instituts Datenbank und Informationssysteme für die tolle Zusammenarbeit in verschiedenen Projekten.

Besonderen Dank an meinen Betreuer *Michael Winter*, der mich während meines Studiums in verschiedenen Projekten begleitet hat und mir immer mit Rat und Tat beiseite stand. Danke für die großartige Betreuung meiner Bachelor- und Masterarbeit und für die Unterstützung im Laufe meines Studiums.

Abschließend möchte ich mich bei meinen *Eltern* für die finanzielle und zwischenmenschliche Unterstützung während meines Studiums bedanken. Danke an meinem Bruder *Martin* und meiner Mutter *Petra* für das Korrekturlesen meiner Arbeiten.

# Inhaltsverzeichnis

# 1

# Einleitung

Prozesse sind ein fester Bestandteil des alltäglichen Lebens. Sie beschreiben, wie ein Vorgang abläuft und erledigt wird [1]. In bestimmten Situationen ist es erstrebenswert, Prozesse in Modellen zu abstrahieren. Diese Prozessmodelle bieten eine Übersicht über Prozessvorgänge. Die Disziplin der Verwaltung von Prozessen und die Erstellung von Prozessmodellen wird als *Business Process Management (BPM)* bezeichnet. Das Ziel von BPM ist die Analyse, Optimierung und Automatisierung von Prozessen. Zur Erreichung dieser Ziele werden einheitliche Normen beim Modellieren von Prozessen benötigt. Ein führender Modellierungsstandard ist Businness Process Model and Notation (BPMN 2.0)[1].

Um die Verständlichkeit von Prozessmodellen weiter zu steigern und resultierende Prozessmodelle zu verbessern werden empirische Studien zur Modellierung von Prozessen durchgeführt. Diese Studien untersuchen den Modellierungsvorgang zur Erstellung von Prozessmodellen und die Qualität daraus resultierender Prozessmodelle. Auch die Einführung von Neuanwendern in BPMN 2.0 soll verbessert werden. Bei der Durchführung von diesen empirischen Studien müssen Daten systematisch erhoben, ausgewertet und analysiert werden. Eine wesentliche Methode zur Datenerhebung in der Empirie stellt die Befragung dar [2]. Es wird zwischen mündlicher und schriftlicher Befragung unterschieden. Neben Interviews und Fragebogen in Papierform, ist es möglich, die Datensammlung mithilfe technologischer Hilfsmittel zu unterstützen. Mithilfe des Computers sind Fragebögen effizient und schnell auswertbar, wodurch komplexere empirische Studien über größere Testmengen möglich sind. Heutzutage sind mit Befragungen über das Internet beliebig große Personengruppen erreichbar und es ist möglich, die Daten automatisch auszuwerten und zu speichern [3].

Zur Durchführung von empirischen Studien zur Prozessmodellierung wurde in dieser Masterarbeit eine über das Internet zugängliche Plattform zur Durchführung von empirischen Studien in Form von Fragebögen entwickelt. Diese Plattform die Durchführung empirischer Studien zur Prozessmodellierung unterstützen. Hierfür muss es möglich sein, für Benutzer Prozessmodelle in Studien zu erstellen und für Studienleiter auszuwerten. Die Plattform sollte über das Internet für jeden verfügbar sein, um eine möglichst große Personengruppe auf einfache Weise erreichen zu können.

## 1.1 Zielsetzung

Ziel dieser Arbeit ist die Konzeption und Entwicklung einer Web-Plattform für die Definition, Durchführung und Auswertung von empirischen Studien im Kontext von Business Process Management. Für diese Studien soll es möglich sein, Fragebögen zu erstellen, welche mithilfe eines Zugangscodes für nicht registrierte Benutzer zugänglich sind. Das Besondere an diesen Fragebögen ist, dass sie die Möglichkeit bieten, Geschäftsprozesse in der BPMN 2.0 Norm zu definieren. Auch das Auswerten von vorgegebenen Prozessmodellen im Fragebogen soll möglich sein. Die auf diese Weise gesammelten Daten werden im System gespeichert und für registrierte Benutzer auswertbar sein. Diese Plattform soll unabhängig vom Endgerät auf jeden Web-Browser zugänglich sein.

## 1.2 Struktur der Arbeit

Diese Arbeit ist in acht Kapitel unterteilt. In diesem Kapitel wurde eine kurze Einführung und Motivation zur Arbeit gegeben. In Kapitel 2 werden ähnliche Arbeiten und Systeme beleuchtet. Anschließend werden wichtige Grundlagen zu dieser Arbeit in Kapitel 3 vermittelt. Eine genaue Aufgabenstellung und Anforderungen an das System werden in Kapitel 4 vorgestellt. In Kapitel 5 wird die benötigte Architektur des Systems ermittelt. Es werden die verwendeten Frameworks begründet und der Aufbau des Systems beleuchtet. In Kapitel 6 wird das daraus implementierte System vorgestellt. Anhand der Implementierung wird in Kapitel 7 das System mit den Anforderungen aus Kapitel 4

abgeglichen. Abschließend folgt mit Kapitel 8 eine Zusammenfassung dieser Arbeit und es werden mögliche Erweiterungen zum System in weiterführenden Arbeiten vorgestellt. Einzelne Quellcodeausschnitte werden im Anhang A vorgestellt.

# 2

# Verwandte Arbeiten

Zunächst werden in diesem Kapitel die zu dieser Arbeit verwandten und relevanten Arbeiten und Systeme vorgestellt. Zum Kapitelende wird eine Studie vorgestellt, die einen möglichen Anwendungsfall für diese Arbeit repräsentiert.

### 2.0.1 PEx

*PEx* [4] stellt das Vorgängersystem zu dieser Arbeit dar. Der Hauptfokus von *PEx* liegt auf das Durchführen von Experimenten. Es ist möglich in unterschiedlichen Schritten Fragen zu beantworten und Modelle zu erstellen. Zudem hat der Anwender am Ende eines Experiments die Möglichkeit Feedback zu geben. Auch dieses System wurde als Web-Anwendung realisiert. Über Schlüssel werden Experimente mit anderen Benutzern geteilt. Analog zu dem in dieser Arbeit entwickeltem System ist es in PEx möglich, über einen entsprechenden Code über das Internet ein Experiment durchzuführen. Der Anwender muss dazu Fragen beantworten und Modelle erstellen. Diese Modelle sind auswertbar. Ein spezieller Experiment-Editor erlaubt das Erstellen von alterierenden Experimenten, und ermöglicht damit das Ausführen unterschiedlicher Aufgaben innerhalb des Experiments. Dies ist interessant für das automatische Durchführen von Experimenten mit Test- und Kontrollgruppen. Abbildung 2.1 zeigt den Experiment Editor von *PEx*.

Dieses System wird in dieser Arbeit mit Fokus auf das Durchführen von Umfragen und Fragebögen neu konzipiert. Besonders nicht-funktionale Anforderungen wie Performance und Stabilität liegen in dieser Arbeit im Fokus.

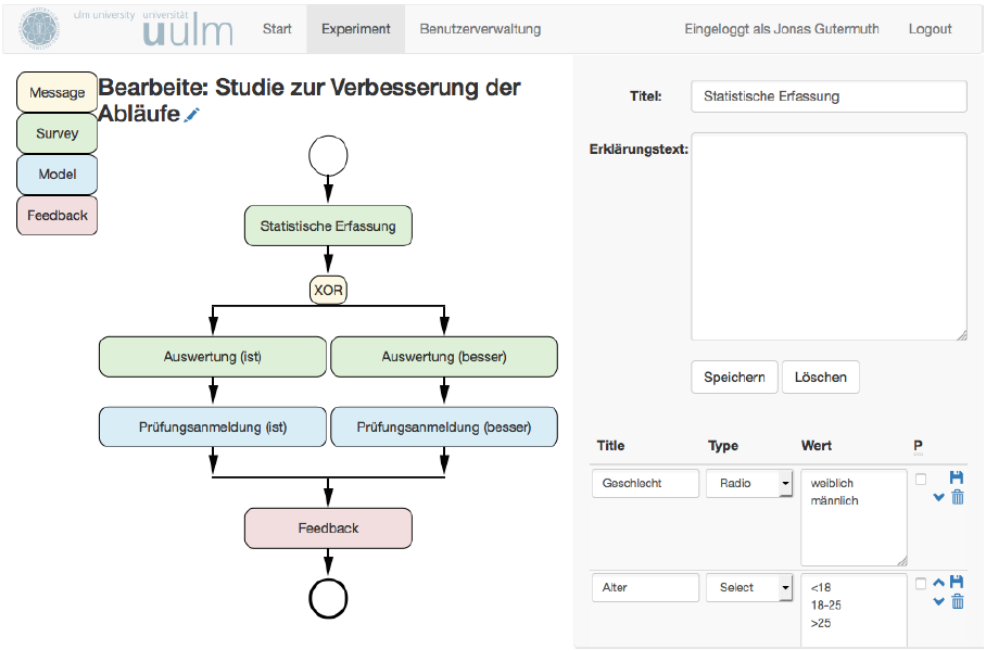



Abbildung 2.1: Experiment-Editor von *PEx* [4]

### 2.0.2 Cheetah Experimental Plaform

Das eben beschriebene System *PEx* hatte als Vorlage das System *Cheetah Experimental Platform* [5]. Cheetah wurde an der Universität Innsbruck entwickelt und ist ein Werkzeug für das Durchführen von Experimenten im Kontext der Erstellung von Geschäftsprozessen. Cheetah ermöglicht in einer lokalen Anwendung das Erstellen und Durchführen von Experimenten. Über die Eingabe eines Codes wird das entsprechende Experiment geladen. Dieses Experiment besteht zunächst aus einer demografischen Umfrage. Anschließend wird über ein Tutorial in die Modellierung von Geschäftsprozessen eingeführt, um darauffolgend Modellieraufgaben durchzuführen und Modelle zu erstellen. Es ist in dieser Anwendung ebenfalls möglich unterschiedliche Aufgaben unterschiedlichen Gruppen im selben Experiment zuzuweisen und dadurch Test- und Kontrollgruppen im selben Experiment einzubinden. In einer abschließenden Befragung werden Fragen über die kognitiven Anforderungen der Modellieraufgaben gestellt. In Abbildung 2.2 wird dieser allgemeine Worklow von Cheetah visualisiert.

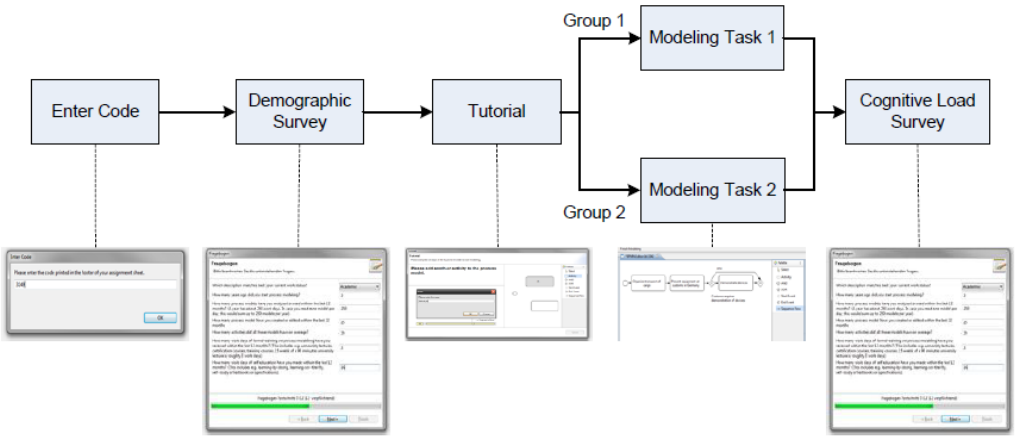



Abbildung 2.2: Cheetah Experimental Workflow [5]

Diese Anwendung diente dem Institut für Datenbanken und Informationssystemen als eine Experimentierumgebung für das Durchführen von Experimenten zu kognitiven Vorgängen bei der Modellierung von Geschäftsprozessen. Da diese Anwendung nur lokal verfügbar ist, ist das Durchführen auf eine kleinere Gruppe eingeschränkt und mit höheren Aufwand für die Testkandidaten verbunden. Deswegen wurde mit *PEx* und mit dem in dieser Arbeit entwickelten System *ProMoEE* eine Web-Anwendung entwickelt, um das Sammeln der empirischen Daten zu vereinfachen und die Anwendung einer größeren Gruppe leichter zugänglich zu machen.

### 2.0.3 QuestionSys

*QuestionSys* wird an der Universität Ulm im Institut Datenbanken und Informationssysteme entwickelt [6, 7]. Das Ziel von *QuestionSys* ist es, die Erstellung und Evaluation von Fragebögen zu vereinfachen. Die Fragebögen werden in diesem System als Prozessmodelle aufgebaut. Durch diesen modellbasierten Ansatz sind Beziehungen und Abhängigkeiten von Fragen untereinander grafisch abbildbar. Über eine Prozessengine ist dieses Prozessmodell auf einen mobilen Endgerät als ein Fragebogen darstellbar. Die gesammelten Daten werden im System gespeichert und es ist für Anwender möglich,

Daten effizient zu sammeln und auszuwerten. Das Erstellen der Fragebögen als Prozessmodell ermöglicht einen vereinfachten Überblick über komplexere Fragebögen mit dem Ziel, die Erstellung von Fragebögen zu vereinfachen und die Qualität resultierender Fragebögen zu verbessern.

### 2.0.4 Activity labeling in process modeling: Empirical insights and recommendations

In diesen Abschnitt wird eine Studie als Beispiel vorgestellt, dessen Durchführung das in dieser Arbeit entwickelte System unterstützen würde. Die vorgestellte Studie trägt den Titel “Activity labeling in process modeling: Empirical insights and recommendations“ [8] und wurde 2010 veröffentlicht. Die Studie befasst sich mit der Beschriftung für die verschiedenen grafischen Elemente in Prozessmodellen. Für diesen Zweck wurden unterschiedliche Beschriftungsarten miteinander empirisch abgeglichen. In dieser Studie wurden über 600 Prozessmodelle abgeglichen und die Wahrnehmung der Benutzer bei der Betrachtung dieser Modelle analysiert. An der Studie haben 29 Studenten teilgenommen. In einem Fragebogen wurden zunächst demografische Fragen gestellt. Im Anschluss wurden verschiedenen Prozessmodelle vorgestellt mit unterschiedlichen Beschriftungen. Die Studenten haben diese Beschriftung nach Eindeutigkeit und Verständlichkeit bewertet.

Die Studie hat als Ergebnis, dass Beschriftung für Aktivitäten in Verb-Objekt-Form als besonders eindeutig und verständlich wahrgenommen werden [8]. Eindeutigkeit und Verständlichkeit werden als wichtige Qualitäten für die Beschriftung von Elementen in der Studie festgelegt. Ein weiteres Ergebnis der Studie ist die Erkenntnis, dass sowohl Expertise in der entsprechenden Anwendung des Prozessmodels sowie in der Modellierung allgemein zu keiner qualitativ besseren Beschriftung führen.

Die betrachteten Prozessmodelle wurden in EPC-Notation modelliert, eine Durchführung dieser Studie analog in BPMN 2.0 ist denkbar.

# 3

# Grundlagen

Das in dieser Arbeit entwickelte System *ProMoEE* beruht auf einen Webservice der es ermöglicht, Online-Fragebögen auszufüllen und auszuwerten. Zusätzlich soll es möglich sein, Prozessmodelle nach BPMN 2.0 zu erstellen und anzuzeigen. In diesen Kapitel werden die primären Grundlagen von dieser Arbeit beleuchtet. Es wird auf die Theorie hinter Online-Fragebögen eingegangen und die Spezifikation von BPMN 2.0 erklärt.

## 3.1 Online-Befragung

*Befragungen* sind eine wesentliche Methode zur Durchführung empirischer Studien. Sie ermöglichen Einblicke in verschiedene soziale und kognitive Erfahrungen von Personengruppen [2]. Die Empirie beschäftigt sich mit dem methodischen Sammeln und Auswerten von Daten. Die Befragung ist neben der Beobachtung und dem Experiment eine der wichtigsten und am häufigsten verwendete Methode zur Datenerhebung [9].

Bei der Befragung ist besonders die Wahl geeigneter Fragen entscheidend. Eine Frage muss verständlich und im Sinne der Studie gestellt werden [10]. Auch die Art und Durchführung der Befragung und die Umgebung ist entscheidend. Bei der Befragung wird zwischen mündlicher (Interview) und schriftlicher Befragung unterschieden. Ausprägungen davon sind besonders durch technische Hilfsmittel entstanden, beispielsweise das Telefoninterview und die Online-Befragung. Mithilfe technischer Hilfsmittel lässt sich schneller eine größere Personengruppe erreichen und ermöglicht somit quantitativere Datenmengen [2]. Die Onlinebefragung hat folgende Vor- und Nachteile (vgl. [2, 3]).

**Vorteile:**

- Einbindung zusätzlicher multimedialer Präsentationsmöglichkeiten und Werkzeuge.
- Feedback der automatisch ausgewerteten Ergebnisse an den Befragten.
- Selektives und automatische Anpassung des Fragebogens anhand bereits getätigter Antworten.
- Ermitteln von zusätzlichen Daten, die bei der Befragung angefallen sind, zum Beispiel die Dauer der Befragung.
- Automatische Fehlererkennung durch interne Validierungen.
- Ermöglichung eines anonymen Feedbacks des Benutzers zum Fragebogen.
- Geringe Kosten durch automatische Auswertung. Dateneingaben entfallen.
- Höhere Offenheit des Befragten durch gesteigerte wahrgenommene Anonymität und vertrauter Umgebung.
- Leichtere Erreichbarkeit nahezu unbegrenzt großer Zielgruppen.
- Geringerer Fremdeinfluss.

**Nachteile:**

- Ungeklärte Repräsentativität von Online-Befragungen. Die meisten Internetbenutzer gehören zur jüngeren Bevölkerungsgruppen. Laut *statista* benutzen fast 100% der bis zu 40-jährigen das Internet, jedoch nur 50% der über 60-jährigen Bevölkerung [11]. Auch Bots und automatische Programme können das Ergebnis verfälschen. Dadurch kommt es vor, das bestimmte Bevölkerungsgruppen über- oder unterrepräsentiert sind.
- Ungeklärte Verbindlichkeit der Befragung. Hohe Offenheit birgt das Risiko verzerrterer Antworten, erhöht durch eine andere Identitätswahrnehmung im anonymen Internet.
- Letztendlich keine Kontrolle oder Aussagemöglichkeit über die Grundgesamtheit der Befragten.

Daraus wird ersichtlich, dass Online-Befragungen zwar die Quantität der erhobenen Daten steigern, nicht jedoch die Qualität. Nichtsdestotrotz sind Online-Befragungen aufgrund der geringen Kosten, leichtere Datenauswertung und höhere Erreichbarkeit eine beliebte Methode. So ist die Online-Befragung beispielsweise laut dem *Arbeitskreis deutscher Markt- und Sozialforschungsinstitute (ADM)* die meistverwendete Befragungsart der letzten Jahre gewesen [12] (siehe Abbildung 3.1).

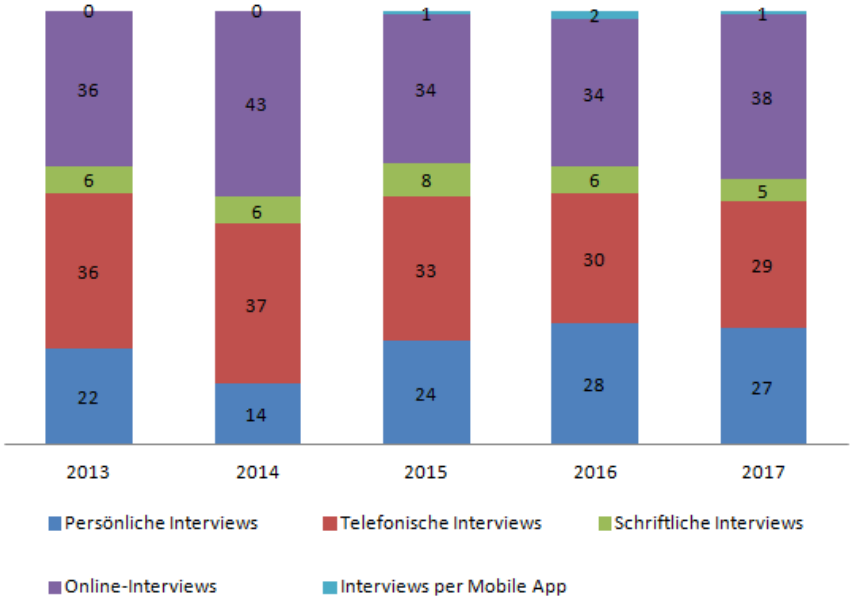



Abbildung 3.1: Verteilung verwendeter Befragungsarten im Jahr 2013 bis 2017 in % [12]

Bei diesen Online-Umfragen wird in der Regel ein Fragebogen von dem Benutzer ausgefüllt. Mittlerweile gibt es eine Menge im Internet verfügbarer Werkzeuge zum Erstellen und Verbreiten von Online-Fragebögen wie *SurveyMonkey* [13], *Survio* [14], *Doodle* [15], *LimeSurvey* [16] und viele mehr.

Ein Vorteil von Online-Fragebögen ist die Möglichkeit, zusätzliche Werkzeuge mit einzubinden. Es ist beispielsweise möglich, den Benutzer Prozessmodelle nach der *BPMN 2.0* Spezifikation (*Business Process Model and Notation*) im Fragebogen erstellen zu lassen. Im Kontext dieser Arbeit soll es möglich sein, im Fragebogen anhand von Aufga-

benstellungen Geschäftsprozessmodelle zu erstellen, welche wiederum vom Benutzer evaluierbar sind. Es sollen Daten für empirische Studien gesammelt werden, um mithilfe dieser Daten den Prozess der Modellierung zu verbessern und resultierende Prozessmodelle verständlicher zu gestalten. Im folgenden Abschnitt werden die Grundlagen für BPMN 2.0 beleuchtet.

## 3.2 BPMN 2.0

*BPMN 2.0* bezeichnet einen international anerkannten Standard zur Modellierung von Geschäftsprozessen. Es wird benutzt, um Geschäftsprozesse einheitlich zu modellieren, wodurch die Einarbeitung in diese Geschäftsprozessen verkürzt und die Verständlichkeit gesteigert wird. Diese Prozessmodelle werden zudem in Informationssystemen verwendet, um diesen Prozess auszuführen und zu unterstützen. *BPMN 2.0* ist ein führender Standard zur Modellierung und Ausführung von Geschäftsprozessen [17].

*BPMN* dient der Beschreibung des Aufbaus und Ablaufs von Prozessen. In einfachen Modellen werden zwischen den Start und dem Ende eines Ablaufes *Aktivitäten* (Tasks) in Reihenfolge aufgereiht und beschrieben. Diese Aktivitäten sind mit *Kontrollflüssen* (Sequence-Flows) verbunden. Zwischen diesen Aktivitäten sind *Abzweigungen* (Gateways) aufbaubar. Diese Abzweigungen verlaufen entweder exklusiv, inklusiv oder parallel. Komplexere Modelle bestehen meist aus *Pools*, die eine Organisationseinheit beschreiben, und *Lanes*, die einzelne Personen(-gruppen) darstellen. Die Lanes grenzen die jeweiligen Aufgaben der einzelnen Personen ab, während Pools den gesamten Geschäftsprozess innerhalb der Organisationseinheit abgrenzen. Zwischen Pools wird mit *Nachrichtenflüssen* (Message-Flows) kommuniziert. Nachrichtenflüsse stellen die Kommunikation und Interaktion zwischen den Systemen dar. Über diese Nachrichtenflüsse ist es möglich sogenannte *Ereignisse* (*Events*) auszulösen. Ein Prozessmodell startet immer mit einem Startereignis und endet mit Endereignissen. Weitere Annotationsmöglichkeiten sind Kommentare, Dateien und Organisationshardware wie zum Beispiel Server oder Datenbanken. In Abbildung 3.2 wird ein Prozessmodell nach *BPMN 2.0* mit den wichtigsten Elementen vorgestellt.

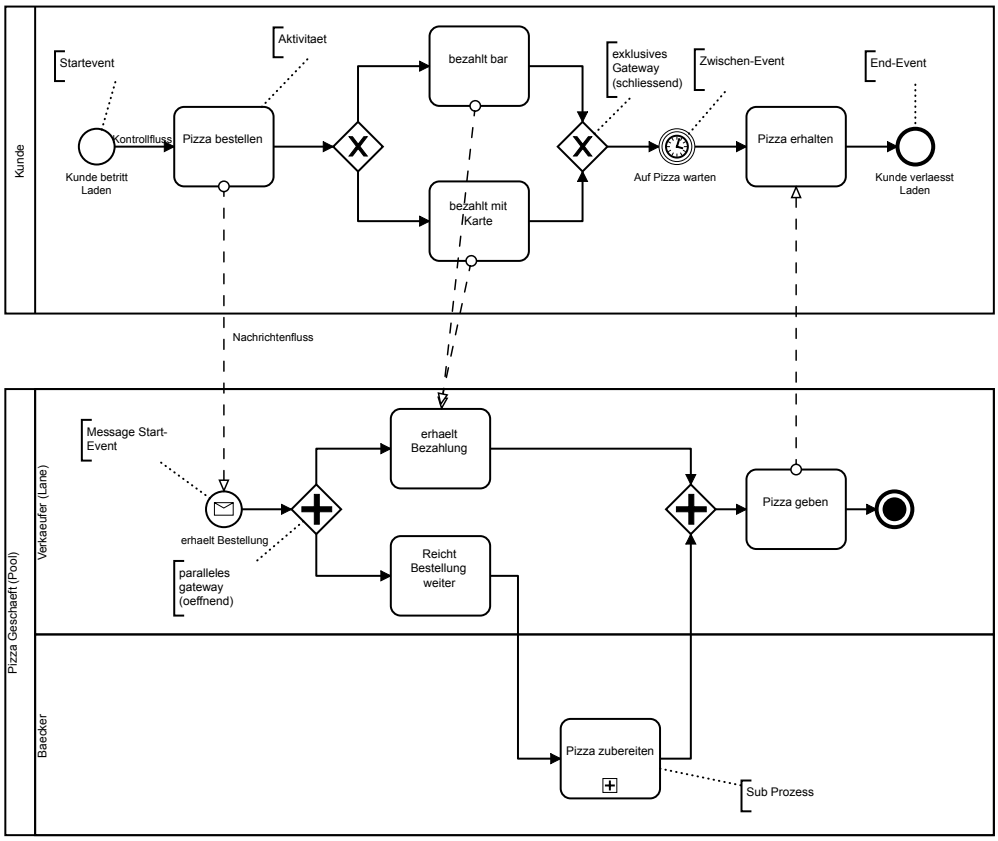



Abbildung 3.2: Beispiel für ein *BPMN 2.0* Modell: Pizza bestellen

Dieses Prozessmodell beschreibt beispielhaft den Ablauf zur Bestellung einer Pizza in einem Pizzageschäft. Die Organisationseinheit Pizzageschäft ist als Pool von dem Kunden getrennt. Zwischen den Kunden und dem Geschäft wird mit Nachrichtenflüssen kommuniziert. Der Verkäufer und der Bäcker sind Personen in der Organisationseinheit.

# 4

# Anforderungsanalyse

In diesem Kapitel wird die Aufgabenstellung betrachtet. Es wird auf die verschiedenen Anforderungen eingegangen, die dieses System erfüllen muss. In Abschnitt 4.1 wird die Aufgabenstellung im Detail beleuchtet. Daraus werden im Abschnitt 4.2 die funktionalen Anforderungen abgeleitet. Im Abschnitt 4.3 werden abschließend die nicht-funktionalen Anforderungen erläutert.

## 4.1 Aufgabenstellung

Die Hauptaufgabe des Systems ist das Durchführen von Studien bezüglich der Prozessmodellierung auf einer Webseite. Hierfür werden primär Fragebögen erstellt und ausgefüllt. Es soll möglich sein, BPMN-Modelle anhand von Fragestellungen zu erstellen und anhand von vorgegebenen Prozessmodellen Fragen zu beantworten. Der jeweilige Fragebogen soll mithilfe eines Codes von nicht registrierten Benutzern angefordert und ausgefüllt werden. Abschließend soll es für den Benutzer möglich sein den ausgefüllten Fragebogen zu einem Server abzusenden.
Damit wird ein Web-Client benötigt, welche die Benutzeroberfläche für den Anwender bereitstellt, sowie ein Server, der HTTP-Anfragen annimmt und verarbeitet. Der Server muss zusätzlich mit einer Datenbank verbunden sein, auf der Fragebögen und Antworten abgespeichert werden.
Des Weiteren muss eine Möglichkeit bestehen, Fragebögen für angemeldete Benutzer zu verwalten. Dies umfasst das Anlegen von neuen Fragebögen, das Bearbeiten von existierenden Fragebögen, sowie das Löschen von Fragebögen. Angemeldete Benutzer sollen nur in der Lage sein auf ihre eigenen Fragebögen zuzugreifen. Eine weitere

wichtige Funktion ist das Evaluieren der Antworten zu einem Fragebogen. Dies umfasst die Möglichkeit, alle Antworten zu einer Fragestellung zu erhalten, sowie die Antworten eines ausgefüllten Fragebogens einzusehen.
Es soll die Möglichkeit bestehen, als Administrator neue Benutzer anzulegen und zu verwalten.

## 4.2 Funktionale Anforderungen

Aus der Aufgabenstellung von Abschnitt 4.1 lassen sich folgende funktionale Anforderungen ableiten.

**FA-01 Code absenden**

Ein nicht-registrierter Benutzer soll die Möglichkeit besitzen, einen Code zum Server zu senden. Der Server schickt anschließend einen zugehörigen Fragebogen zum Web-Client zurück, der auf diesen dann angezeigt wird.

**FA-02 Fragebogen ausfüllen**

Ein Fragebogen soll auf der Webseite korrekt angezeigt werden und ausfüllbar sein. Die gesetzten Antworten sollen im Browser-Cache mit abgespeichert werden. Es soll möglich sein, anhand von Fragestellungen BPMN-Modelle zu erstellen. Außerdem soll die Möglichkeit bestehen, anhand eines vorgegebenen Modells zugehörige Fragen zu beantworten.

**FA-03 Fragebogen absenden**

Sobald der Anwender den Fragebogen zu seiner Zufriedenheit ausgefüllt hat, soll er die Möglichkeit besitzen, die Antworten und erstellten Prozessmodelle zum Server zu schicken. Diese getätigten Antworten sollen anschließend auf einer Datenbank abgespeichert werden.

**FA-04 Benutzer Login**

Es soll für Studienleiter und Mitarbeiter möglich sein, sich auf den System einzuloggen. Dafür füllt der Mitarbeiter ein HTML-Formular mit seinen Anmeldedaten (Benutzername und Passwort) aus.

**FA-05 Fragebögen anzeigen**

Der eingeloggte Benutzer soll die Möglichkeit besitzen die von ihm erstellten Fragebögen einzusehen.

**FA-06 Fragebogen erstellen**

Ein eingeloggter Benutzer ist in der Lage einen neuen Fragebogen zu erstellen und diesem einem Code zuzuweisen.

**FA-07 Fragebogen bearbeiten**

Ein eingeloggter Benutzer soll in der Lage sein einen bereits vorhandenen Fragebogen, auf welchen er Zugriff besitzt, zu bearbeiten. Eine Möglichkeit Fragebögen zu kopieren ist ebenfalls denkbar.

**FA-08 Fragebogen löschen**

Das Löschen von Fragebögen soll für berechtigte Benutzer möglich sein.

**FA-09 Ausgefüllte Fragebögen anzeigen**

Ein Benutzer muss eine Übersicht auf alle ausgefüllten Fragebögen eines Fragebogens haben, auf welchen er Zugriff besitzt. Der Benutzer hat zudem Einblick auf die getätigten Antworten und erstellten Prozessmodelle eines ausgefüllten Fragebogens.

**FA-10 Fragebogen evaluieren**

Der Benutzer soll in der Lage sein, einzelne Aufgabenstellungen zu evaluieren. Dabei erhält er Einblick über alle getätigten Antworten zu einer Frage, beziehungsweise alle erstellten Prozessmodelle zu einer Modellierungsaufgabe.

**FA-11 Benutzer verwalten**

Der Administrator besitzt Zugriff auf alle Fragebögen und verwaltet alle Benutzer. Zur Benutzerverwaltung gehört das Sperren und Entsperren von Benutzern.

**FA-12 Benutzer anlegen**

Der Administrator erstellt mithilfe eines HTML-Formulars neue Benutzer.

### 4.2.1 Zusätzliche Anforderungen

Neben den eben genannten Anforderungen existieren weitere optionale funktionale Anforderungen (OFA), welche das Projekt weiter verbessern.

**OFA-01 Fragebogendauer**

Es ist nützlich zu wissen, wie lange ein Benutzer für einen Fragebogen benötigt hat. Auch das Datum das angibt, wann der Fragebogen ausgefüllt wurde, soll mit gespeichert werden.

**OFA-02 Feedback**

Im Optimalfall soll ein Benutzer, der gerade einen Fragebögen ausgefüllt hat auch die Möglichkeit besitzen, Feedback zum Fragebogen zu geben.

**OFA-03 Modelldaten**

Es soll anhand eines erstellten Modells verschiedene Daten zum Modell ermittelt werden, zum Beispiel wie viele Elemente verwendet werden. Zudem sind Angaben über die Komplexität und die Validität eines Modells denkbar.

**OFA-04 Modell Replay**

Die Möglichkeit den Erstellvorgang eines Fragebogens mithilfe einer Replayfunktion Schritt für Schritt zu verfolgen würde ein tieferes Verständnis über das Modell ermöglichen.

**OFA-05 Excel Export**

Die verschiedenen Antworten zu einem Fragebogen sollen in ein Excel-Dokument exportierbar sein.

## 4.3 Nicht-Funktionale Anforderungen

Es ist zudem wichtig, dass das System die in diesem Abschnitt vorgestellten nicht-funktionalen Anforderungen erfüllt. Nicht-funktionale Anforderungen beschreiben verschiedene Aspekte der Leistung und Qualität eines Systems.

**NFA-01 Verfügbarkeit**

Der Webservice muss über das Internet für jeden jederzeit erreichbar sein. Zusätzlich muss die Webseite auf allen gängigen Browsern und auf Mobilgeräten benutzbar sein.

**NFA-02 Zuverlässigkeit**

Das System ist in der Lage, dauerhaft seine Funktionen zu erfüllen.

**NFA-03 Intuitive Bedienbarkeit**

Die Bedienung der Webseite muss intuitiv sein, da besonders viele nicht registrierte Benutzer die Webseite verwenden. Aber auch der Mitarbeiterbereich muss navigierbar und übersichtlich sein.

**NFA-04 Einheitliches Aussehen**

Die Webseite soll möglichst schlicht und einheitlich gehalten werden, damit der Benutzer nicht unnötig abgelenkt wird. Außerdem muss sie dem Design der Webseite der Universität Ulm ähneln.

**NFA-05 Vorhersehbares Verhalten**

Das Verhalten des Systems soll vorhersehbar sein. Dies wird beispielsweise durch eine eingehende Beschreibung der Funktionen erreicht.

**NFA-06 Leistungsfähig**

Das System muss in der Lage sein, eine Vielzahl von Sessions gleichzeitig und ohne Leistungseinbußen zu verarbeiten. Die Antwortzeiten sollen so gering wie möglich sein.

**NFA-07 Sicherheit**

Das System muss sicher vor Angriffen von außen sein. Deswegen ist eine Verwendung von Verschlüsselungen und Authentifizierungen sinnvoll.

**NFA-08 Datenschutz**

Nur befugte Benutzer dürfen Daten abrufen und verändern. Zudem gelten die allgemeinen Datenschutzgesetze.

**NFA-09 Datenintegrität**

Daten dürfen nicht verloren gehen. Erhaltene Daten müssen vom System auf Validität überprüft werden.

**NFA-10 Skalierbarkeit**

Das System muss lange im Betrieb sein, ohne zu viele Ressourcen zu benötigen. Auch große Datenmengen müssen problemlos verarbeitbar sein.

**NFA-11 Stabilität**

Das System muss über einen langen Zeitraum fehlerfrei laufen. Es darf zu keinen internen Fehlerzuständen oder Inkonsistenzen kommen.

**NFA-12 Robustheit**

Das System muss sich gegenüber fehlerhaften Eingaben robust verhalten und den Benutzer auf Fehler auf einer transparenten Art hinweisen.

**NFA-13 Erweiterbarkeit**

Das System muss erweiterbar sein. Zusätzliche Funktionen sind ohne großen Mehraufwand implementierbar.

**NFA-14 Portierbarkeit**

Das Installieren und Portieren des Servers muss einfach und schnell sein.

# 5

# Architektur

In diesem Kapitel wird der allgemeine Aufbau und die Architektur von *ProMoEE* beleuchtet. Das System ist nach dem Client-Server Prinzip aufgebaut, das bedeutet, dass unterschiedlich viele Benutzer über einen Client (in diesem Fall der Browser) auf den Server gleichzeitig zugreifen. Im Client wird die Benutzeroberfläche angezeigt und kleinere Logik ausgeführt. Der Browser generiert die Oberfläche über *HTML/CSS* und es sind mit *JavaScript* interne Berechnungen durchführbar. Der Client sendet Anfragen zum Server über die *RESTful API*. Der Server verarbeitet diese Anfragen und speichert Daten in eine Datenbank. Da es sich hierbei um zwei Komponenten handelt, die miteinander kommunizieren, wird im folgenden auch von Front-End (Client) und Back-End (Server) gesprochen. In Abbildung 5.1 ist das Client-Server Prinzip veranschaulicht.

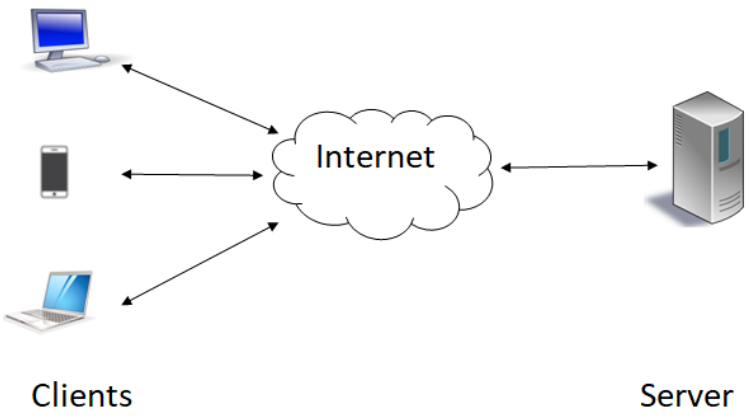



Abbildung 5.1: Client-Server-Aufbau (vereinfacht)

## 5.1 Programmiersprachen und Umgebungen

Die erste Konzeptentscheidung war die Wahl der serverseitigen Programmiersprache. PHP wird in den meisten Fällen für Server verwendet. Aber auch andere Programmiersprachen bieten mit modernen Frameworks eine gute Alternative zu *PHP* an. Sprachen wie *Java*, *Node.js* (serverseitiges Javascript), *C#* und *ASP.NET* bieten die Möglichkeit an, performante Server zu entwickeln. Im Hinblick auf meine Vorerfahrung und Programmierkenntnisse habe ich mich für Java als serverseitige Programmiersprache entschieden.

Java bietet die Möglichkeit, den Server auf eine portierbare Datei zu reduzieren. Eine weitere Anforderung an das System ist eine einfache Erweiterbarkeit. Da ein Großteil der Studenten an der Universität Ulm mit Java vertraut sind, ist dies ebenfalls vorteilhaft. Weitere Vorteile von Java sind Objekt-Orientierung, Bekanntheit, Dokumentation, Multi-Threading, Portierbarkeit, automatische Speicherverwaltung und Skalierbarkeit. Als Entwicklungsumgebung bietet sich das Open-Source Programm Eclipse an.

Für das Front-End wird zwangsläufig *HTML/CSS/JavaScript* vorgegeben, da die Anwendung auf jeden Browser laufen soll. Als unterstützende Entwicklungsumgebung wurde Visual Code verwendet.

## 5.2 Frameworks

Um die Entwicklung zu vereinfachen und das System performant zu halten, werden verschiedene moderne Frameworks verwendet. In diesen Abschnitt werden zunächst die verwendeten Frameworks im Back-End beleuchtet. Ab den Unterabschnitt 5.2.8 *React* werden die Frameworks im Front-End vorgestellt.

### 5.2.1 Dropwizard

*Dropwizard* [18] ist eine Kollektion von unterschiedlichen modernen und stabilen Frameworks zur Erstellung von *RESTful Web Services* in einer einzigen Bibliothek. Diese

Frameworks werden über Dropwizard intern passend zusammengebaut, wodurch eine sehr schnelle Erstellung eines funktionierenden Web Servers ermöglicht wird. Dropwizard ist sehr leicht konfigurierbar, bietet sehr viele dokumentierte Erweiterungen für verschiedene Zwecke an und ist sehr performant. Die Dropwizard Bibliothek umfasst unter anderen das Framework *Jetty* zum Aufbau eines HTTP-Servers, *Jersey* für das Mappen von REST-Anfragen auf Funktionen, *Jackson* für das De-/Serialisieren von *JSON* Datenformaten (JSON wird als Datenformat in Anfragen und Antworten verwendet), *JDBI* zum Zugriff auf Datenbanken und *Liquibase* zum Aufbauen und Migrieren von Datenbankschemas.

### 5.2.2 Maven

*Maven* [19] ist ein Werkzeug um Java Bibliotheken und Frameworks zu verwalten und unterstützt beim Kompilieren, Aufbauen, Testen und Verpacken von Java-Projekten. Es ermöglicht zudem das automatische Einbinden von Bibliotheken und das Erstellen des Servers in einer einzigen .jar Datei für leichte Portierbarkeit. Dropwizard ist eng an Maven ausgerichtet und empfiehlt die Verwendung von Maven. Maven ermöglicht zudem *"annotation processing"*, was für das Framework *Map Struct* in Abschnitt 5.2.5 benötigt wird.

### 5.2.3 Guice

*Guice* [20] ist ein von Google entwickeltes Framework zur Unterstützung des Entwicklers beim Entwickeln von Java-Software. Ein Problem von Java ist das Verbinden von Abhängigkeiten der Klassen untereinander. Dies geschieht häufig beim Konstruieren oder Bauen einer Klasse. Das Verbinden von Abhängigkeiten ist fehleranfällig und stellt für den Entwickler eine mühsame Aufgabe dar. Über *Dependency Injection* ist es möglich, Klassen- und Instanzabhängigkeiten direkt in eine Klasse sozusagen "einzuspritzen", wodurch die Lesbarkeit verbessert wird, potentielle Fehlerquellen vermieden und die Entwicklung vereinfacht und beschleunigt wird. Guice bietet hierfür ein einfaches und leichtgewichtiges Framework an. In Listing 5.1 ist ein Beispiel gegeben für das

automatische Einfügen von benötigten Klassenabhängigkeiten im Konstruktor. Diese Abhängigkeiten werden automatisch im Hintergrund beim Erstellen der Klasse von Guice hinzugefügt.

```
1 public class CodeResource {
2
3   private final CodeService codeService;
4
5   @Inject
6   public CodeResource (CodeService codeservice) {
7     this.codeService = codeservice;
8   }
```

Listing 5.1: Automatisches Einfügen der Abhängigkeit von der Klasse *CodeService* in die Klasse *CodeResource*

### 5.2.4 EclipseLink JPA

*EclipseLink* bietet über eine *Java Persistence API (JPA)* eine Schnittstelle zwischen Java Objekten und der Datenbank an [21]. Daten, die auf den Server als Java Objekte hinterlegt sind, werden zu Einträgen in der Datenbank transformiert und vice versa. Dies ermöglicht einen einfachen Zugriff auf die Datenbank und den darauf befindlichen Daten. *JPA* und *Eclipselink* übernehmen zusätzlich das Caching von Daten für verbesserte Performance. Der Zugriff auf Daten erfolgt über Instanzen der *EntityManager* Klasse. Da in dieser Anwendung nur eine Instanz dieser Klasse aufgrund von Datenkonsistenz läuft, wird *Guice* für die Instanzverwaltung verwendet. *Guice* bietet mit *Guice Persist* hierfür das passende Framework an [22].

### 5.2.5 Map Struct

Mit *MapStruct* [23] wird ein weiteres Framework zur Unterstützung und Vereinfachung des Entwicklungsprozesses sowie zur Steigerung der Lesbarkeit von Code verwendet.

*MapStruct* bietet unter anderen das ”Mapping” von Objekten an, also das Transformieren von Objektinstanzen zu Objektinstanzen anderer Klassen. Dieses Mapping wird im System verwendet, um zum Beispiel Objekte, die für die Kommunikation mit der REST-Schnittstelle verwendet werden, zu entsprechenden Objektinstanzen von Klassen, die für die Datenbank verwendet werden, zu transformieren. Der entsprechend benötigte Code wird im Hintergrund beim Kompilieren über einen *Annotations Prozessor* automatisch generiert. Listing 5.2 beispielsweise erhält von der Methode *getByCode* ein *Questionnaire* Objekt, welches von dem *QuestionnaireMapper* automatisch zu einem Objekt der Klasse *QuestionnaireResponse* transformiert wird. In Anhang A.1 wird die Klasse *QuestionnaireMapper* beschrieben.

```
1 public QuestionnaireResponse receiveCode(@PathParam("code") final
     String code) {
2     return QuestionnaireMapper.INSTANCE.map(codeService.getByCode(
          code));
3   }
```

Listing 5.2: MapStruct Besipiel

### 5.2.6 JXLS

Um Excel Dateien schnell und einfach zu erstellen, wird das *JXLS* Framework verwendet [24]. Dafür wird eine Vorlage verwendet, in der Daten automatisch eingetragen werden, wodurch das Design einheitlich gehalten wird. In Anhang A.2 wird der Code zum Erstellen und Ausfüllen einer Exceldatei mit JXLS vorgestellt.

### 5.2.7 Camunda BPM

*Camunda* [25] ist ein Unternehmen welches Software in Hinblick auf die Modellierung und Ausführung von Geschäftsprozessen entwickelt. Neben Softwarelösungen zur Modellierung von Geschäftsprozessen bietet *Camunda* eine Open-Source Bibliothek zur

Modifikation einer Modellierungsumgebung an. Die *Camunda BPM* Plattform ermöglicht das Lesen eines Modells. Dies erleichtert das Analysieren und Evaluieren von BPMN 2.0 Modelle auf den Server. In Anhang A.3 wird eine Methode vorgestellt, Die anhand eines gegebenen Prozessmodells mithilfe von Camunda die verschiedenen benötigten Metriken abliest.

### 5.2.8 React

Zur Erstellung von interaktiven Benutzeroberflächen im Browser wird *React* [26] verwendet. Mithilfe von *React* wird der Aufbau der Seite vollständig von *JavaScript* übernommen. Dies ermöglicht das Einbinden von eigenen Komponenten in der HTML. Diese Komponenten sind beliebig austauschbar, ohne dass die Seite neu geladen werden muss, wodurch es möglich ist, eine Anwendung im Browser zu simulieren. Das System ist eine *Single-Page-Applikation*, also eine Anwendung die auf einer einzigen Internetseite basiert. Navigation und Datenhaltung wird vom React-Framework übernommen. *React* ermöglicht die Erstellung von performanten und interaktiven Benutzeroberflächen im Browser über JavaScript.

### 5.2.9 Typescript

Eine Besonderheit von *JavaScript* ist der Verzicht von Typisierung. Dies hat einige Vorteile, jedoch erschwert es die Verständlichkeit und Lesbarkeit von Code und erhöht die Wahrscheinlichkeit von syntaktischen Fehlern. *TypeScript* [27] ist ein Javascript Framework, welches das Schreiben von typisierten JavaScript Code ermöglicht. Vorteile von TypeScript sind unter anderem *Type Checking* und gesteigerte Wiederverwendbarkeit von Code. Der TypeScript-Code wird zur Laufzeit in normalen JavaScript-Code kompiliert und ist somit auf jeden Browser ausführbar.

### 5.2.10 Redux

Um auf der Webanwendung die Datenhaltung besser zu verwalten, wird das *Redux* Framework [28] verwendet. *Redux* stellt in der Anwendung eine definierbare Datenstruktur zur Verfügung, auf die React-Komponenten zugreifen. Diese Datenstruktur ist zustandsdefiniert, das bedeutet, dass die Daten sich nur von Zustand zu Zustand ändern. In Komponenten werden zum Beispiel durch Benutzerinteraktionen *"Actions"* ausgelöst, welche von *Redux* registriert werden. Abhängig vom vorherigen Zustand und der Action wird ein neuer Zustandsbaum aufgebaut. Diese Datenstruktur wird im Browser-Cache gespeichert, wodurch Daten über Sessions erhalten bleiben.

### 5.2.11 bpmn-js

*bpmn-js* [29] ist ein JavaScript Framework zur Erstellung und Anzeige von BPMN 2.0 Modellen im Browser. Dieses Framwork wurde ebenfalls von Camunda entwickelt. In Anhang A.10 wird der Code für die Darstellung der Modellierseite vorgestellt, die über *bpmn-js* eine Modellierumgebung zur Modellierung von Prozessmodellen in BPMN 2.0 bereitstellt.

### 5.2.12 Bulma

Das Layout von HTML Seiten wird mit *Cascading Stylesheets (CSS)* definiert. Um das Layout einheitlich, schön und einfach zu halten, wird auf das *Bulma* Framework zurückgegriffen. *Bulma* [30] bietet eine große Bibliothek verschiedener Style-Definitionen an. Die bereitgestellten CSS-Klassen vereinfachen das Entwickeln, reduzieren Fehler und ermöglichen das korrekte Anzeigen der Anwendung auf mobilen Endgeräten.

## 5.3 Aufbau

Das System ist in zwei Bereiche ausgeteilt: Back-End (Server) und Front-Eend (Webanwendung). Diese zwei Bereiche sind jeweils unterschiedlich aufgebaut, weswegen eine Trennung sinnvoll ist. In den folgenden zwei Abschnitten wird auf den jeweiligen Aufbau von Back-End und Front-End eingegangen.

### 5.3.1 Back-End

Die Hauptstruktur des Server wird von *Dropwizard* bestimmt. Mit der Klasse *ApiApplication* wird der Server gestartet und alle benötigten Module registriert (siehe Anhang A.4). Benötigte Konfigurationen werden von der .yml-Datei mithilfe der Klasse *ApiConfigurations* bereitgestellt. Im *ApiModule* werden Klassen-Instanzen für *Guice* registriert (siehe Anhang A.5). Über *Jersey* werden in *Resource*-Klassen die REST-Schnittstellen definiert. In Listing 5.3 wird die Resource für das Anfordern eines Fragebogens mithilfe eines Codes vorgestellt.

```
1  /**
2   * Resource for requesting a questionnaire using a code identifying
        the questionnaire
3   */
4  @Path("/code")
5  @Produces(MediaType.APPLICATION_JSON)
6  public class CodeResource {
7
8    private final CodeService codeService;
9
10   @Inject
11   public CodeResource (CodeService codeservice) {
12     this.codeService = codeservice;
13   }
14
```

```
15    /*
16     * Requests a questionnaire to be filled in using a code
          corresponding to the questionnaire.
17     * Does not require an authentication token
18     */
19    @GET
20    @Path("/{code}")
21    @Transactional(rollbackOn = Exception.class)
22    public QuestionnaireResponse receiveCode(@PathParam("code") final
          String code) {
23      return QuestionnaireMapper.INSTANCE.map(codeService.getByCode(
            code));
24    }
25  }
```

Listing 5.3: Resource für die Verarbeitung von Code-Anfragen

Zunächst wird der URI-Pfad definiert. Um beispielsweise einen Fragebogen mit dem Code ”Beispielcode” anzufordern, muss auf den Serverpfad */api/code/Beispielcode* über eine GET-Anfrage zugegriffen werden. Die Anfrage wird auf eine Instanz der Klasse *CodeService* weitergeleitet. Die Methode *getByCode* gibt den Fragebogen zurück. Wird kein Fragebogen gefunden, wird eine *ApplicationException* ausgelöst. Die Klasse *ApplicationExceptionMapper* verarbeitet ausgelöste *ApplicationException* und lässt den Server mit einer entsprechenden Nachricht antworten. Die *ApplicationException* definiert die Nachricht und den HTTP-Statuscode dieser Antwort. In den Anhängen A.6 und A.7 werden diese beiden Klassen vorgestellt.

Die Klasse *CodeService* ist eine Service-Klasse. Zu den Aufgaben von Service-Klassen gehören das Verarbeiten von Anfragen und das Verbinden mit der Datenbank. In Listing 5.4 wird die Klasse *CodeService* vorgestellt.

```
1  /**
2   * Service class for retrieving a questionnaire with the given code
       from the database
3   */
4  public class CodeService {
5
6    private final Provider<EntityManager> em;
7
8     @Inject
9     CodeService(Provider<EntityManager> em) {
10       this.em = em;
11    }
12
13    /*
14     * Returns the questionnaire corresponding to the given code or
         throws an exception if unable to find any
15     */
16   public Questionnaire getByCode(@Nonnull final String code) {
17     try {
18       return em.get().createNamedQuery("Questionnaire.findByCode",
           Questionnaire.class)
19         .setParameter("code", code)
20         .getSingleResult();
21     } catch (NoResultException e) {
22       throw new ApplicationException("Der \u00fcbergebene Code ist
           ung\u00fcltig", Response.Status.NOT_FOUND.getStatusCode());
23     }
24   }
25 }
```

Listing 5.4: Service-Klasse für das Überprüfen von Code-Anfragen

Auch hier wird wieder über Guice eine benötigte Instanz der Klasse *EntityManager* bereitgestellt, um sich mit der Datenbank zu verbinden. In der Funktion *getByCode*

wird in der Datenbank nach dem entsprechenden Fragebogen (*Questionnaire*-Klasse) gesucht. Falls kein Fragebogen gefunden wird, wird eine *ApplicationException* ausgelöst. Klassen, die auf Entitäten in der Datenbank abbilden, wie beispielsweise *Questionnaire* sind *Model*-Klassen (siehe Anhang A.8). Klassen, welche in Anfragen und Antworten über JSON serialisiert werden, wie beispielsweise *QuestionnaireResponse*, werden als *Transfer*-Klassen bezeichnet (siehe Anhang A.9).

Der Server verfügt über zwei unterschiedliche Arten von Authentifizierungsverfahren. Über BasicAuth wird der Login ermöglicht. Bei diesen Verfahren wird der Benutzername und das Passwort verschlüsselt zum Server geschickt. Nach erfolgreichen Login generiert der Server ein Token für den Benutzer, mit dem er sich für weitere Anfragen authentisiert. Dieses Token verwendet den JWT-Algorithmus und besitzt neben Angaben zum Benutzer ein Verfallsdatum. Läuft ein Token aus, muss sich der Benutzer neu anmelden.

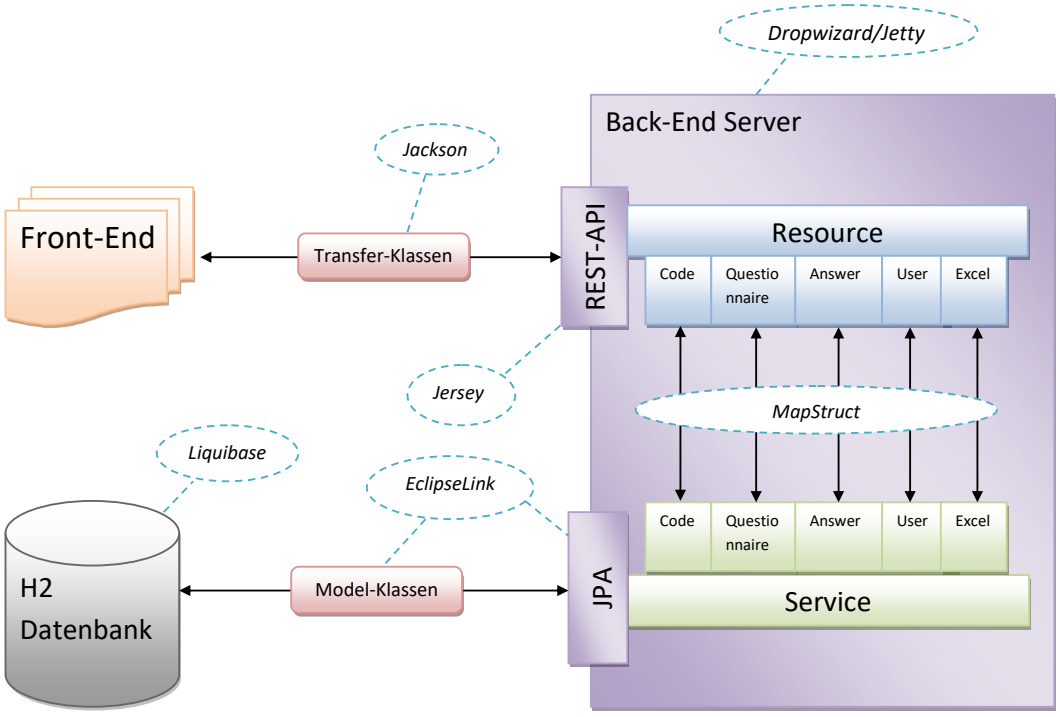



Abbildung 5.2: Aufbau des Back-End's

Der Server wird von *Jetty* über *Dropwizard* bereitgestellt. Eingehende Anfragen werden von *Jersey* in Resource-Klassen angenommen und zu Service-Klassen weitergeleitet. Diese sind über *EclipseLink JPA* mit einer H2-Datenbank verbunden. *Guice* stellt die benötigten Klasseninstanzen zur Verfügung. Mithilfe von *Jackson* werden die *JSONs* de-/serialisiert. *Liquibase* übernimmt das Erstellen des Datenbankschemas. Mithilfe von *MapStruct* werden Instanzen zwischen Model- und Transfer-Klassen transformiert. In Abbildung 5.2 ist der Serveraufbau im Detail abgebildet.

### 5.3.2 Front-End

Zunächst wird der generelle Ablauf zum Zugriff auf die Webanwendung beschrieben. Greift ein Benutzer auf die Webseite über *HTTP* beziehungsweise *HTTPS* zu, antwortet der Server mit allen Daten, die zur Darstellung der Webanwendung im Browser benötigt werden. Im Allgemeinen wird die Struktur der Webseite über *HTML* beschrieben, *CSS* definiert das Aussehen der Webseite und mithilfe von *JavaScript* wird die clientseitige Logik implementiert. In dieser Webanwendung übernimmt JavaScript über *React* [26] den Aufbau der HTML und bestimmt somit das Aussehen und das Verhalten der Webseite. Dies ermöglicht eine dynamische Anwendung im Browser. *React* baut über JavaScript die HTML-Struktur dynamisch mithilfe von Komponenten auf. Dafür wird in einer React-Komponente die zu erzeugende HTML-Struktur im XML-Format definiert. Diese besitzt gegebenenfalls zusätzlich zu *HTML* weitere React-Komponenten, CSS-Klassen und -Definitionen. Da *React* in JavaScript definiert wird, ist auch das Einbinden weiterer Funktionen möglich. Bei der Deklaration von weiteren Komponenten ist es möglich, diesen Komponenten Variablen und Funktionen zu übergeben, welche als *Props* (Kurzform für Properties/Eigenschaften) bezeichnet werden. Über diesen Weg werden zwischen Komponenten Informationen ausgetauscht. *React* bietet zusätzlich bestimmte Einstiegspunkte für Funktionen an, die vom React-Lebenszyklus definiert sind, zum Beispiel der Zeitpunkt, zu dem eine Komponente aufgebaut wird. React-Dateien werden aufgrund der Erweiterung um eine XML-Strukturen mit der Dateiendung *.jsx* statt *.js* definiert, beziehungsweise bei *TypeScript* mit *.tsx*. TypeScript erweitert JavaScript um Typdefinitionen und erleichtert das Arbeiten und Einarbeiten in komplexe JavaScript-

Anwendungen. In Listing 5.5 ist eine einfache im Projekt verwendete React-Komponente beschrieben.

```
1  import * as React from 'react';
2  import { Question } from '../../state-management/questionnaire';
3
4  interface Params {
5     readonly question: Question; // The question to display
6     readonly position: number; // The position of the question on the
          page
7     readonly handleChange: (value: string, position: number) => void;
8  }
9
10 interface State {
11    value: string; // The given answer
12 }
13
14 /**
15  * Render a text input field for questions with the type TEXT_INPUT
16  */
17 export class QuestionTextInput extends React.Component<Params, State>
       {
18
19    constructor(props: Params) {
20       super(props);
21       this.state = {value: props.question.answer ? props.question.
             answer : ''};
22
23       this.handleChange = this.handleChange.bind(this);
24    }
25
26    public render() {
27       return (
28          <div className="field">
```

```
29              <label className="label">
30                  {this.props.question.question}
31              </label>
32              <div className="control">
33                  <input
34                      className="input"
35                      type="text"
36                      value={this.state.value}
37                      onChange={this.handleChange}
38                  />
39              </div>
40          </div>
41      );
42  }
43
44  // Saves the current answer to the state of the PageInputComponent
        on changes
45  private handleChange(event: any) {
46      this.setState({value: event.target.value});
47      this.props.handleChange(event.target.value, this.props.position
            );
48  }
49 }
```

Listing 5.5: QuestionTextInput-Komponente für das Beantworten einer Frage

Diese Komponente zeigt eine Frage an und bietet als Antwortmöglichkeit ein Texteingabefeld. *Params* bezeichnen die Variablen und Übergabeparameter, die diese Komponente von der Vaterkomponente erhalten hat. Diese Komponente verwaltet die Antwort des Benutzers in einem eigenen Zustand. Dieser Zustand wird mit der Funktion *handleChange* an die Vaterkomponente gegeben, welche die getätigten Antworten verwaltet. In der Funktion *render* wird die Struktur der Komponente definiert.

Um die Datenhaltung von *React* zu erweitern wurde das Framework *Redux* verwendet. *Redux* verwaltet einen sogenannten *Store*, in dem der aktuelle Datenzustand in einem *State* gespeichert wird. Komponenten werden mit dem *Store* verbunden, wodurch sie Daten aus dem *State* als *Props* erhalten. Um den *State* zu ändern, müssen die Komponenten *Actions* auslösen. Diese Aktionen werden von Redux in einem *Reducer* verarbeitet. Im *Reducer* wird definiert, wie sich die Daten im *State* abhängig von der *Action* und den vorherigen Datenzustand verändern. Die verbundenen Komponenten werden anschließend mit den neuen Daten versorgt. In den Anhängen A.11, A.12 und A.13 werden State, Actions und Reducer zur Verwaltung des Login-Zustandes vorgestellt. Der Lebenszyklus von *Redux* wird in Abbildung 5.3 visualisiert.

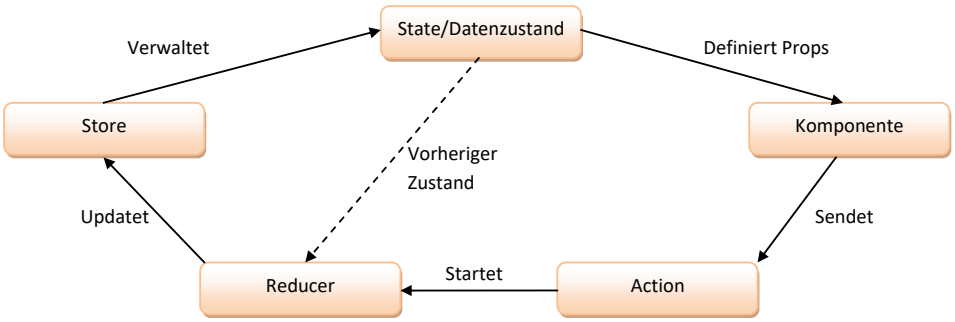



Abbildung 5.3: Redux-Lebenszyklus

Dieser Redux-State wird im Browser-Cache gespeichert. Dies ermöglicht das Herstellen der vorherige Sitzung beim neuen Laden der Seite. Da die Anwendung auf einer einzigen Seite benutzt wird (*Single-Page-Application*), wird im State die Navigation der Seite übernommen. Zusätzlich werden alle Daten vom Server wie Logindaten oder Fragebögen und Benutzereingaben im State abgespeichert. Anfragen zum Server werden über eine Middleware realisiert. Über die Middleware werden Aktionen ausgelöst, sobald eine Antwort vom Server erhalten wurde. In Abbildung 5.4 ist die Architektur der Web Anwendung abgebildet.

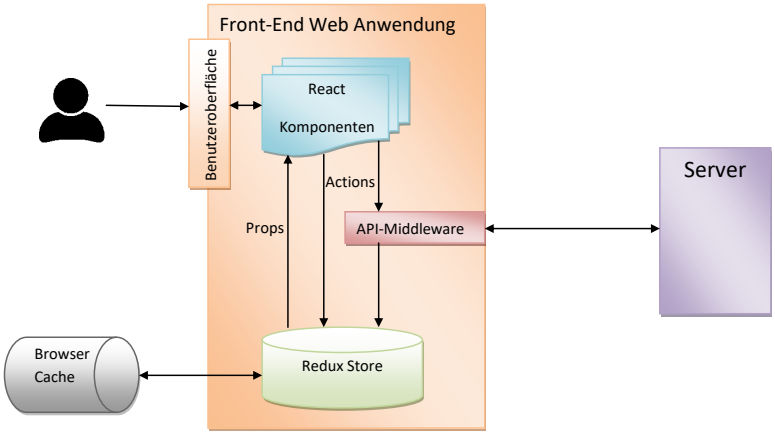



Abbildung 5.4: Aufbau des Front-End's

## 5.4 REST-Schnittstellen

Die Serverschnittstellen sind über *REST* beschrieben. *REST* (Representatational state transfer) ist ein Programmierparadigma zur Entwicklung und Bereitstellung von serverseitigen einheitlichen Schnittstellen zur Kommunikation und Interaktion zwischen Client (Browser) und Server [31]. Über *URIs* (Uniform Resource Identifier) und *HTTP-Protokoll* sendet ein Client (Browser) Anfragen zum Server. Der Server verarbeitet diese Anfragen und antwortet mit einen *HTTP-Statuscode* und gegebenenfalls mit gesendeten Daten. Diese Daten sind meistens im *JSON*-Format. Auch Dateien, HTML-Seiten und andere Datenformate werden auf diesen Weg versendet, wodurch weitere Interaktionsmöglichkeiten über *HTTP* ermöglicht werden. Im System ist es möglich, über vier verschiedene HTTP-Methoden auf den Server zuzugreifen. Über *GET*-Anfragen werden Daten angefordert. Über POST werden Daten geändert oder abhängig von dem im *JSON* befindlichen Objekt neu in der Datenbank angelegt. Über PUT werden Daten abhängig von der mitgelieferten JSON in der Datenbank geändert. In Tabelle 5.4 wird eine Zusammenfassung der verwendeten Schnittstellen gegeben.

| Methode | Pfad | Beschreibung | Berechtigung |
|---|---|---|---|
| GET | /api/code/code | Fragt einen Fragebogen abhängig von mitgegebenen *Code* an | Benutzer |
| POST | /api/answer | Schickt einen ausgefüllten Fragebogen zum Server | Benutzer |
| GET | /api/login | Loggt den Benutzer im System ein | Benutzer |
| GET | /api/questionnaire | Liefert eine Liste aller zugänglichen Fragebögen | Mitarbeiter |
| GET | /api/questionnaire/id | Liefert einen Fragebogen abhängig von der *ID* | Mitarbeiter |
| POST | /api/questionnaire | Erstellt einen neuen Fragebogen | Mitarbeiter |
| PUT | /api/questionnaire/id | Verändert den Fragebogen mit der gegebenen *ID* | Mitarbeiter |
| DELETE | /api/questionnaire/id | Löscht den Fragebogen mit der gegebenen *ID* | Mitarbeiter |
| GET | /api/answer/completed/id | Fragt einen ausgefüllten Fragebogen mit der gegebenen *ID* an | Mitarbeiter |
| GET | /api/answer/question/id | Fragt alle Antworten zu einer Frage an | Mitarbeiter |
| GET | /api/answer/page/id | Fragt alle Modelle zu einer Modellierseite an | Mitarbeiter |
| GET | /api/excel/id | Fordert eine Excel-Datei zum Fragebogen mit der *ID* an | Mitarbeiter |
| POST | /api/user/change-password | Verändert das Passwort eines Mitarbeiters | Mitarbeiter |
| GET | /api/user | Liefert eine Liste aller Mitarbeiter | Administrator |
| POST | /api/user | Erstellt einen neuen Mitarbeiter | Administrator |
| POST | /api/user/deactivate/id | Deaktiviert einen Benutzer mit der gegebenen *ID* | Administrator |
| POST | /api/user/activate/id | Aktiviert einen Benutzer mit der gegebenen *ID* | Administrator |

Tabelle 5.1: Verwendete REST-Schnittstellen

Die jeweiligen Berechtigungen werden über Basic-Authentifikation und JWT-Token Authentifikation realisiert. Jeder *Benutzer* besitzt ohne Authentifizierung Zugriff auf die Codeeingabe und ist in der Lage, Fragebögen auszufüllen. Für die Berechtigung *Mitar-*

*beiter* muss sich ein Anwender im System einloggen. Mitarbeiter besitzen Zugriff auf ihre eigenen Fragebögen. Ein *Administrator* ist in der Lage auf alle Fragebögen zuzugreifen und die Benutzer zu verwalten.

## 5.5 Datenmodell

In diesem Abschnitt wird ein Einblick über die Datenbank und das verwendete Datenmodell gegeben. Für das System wird ein relationales Datenbank Management System (*RDBMS*) für die Datenhaltung verwendet. Die Entscheidung ist auf eine *H2* Datenbank gefallen, da diese *ACID*-Transaktionen (Wahrung von Integrität und Konsistenz der Daten) unterstützt und leicht einzurichten ist. *H2* bietet die Möglichkeit die Datenbank als Datei (*Embedded Mode*) statt als Server anzulegen und ist dadurch leichtgewichtiger und portierbar [32]. Das verwendete Datenmodell wird in Abbildung 5.5 beschrieben.

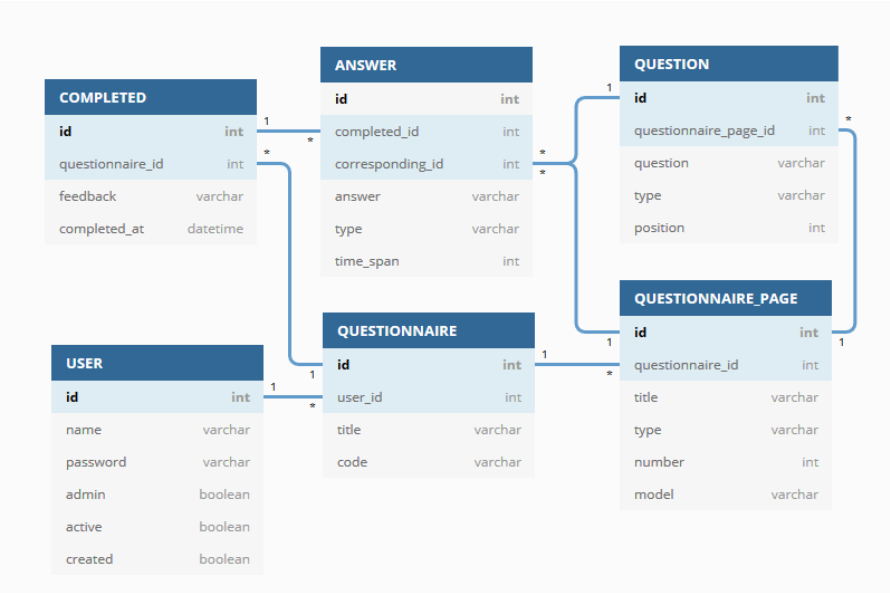



Abbildung 5.5: *ProMoEE* Datenmodell

In den nachfolgenden Abschnitten wird das Datenmodell erläutert.

### 5.5.1 Fragebögen

Jeder Fragebogen (*Questionnaire*) hat neben der ID auch einen einzigartigen Code, mit dem dieser Fragebogen identifizierbar ist. Ein Fragebogen wird aus mindestens einer Seite (*Questionnaire_Page*) aufgebaut. Eine Seite ist eine von drei verschiedene Typen zuteilbar:

1. **Fragebogen**: Einfache Seite, die nur Fragen anzeigt.
2. **Evaluierer**: Zeigt zusätzlich ein BPMN Modell an. Dies ermöglicht das Beziehen von Fragen auf das Modell. Das Modell wird in der *model* Variable definiert.
3. **Modellierer**: Zeigt keine Fragen an. Stattdessen wird eine Modellierungsumgebung zur Erstellung eines BPMN-Modells angezeigt. Über *model* ist ein Startmodell optional vorgebbar. Die Aufgabenstellung dieser Modellierseite wird im *title* definiert.

Die Seitenzahl wird in *number* definiert. Seiten vom Typ Fragebogen und Evaluierer beinhalten eine bis beliebig viele Fragen (*Question*). Neben der eigentlichen Frage (*question*) gibt es vier mögliche unterschiedliche Antworttypen:

1. **Textfeld**: Einfaches Antwortfeld zur Eingabe von kurzen Text.
2. **Textbereich**: Antwortfeld zur Eingabe von langen Text.
3. **Single-Choice**: Der Benutzer ist in der Lage, aus einer Auswahl von vorgefertigte Antworten genau eine Antwort auszuwählen.
4. **Multiple-Choice**: Der Benutzer besitzt die Möglichkeit, aus einer Auswahl beliebig viele Antworten auszuwählen.

Über *position* wird die Reihenfolge der Fragen auf der Seite festgelegt.

### 5.5.2 Ausgefüllte Fragebögen

Einen Fragebogen lassen sich beliebig viele ausgefüllte Fragebögen (*Completed*) zuordnen. Als ausgefüllte Fragebögen werden die getätigten Antworten von einem Benutzer zu einen Fragebogen in einer Ausfüllsitzung bezeichnet. Neben den Antworten wird hier

auch das Feedback und der Zeitpunkt gespeichert, wann der Fragebogen ausgefüllt wurde (*completed_at*). Ein ausgefüllter Fragebogen hat eine feste Anzahl von Antworten (*Answer*). Diese Anzahl ist abhängig von der Anzahl der Fragen und Modellierseiten des zugehörigen Fragebogens. Antworten nehmen einen von zwei verschiedene Typen an. Bei Antworten zu einer Fragen referenziert *corresponding_id* eine Frage. Bei einen erstellten Modell zu einer Modellierseite referenziert *corresponding_id* eine Seite vom Typ Modellierer. Die Antwort beziehungsweise das Modell wird in *answer* gespeichert. Zusätzlich wird zu jeder Antwort die Zeitdauer gespeichert, die der Benutzer zum Ausfüllen einer Seite benötigt hat.

### 5.5.3 Benutzer

Als Benutzer (*User*) wird in diesem Falle ein Mitarbeiteraccount bezeichnet. Ein Benutzer besitzt Zugriff auf die von ihm erstellten Fragebögen. Das Passwort eines Benutzers wird verschlüsselt in der Datenbank gespeichert. Über das Feld *admin* wird definiert, ob ein Benutzer zusätzliche Administratorrechte besitzt. Um einen Benutzer anzulegen, muss ein Administrator einen Benutzer mit Namen und Passwort erstellen. Beim erstmaligen Login erhält dieser Benutzer die Möglichkeit, sein Passwort einmalig zu ändern. Dadurch wird sichergestellt, dass nur der Benutzer sein Passwort kennt. Dieser Vorgang wird über das Feld *created* definiert. Ein Administrator besitzt die Möglichkeit, Benutzer zu deaktivieren und zu aktivieren. Die Variable *active* bezeichnet ob ein Benutzer aktiv oder deaktiviert ist.

# 6

# Implementierung

In diesem Kapitel wird die Implementierung von *ProMoEE* vorgestellt. Es wird auf die jeweiligen Sichten und Interaktionsmöglichkeiten eingegangen.

## 6.1 Startseite

Beim erstmaligen Laden der Internetseite landet ein Benutzer auf der Startseite. Hat der Benutzer bereits die Seite besucht (mit dem selben Browser), wird er auf die von ihm zuletzt besuchte Seite navigiert. Auf der Startseite wird ein Willkommenstext angezeigt und der Benutzer erhält die Möglichkeit, einen Code einzugeben. Des Weiteren besitzt der Benutzer über einen Knopf rechts oben Zugriff zum Mitarbeiterbereich (Login erforderlich). In Abbildung 6.1 wird die Startseite vorgestellt.

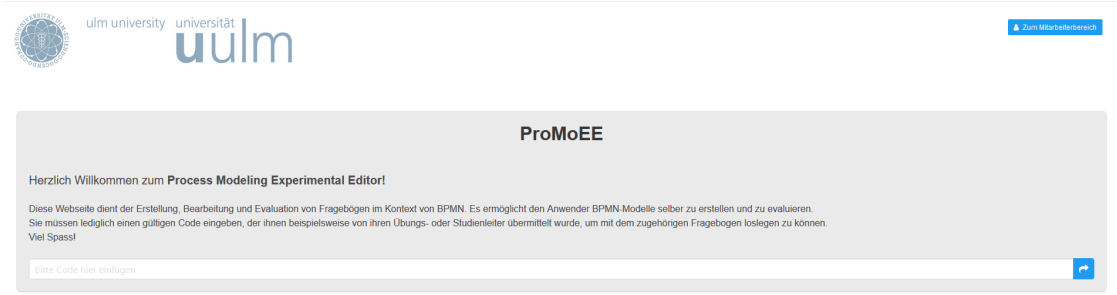



Abbildung 6.1: *ProMoEE* Startseite

## 6.2 Fragebogen ausfüllen

Nach erfolgreicher Code-Eingabe wird der entsprechende Fragebogen angezeigt. Jeder Fragebogen wird in Seiten unterteilt. Es wird zwischen normalen Fragebogenseiten und Modellierseiten unterschieden. Eine normale Fragebogenseite mit Fragen wird in Abbildung 6.2 angezeigt.


Abbildung 6.2: Fragebogenseite mit Fragen

Zu jeder Frage ist eine von vier unterschiedlichen Antwortarten zuteilbar. Diese sind Textfeld, Textbereich, Single-Choice-Auswahl (nur ein Element ist auswählbar) und Multiple-Choice-Auswahl (beliebig viele Elemente sind auswählbar). Es ist möglich, eine Fragebogenseite um die Anzeige eines Modells zu erweitern, wie in Abbildung 6.3 dargestellt. Dies ermöglicht das Stellen von Fragen bezüglich eines Modells.

In Abbildung 6.4 wird hingegen eine Modellierseite angezeigt. Anhand einer Aufgabenstellung ist der Benutzer mithilfe einer Modellierungsumgebung in der Lage, ein eigenes Modell zu erstellen. Es ist möglich ein Startmodell, wie in Abbildung 6.4 dargestellt,

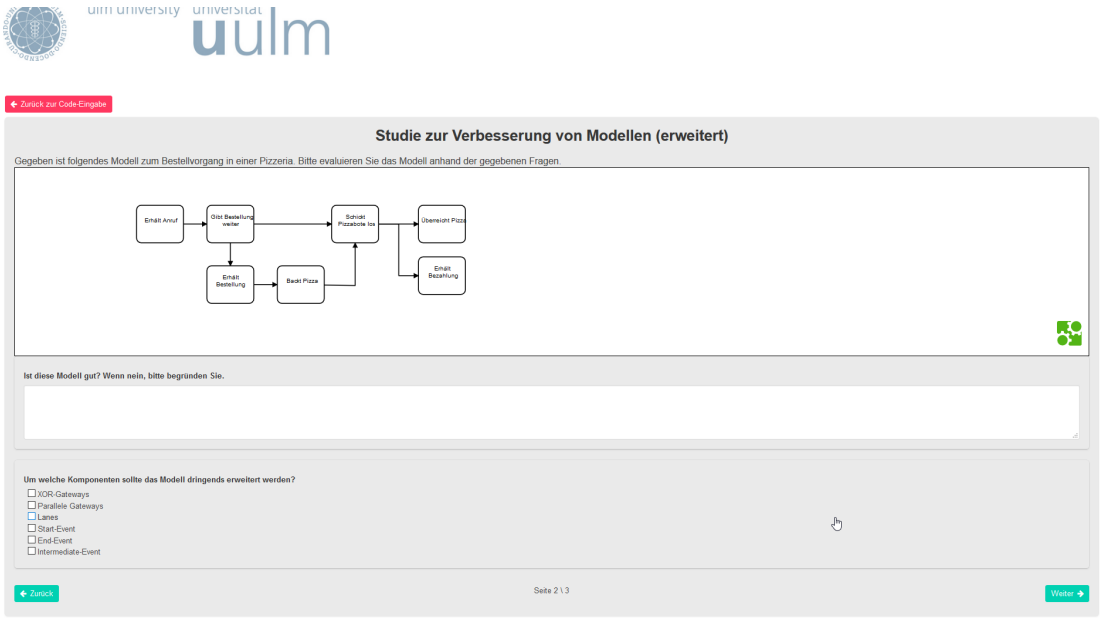



Abbildung 6.3: Fragebogenseite mit Modell und Fragen

vorzugeben. Über die Navigation am Fuß des Fragebogens besteht die Möglichkeit für den Benutzer, zwischen den Fragebogenseiten zu navigieren.

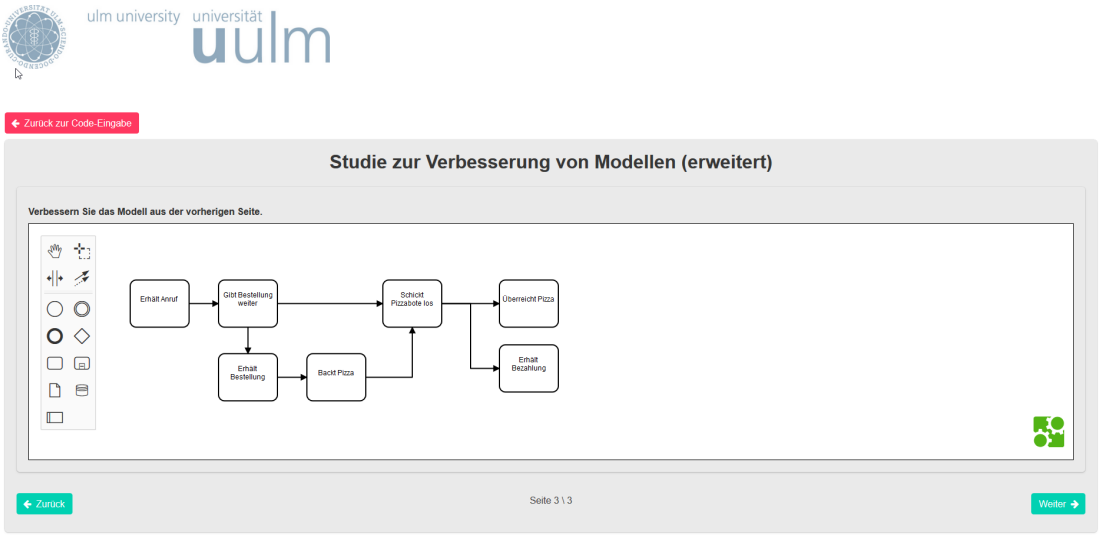



Abbildung 6.4: Fragebogenseite zur Erstellung eines Modells

Am Ende eines Fragebogens erhält der Benutzer die Möglichkeit Feedback zum Fragebogen zu geben und den ausgefüllten Fragebogen abzusenden. Diese Seite wird in Abbildung 6.5 abgebildet.


Abbildung 6.5: Feedbackseite am Ende eines Fragebogens

Zu jedem Zeitpunkt besteht die Möglichkeit, das Ausfüllen des Fragebogens abzubrechen. Da dadurch alle Eingaben verloren gehen, wird der Benutzer in einem zusätzlichen Dialogfenster wie in Abbildung 6.6 um Bestätigung gebeten.

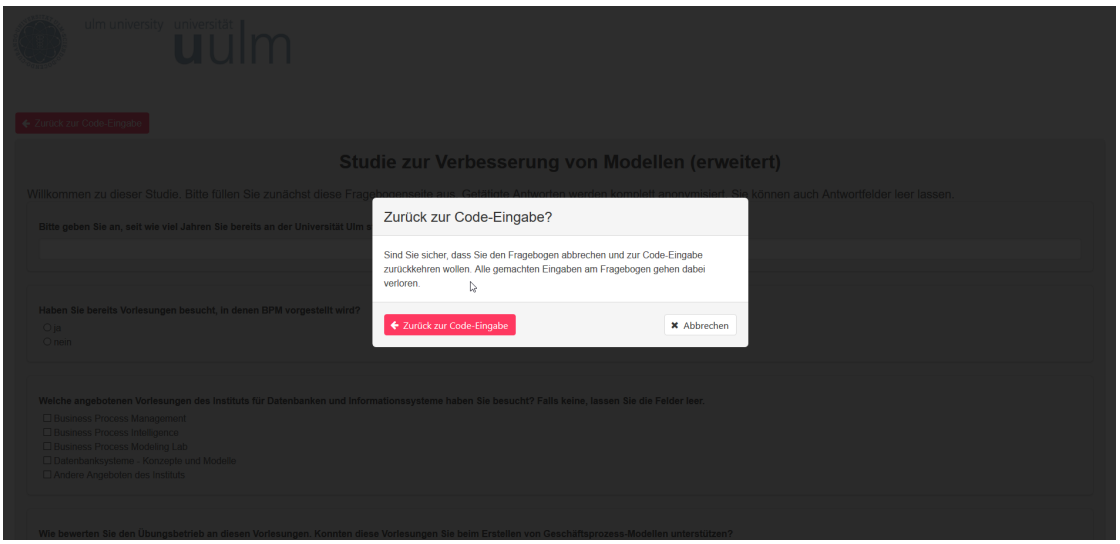



Abbildung 6.6: Warnung beim Abbruch eines Fragebogens

### 6.2.1 Einloggen

Auf der Startseite besitzt ein Benutzer die Möglichkeit, sich durch einen Klick auf das entsprechende Feld rechts oben in den Mitarbeiterbereich zu begeben. Der Zugang zu sämtlichen Bereichen des Mitarbeiterbereichs ist nur für Mitarbeiter über einen Login möglich. In Abbildung 6.7 wird das Login-Formular zur Eingabe seines Benutzernamens und Passworts angezeigt.

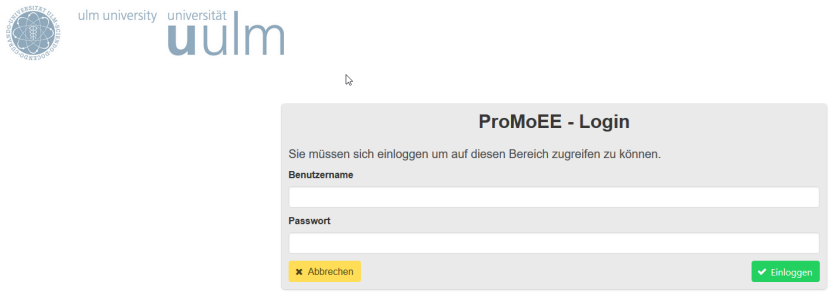



Abbildung 6.7: Login-Formular

Ist der Login erfolgreich, erhält die Anwendung ein Authentifizierungs-Token, mit dem sich die Anwendung als der jeweilige Benutzeraccount beim Server ohne weiteren Login authentifiziert. Das Token ist fälschungssicher und beinhaltet neben den Benutzernamen auch eine Verfallsdauer von 4 Stunden. Wenn ein Token verfällt wird der Benutzer zum Login-Formular navigiert um sich erneut einzuloggen. Auf diese Weise unterbrochene Arbeiten werden gespeichert und stehen dem Benutzer nach erfolgreichen Login wieder zur Verfügung. Bei einem misslungenen Login-Versuch wird der Benutzer über den entsprechenden Fehler in roter Schrift informiert.

### 6.2.2 Übersicht

Nach einem erfolgreichen Login erreicht ein Benutzer in der Regel eine Übersichtsseite. Hier erhält ein normaler Benutzer eine Übersicht über alle von ihm erstellten und verwalteten Fragebögen. Administratoren erhalten Zugang zu allen Fragebögen im System und

zur Benutzerverwaltung. In der Übersicht wird der Name des Fragebogens, der zugehörige Code, die Anzahl der ausgefüllten Fragebögen und der Besitzer des Fragebogens angezeigt. Zudem ist es möglich hier weitere Fragebögen zu erstellen. In der Übersicht besteht die Möglichkeit, zurück auf die Startseite zu navigieren. Zu jedem Zeitpunkt im Mitarbeiterbereich steht eine Funktion zum Ausloggen des Mitarbeiters oben rechts zur Verfügung. In Abbildung 6.8 ist die Übersichtsseite mit einigen Beispielfragebögen abgebildet.

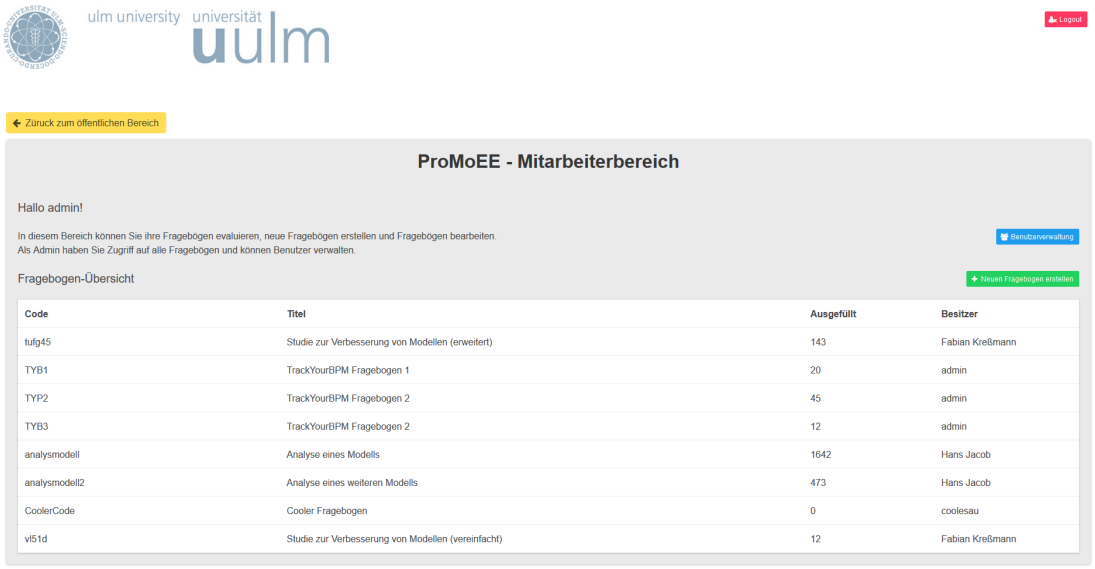



Abbildung 6.8: Übersichtsseite

### 6.2.3 Fragebogen erstellen/bearbeiten

Das Erstellen, Bearbeiten und Kopieren von Fragebögen benutzt die selbe Ansicht. Beim Kopieren und Bearbeiten wird ein Fragebogen vorgegeben, beim Erstellen muss der gesamte Fragebogen gebaut werden. Fragebögen sind in einer Baumstruktur aufgebaut. Ein Fragebogen hat Seiten, diese Seiten besitzen wiederum Fragen. In Abbildung 6.9 wird der Aufbau der Ansicht zum Erstellen beziehungsweise Bearbeiten von Fragebögen abgebildet. In dieser Ansicht wird der Fragebogen in einer Baumstruktur im linken Bereich

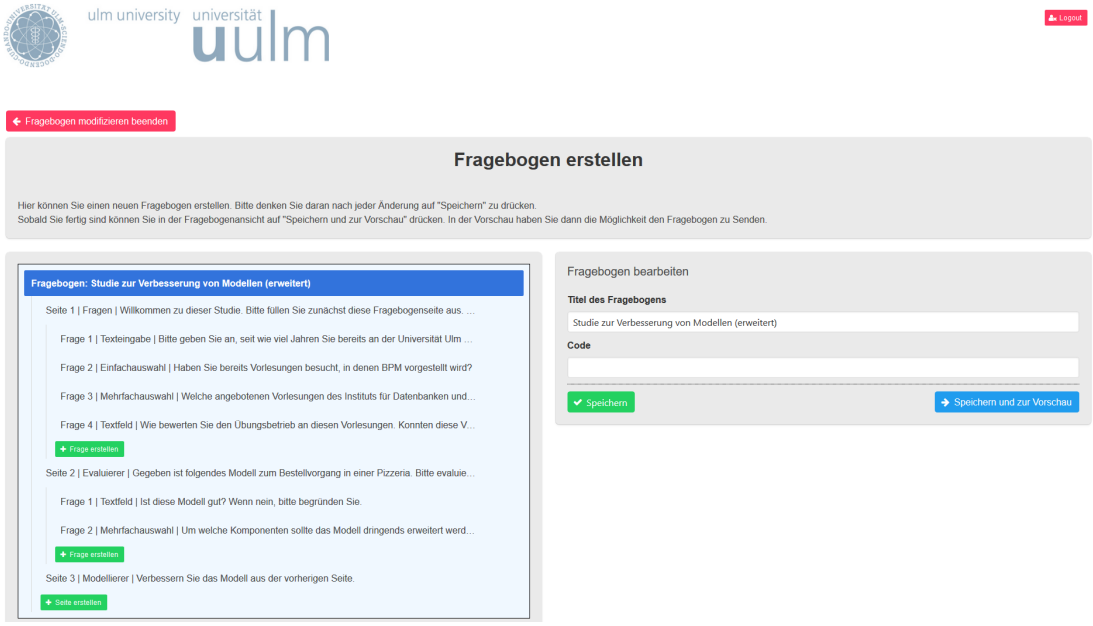



Abbildung 6.9: Erstellen eines Fragebogens

angezeigt. Der Fragebogen ist hier um Seiten und Fragen erweiterbar. Die Seiten, Fragen und der Fragebogen selbst sind in diesem Bereich auswählbar. Ausgewählte Elemente werden rechts zum Bearbeiten angezeigt.

In Abbildung 6.9 ist der Fragebogen ausgewählt und der Titel und Code des Fragebogens ist rechts bearbeitbar. In Abbildung 6.10 wird die Ansicht zur Bearbeitung einer Seite abgebildet. Neben Informationen zur Seite wird hier der Typ der Seite ausgewählt und gegebenenfalls das entsprechende Modell modelliert. Wenn eine Frage oder Seite ausgewählt ist, besteht zusätzlich im linken Abschnitt die Möglichkeit, die Position dieser Seite/Frage im Baum verändert werden. Auch die Funktion eine Seite/Frage zu löschen besteht. Abbildung 6.11 zeigt die Bearbeitung einer Frage an mit dem Typ Mehrfachauswahl. Je nach ausgewählten Typ verändert sich die Ansicht.

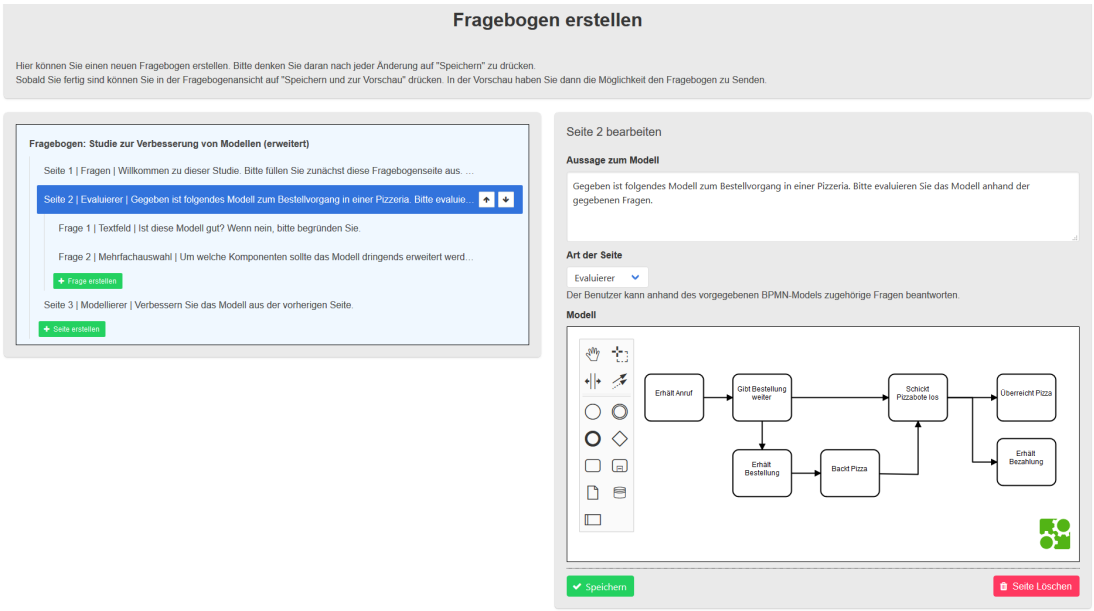



Abbildung 6.10: Bearbeitung einer Seite

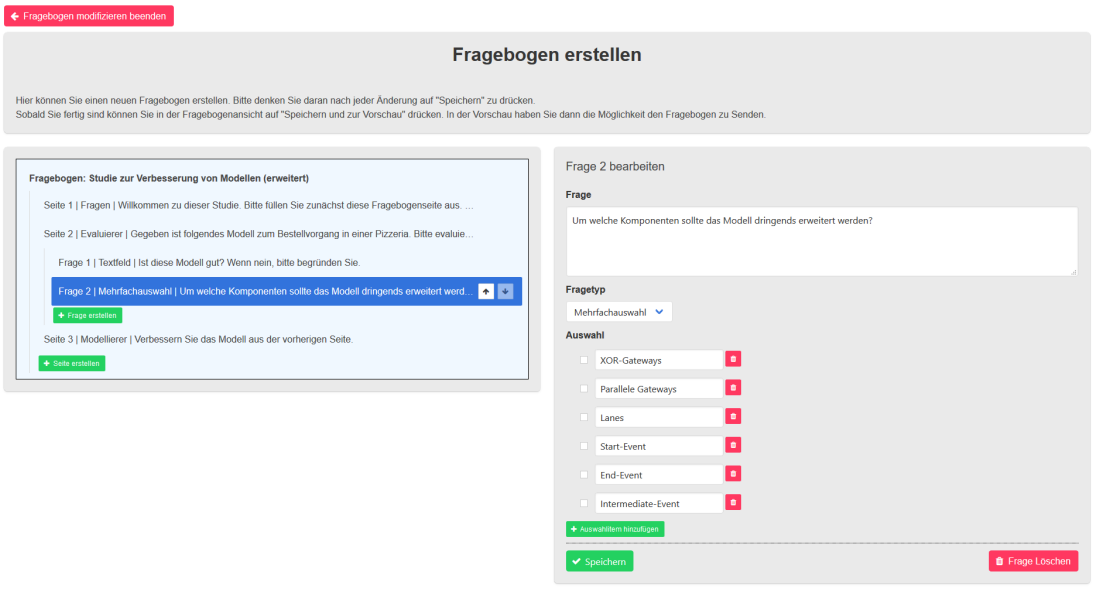



Abbildung 6.11: Bearbeitung einer Frage

Beim Speichern von Fragen und Seiten werden die gespeicherten Eingaben validiert und überprüft. Beispielsweise dürfen Fragen nicht leer oder zu kurz sein und eine Auswahl sollte Auswahlelemente besitzen. Der Vorgang ist jederzeit über die Navigation links oben abbrechbar. Da in diesen Vorgang Daten verloren gehen, wird wie in Abbildung 6.6 dargestellt eine Warnung angezeigt.

Ist der Benutzer mit dem Erstellen des Fragebogens fertig, ist es möglich, den Fragebogen über die Fragebogen-Bearbeiten-Ansicht aus Abbildung 6.9 abzuspeichern und mit der Vorschau fortzufahren. In diesem Schritt wird die ganze Struktur des Fragebogens validiert. Bei erfolgreicher Validierung wird dem Benutzer eine Vorschau des Fragebogens angezeigt. In dieser Vorschau sieht der Benutzer den Fragebogen so, wie dieser Fragebogen auch später beim Ausfüllen angezeigt wird. Über ein Feld wird der Fragebogen zum Server gesendet und dieser damit erfolgreich erstellt beziehungsweise modifiziert. Diese Vorschau wird in Abbildung 6.12 dargestellt.

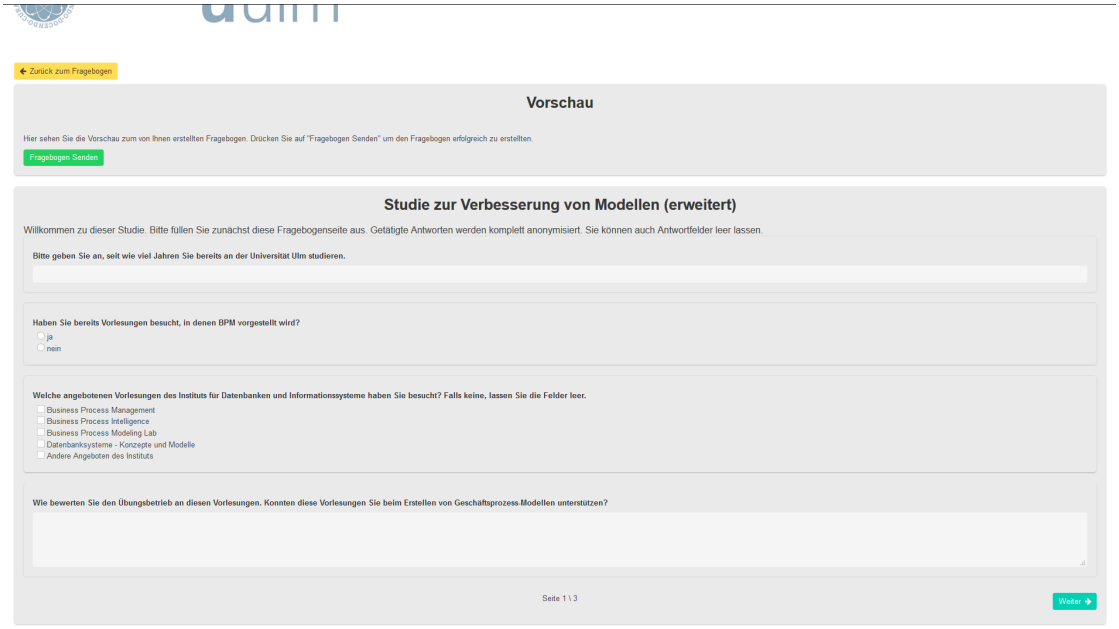



Abbildung 6.12: Fragebogen-Vorschau

### 6.2.4 Fragebogenansicht

Durch einen Klick auf einen Fragebogen in der Übersicht wird der Benutzer zur Fragebogenansicht navigiert. Hier wird eine Übersicht über den Fragebogen angezeigt, wie in Abbildung 6.13 dargestellt.


Abbildung 6.13: Übersicht zu einem Fragebogen

Oben links stehen neben dem Code auch Informationen über die durchschnittliche Vollständigkeit eines Fragebogens und die durchschnittlich benötigte Zeit zum Ausfüllen des Fragebogens. In der Tabelle werden numerische Informationen zum Fragebogen angezeigt. Rechts oben stehen Interaktionsmöglichkeiten zur Verfügung. Um Datenverlust zu vermeiden ist das Löschen und Bearbeiten eines Fragebogens nur möglich, wenn noch keine Antworten zu diesem Fragebogen bestehen. Eine Kopierfunktion steht jederzeit zur Verfügung, um den Benutzer die Verbesserung und Erweiterung von Fragebögen zu ermöglichen. Es ist denkbar, Fragebögen zu versionisieren. Über die Interaktion ”*Antworten herunterladen*” wird ein Excel-Dokument mit allen Antworten zu diesem Fragebogen erstellt. Unten werden Informationen zu allen ausgefüllten Fragebögen angezeigt. Neben der ID zur Identifikation wird das Ausfülldatum angegeben, die benötigte Zeit, ob

Feedback vorhanden ist, die Anzahl beantworteter Fragen und erstellter Modelle und der daraus ermittelte Grad der Vollständigkeit in Prozent. Eine genaue Übersicht über Seiten und Fragen wird durch einen Klick auf ”*Seiteninfos*” gegeben. Dadurch wird die Seitenübersicht aufgeklappt, auf der sämtliche Seiten und Fragen angezeigt werden. Zur Übersicht sind auch hier die Fragen einer Seite wieder auf- und zuklappbar. Abbildung 6.14 zeigt diese Seitenübersicht eines Fragebogens.

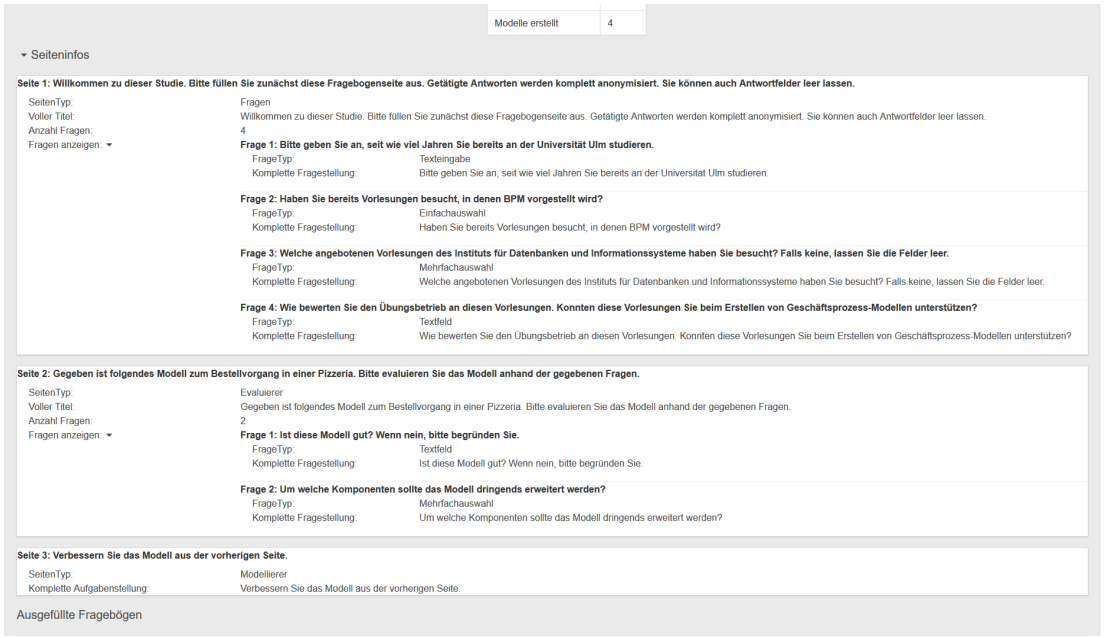



Abbildung 6.14: Übersicht zur Struktur eines Fragebogen

### 6.2.5 Fragebogen evaluieren

Es existieren verschiedene Möglichkeiten die Antworten zu einen Fragebogen darzustellen und zu evaluieren. Es besteht die Möglichkeit einen ausgefüllten Fragebogen anzuzeigen, wodurch dieser Fragebogen mit dem getätigten Antworten und erstellten Modellen dargestellt wird. Es ist möglich zwischen den einzelnen Seiten des ausgefüllten Fragebogens zu navigieren. Zusätzlich werden einige Informationen zum ausgefüllten Fragebogen wie Ausfülldatum und benötigte Zeit dargestellt. In Abbildung 6.15 ist eine Beispielseite eines ausgefüllten Fragebogens gegeben.


Abbildung 6.15: Anzeige eines ausgefüllten Fragebogen

Des Weiteren existiert die Möglichkeit Fragen und Modellierseiten zu evaluieren. Das umfasst die Anzeige aller Antworten zu einer Frage und aller erstellten Modelle zu einer Modellierseite. Der Benutzer erhält Einsicht über weitere Informationen zu der Frage, wie zum Beispiel die Fragestellung, wie viele Fragebögen zu diesen Fragebogen ausgefüllt wurden, wie viele Antworten gültig (nicht leer gelassen) sind und wie lange durchschnittlich für das Ausfüllen der Seite benötigt wurde. Fragen mit Auswahlmöglichkeiten zeigen

die Verteilung der Antwortmöglichkeiten in einem Kuchendiagramm an, wie in Abbildung 6.16 dargestellt.


Abbildung 6.16: Evaluation einer Frage

Darunter werden sämtliche getätigten Antworten aufgelistet mit der jeweiligen Bearbeitungsdauer der Seite, der ID des ausgefüllten Fragebogens und das Ausfülldatum. Besitzt die Fragebogenseite ein Modell, beispielsweise bei einer Evaluationsseite oder einer Modellierseite als vorgegebenes Modell, wird dieses Modell ebenfalls angezeigt. Bei der Evaluation von Modellierseiten wird die Aufgabenstellung und alle erstellten Modelle angezeigt. Die erstellten Modelle werden ausgewertet. Neben der Anzahl bestimmter Elemente, werden verschiedene Metriken in Prozent ermittelt. Dies umfasst den Grad der Beschriftung, die Schönheit (gerade verlaufende Sequenzflüsse) und den Workflow eines Modells (alle Elemente besitzen eine korrekte Anzahl eingehender und ausgehender Sequenzflüsse). In Abbildung 6.17 wird die Evaluationsansicht einer Modellierseite vorgestellt. Nach unten werden alle Antworten, beziehungsweise Modelle und zusätzliche Informationen aufgelistet. Es ist möglich, zwischen den verschiedenen Evaluationsansichten beliebig zu navigieren.

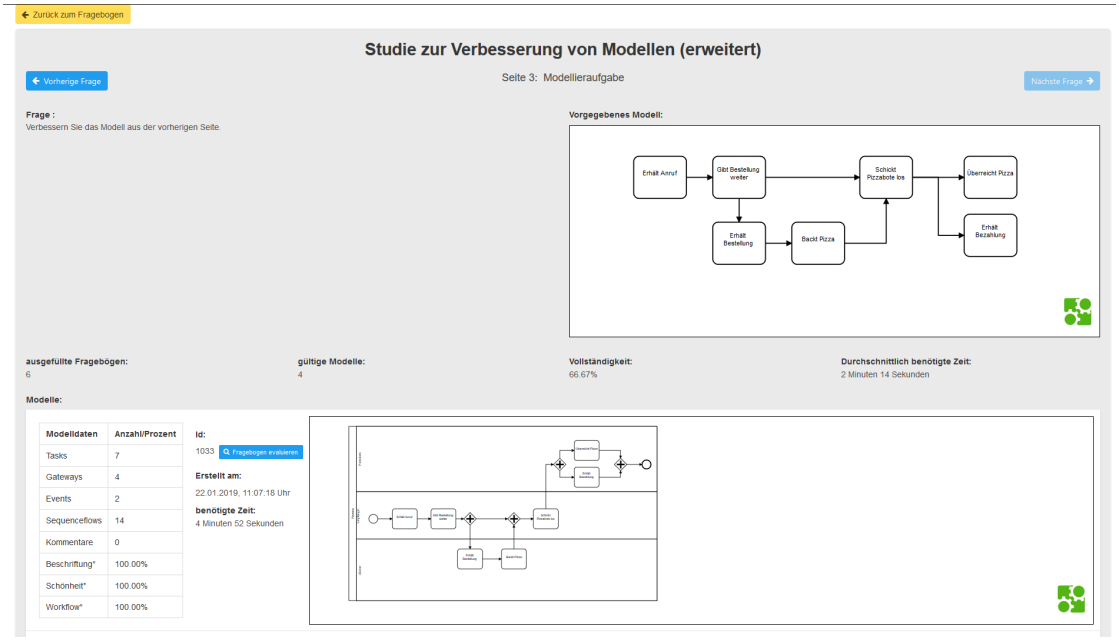
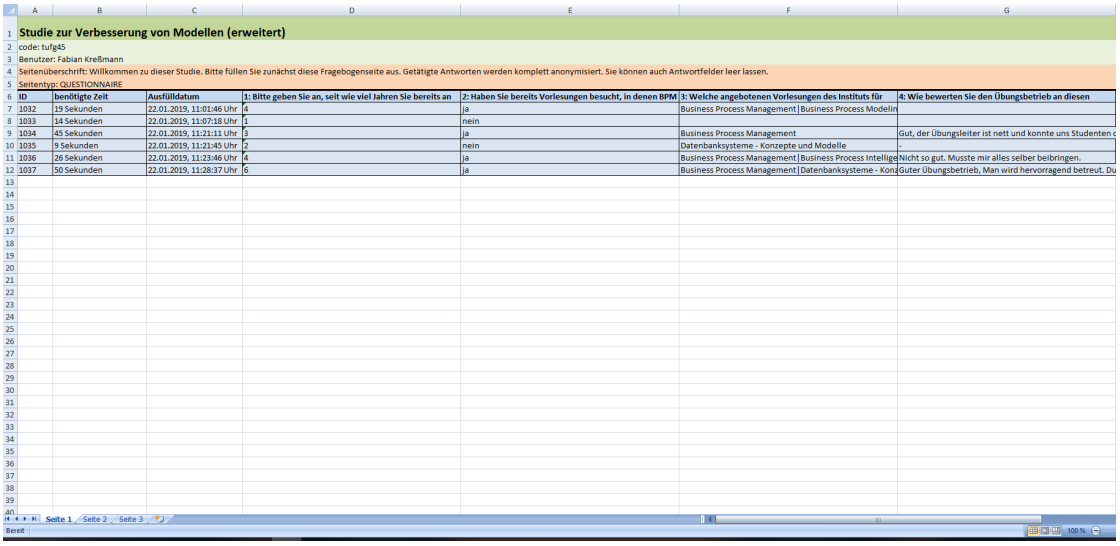



Abbildung 6.17: Evaluation einer Modellieraufgabe

Eine weitere Evaluationsmöglichkeit ist das Erstellen einer Excel-Datei zu einem Fragebogen. Dies ermöglicht das Darstellen aller Antworten zu einem Fragebogen in einer Excel. Jede Seite eines Fragebogens wird in eine andere Tabelle dargestellt. Oben werden zunächst Informationen zu der Fragebogenseite dargestellt, In der darauffolgenden Tabelle die getätigten Antworten. Eine Tabellenreihe stellt ein ausgefüllten Fragebogen dar. In den Tabellenspalten werden neben den Fragen auch weitere Angaben wie Ausfülldatum und benötigte Ausfülldauer angegeben. Im Falle einer Modellierseite wird statt den Fragen die verschiedenen Metriken wie Anzahl von Elementen, Beschriftung, Schönheit und Workflow angegeben. Das Modell an sich wird aus Platz- und Performancegründen nicht dargestellt. Abbildung 6.18 zeigt eine resultierende Exceltabelle zu einem Fragebogen an.



| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | **Studie zur Verbesserung von Modellen (erweitert)** | | | | | | |
| 2 | code: tufg45 | | | | | | |
| 3 | Benutzer: Fabian Kreßmann | | | | | | |
| 4 | Seitenüberschrift: Willkommen zu dieser Studie. Bitte füllen Sie zunächst diese Fragebogenseite aus. Getätigte Antworten werden komplett anonymisiert. Sie können auch Antwortfelder leer lassen. | | | | | | |
| 5 | Seitentyp: QUESTIONNAIRE | | | | | | |
| 6 | **ID** | **benötigte Zeit** | **Ausfülldatum** | **1: Bitte geben Sie an, seit wie viel Jahren Sie bereits an** | **2: Haben Sie bereits Vorlesungen besucht, in denen BPM** | **3: Welche angebotenen Vorlesungen des Instituts für** | **4: Wie bewerten Sie den Übungsbetrieb an diesen** |
| 7 | 1032 | 19 Sekunden | 22.01.2019, 11:01:46 Uhr | 4 | ja | Business Process Management\|Business Process Modelin | |
| 8 | 1033 | 14 Sekunden | 22.01.2019, 11:07:18 Uhr | 1 | nein | | |
| 9 | 1034 | 45 Sekunden | 22.01.2019, 11:21:11 Uhr | 3 | ja | Business Process Management | Gut, der Übungsleiter ist nett und konnte uns Studenten |
| 10 | 1035 | 9 Sekunden | 22.01.2019, 11:21:45 Uhr | 2 | nein | Datenbanksysteme - Konzepte und Modelle | - |
| 11 | 1036 | 26 Sekunden | 22.01.2019, 11:23:46 Uhr | 4 | ja | Business Process Management\|Business Process Intellige | Nicht so gut. Musste mir alles selber beibringen. |
| 12 | 1037 | 50 Sekunden | 22.01.2019, 11:28:37 Uhr | 6 | ja | Business Process Management\|Datenbanksysteme - Kon | Guter Übungsbetrieb, Man wird hervorragend betreut. Du |

Abbildung 6.18: Exceltabelle zur Seite 1

### 6.2.6 Benutzer verwalten

Ein Administrator besitzt Zugang zur Benutzerverwaltung. Diese wird in Abbildung 6.19 vorgestellt.

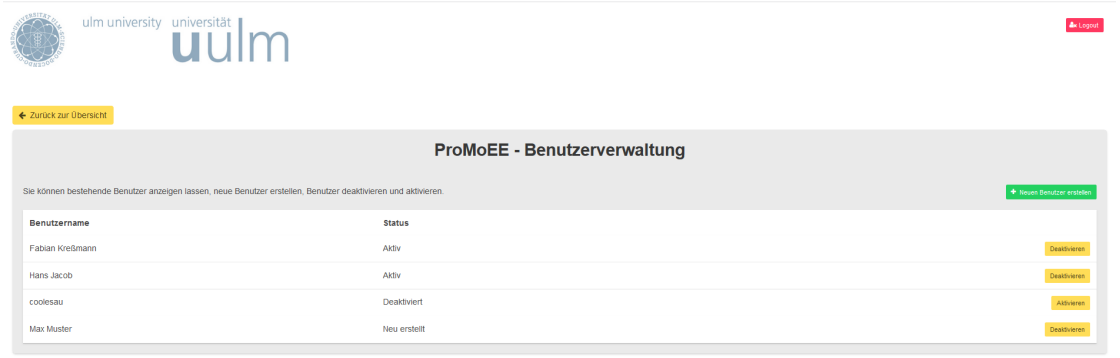



Abbildung 6.19: Benutzerverwaltung

Hier werden Benutzer verwaltet und neue Benutzer vom Administrator erstellt. Alle Benutzer werden in dieser Verwaltung aufgelistet, jedoch keine weiteren Administratoren. Benutzer sind aktivierbar und deaktivierbar. Deaktivierte Benutzer sind nicht in der

Lage, sich einzuloggen. Beim Erstellen eines neuen Benutzers legt der Administrator den Benutzernamen fest und das zugehörige Passwort. Dieser Benutzer wird als ”Neu erstellt” angezeigt. Ein neu erstellter Benutzer muss nach dem ersten Einloggen sein Passwort ändern und ein eigenes setzen. Abbildung 6.20 zeigt das Formular zum Erstellen eine neuen Benutzers.

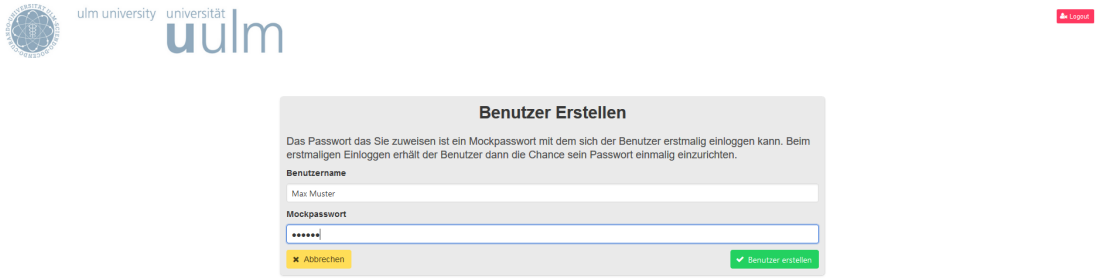



Abbildung 6.20: Neuen Benutzer anlegen

# 7

# Anforderungsabgleich

In diesem Kapitel werden die gestellten Anforderungen von Kapitel 4 mit der vorgestellten Implementierung aus Kapitel 6 abgeglichen. Dieses Kapitel wird in den Abgleich der funktionalen Anforderungen im Abschnitt 7.1 und der nicht-funktionalen Anforderungen im Abschnitt 7.2 unterteilt.

## 7.1 Funktionale Anforderungen

In Tabelle 7.1 werden die funktionalen Anforderungen aus Kapitel 4.2 mit der Implementierung abgeglichen. Anschließend wird im Detail auf die jeweiligen Anforderungen eingegangen.

| Kürzel | Bezeichnung | Erfüllt |
|---|---|---|
| FA-01 | Code absenden | ✓ |
| FA-02 | Fragebogen ausfüllen | ✓ |
| FA-03 | Fragebogen absenden | ✓ |
| FA-04 | Benutzer Login | ✓ |
| FA-05 | Fragebögen anzeigen | ✓ |
| FA-06 | Fragebogen erstellen | ✓ |
| FA-07 | Fragebogen bearbeiten | - |

| | | |
|---|---|---|
| FA-08 | Fragebogen löschen | - |
| FA-09 | Ausgefüllte Fragebögen anzeigen | ✓ |
| FA-10 | Fragebogen evaluieren | ✓ |
| FA-11 | Benutzer verwalten | ✓ |
| FA-12 | Benutzer anlegen | ✓ |
| OFA-01 | Fragebogendauer | ✓ |
| OFA-02 | Feedback | ✓ |
| OFA-03 | Modelldaten | ✓ |
| OFA-04 | Modell Replay | x |
| OFA-05 | Excel Export | ✓ |

Tabelle 7.1: Abgleich der funktionalen Anforderungen

**FA-01 Code absenden**

Umgesetzt. Es ist möglich mithilfe eines passenden Codes den entsprechenden Fragebogen zu erhalten.

**FA-02 Fragebogen ausfüllen**

Umgesetzt. Erhaltene Fragebögen können ausgefüllt werden. Es ist möglich zwischen den verschiedenen Seiten des Fragebogens zu navigieren und BPMN-Modelle zu erstellen.

**FA-03 Fragebogen absenden**

Umgesetzt. Der Benutzer bestätigt das Senden der Daten zum Server. Diese werden dann entsprechend abgespeichert.

**FA-04 Benutzer Login**

Umgesetzt. Mithilfe von Benutzername und Passwort ist das Einloggen für einen Benutzer in den Mitarbeiterbereich möglich.

**FA-05 Fragebögen anzeigen**

Umgesetzt. Nach dem Login erhält ein Benutzer eine Übersicht über alle von ihm erstellten Fragebögen. Die einzelnen Fragebögen sind zugreifbar, wodurch eine Übersicht über diesen Fragebogen und weitere Interaktionsmöglichkeiten ermöglicht wird.

**FA-06 Fragebogen erstellen**

Umgesetzt. Ein Benutzer besitzt die Möglichkeit, die verschiedenen Seiten und Fragen in einer Baumstruktur anzulegen. Anschließend erhält er eine Vorschau, wie der von ihm erstellte Fragebogen aussehen wird. Der Benutzer besitzt die Entscheidung, ob er den Fragebogen weiter bearbeiten will oder den Fragebogen zum Server sendet um diesen zu erstellen.

**FA-07 Fragebogen bearbeiten**

Teilweise Umgesetzt. Um potentiellen Datenverlust zu vermeiden und Datenintegrität zu bewahren, sind Fragebögen nur veränderbar, wenn noch keine ausgefüllten Fragebögen zu diesem Fragebogen existieren. Die verwendete Benutzeroberfläche ist gleiche wie beim Erstellen von Fragebögen. Es ist jederzeit möglich, Fragebögen kopieren und neu anzulegen, wodurch eine indirekte Veränderbarkeit und Versionierung von Fragebögen ermöglicht wird.

**FA-08 Fragebogen löschen**

Teilweise umgesetzt. Auch hier ist das Löschen von Fragebögen nur möglich, wenn noch keine Antwort zu diesen Fragebogen gegeben wurde.

**FA-09 Ausgefüllte Fragebögen anzeigen**

Umgesetzt. Ein Benutzer besitzt die Möglichkeit zu seinem Fragebogen einen ausgefüllten Fragebogen anzeigen lassen. Diese Sicht ist identisch mit der Sicht, die die ausfüllende Person beim Ausfüllen des Fragebogens gesehen hat.

**FA-10 Fragebogen evaluieren**

Umgesetzt. Einzelne Fragen und Modellieraufgaben sind evaluierbar. Es werden sämtliche getätigte Antworten und erstellte Modelle auf einer Seite angezeigt.

**FA-11 Benutzer verwalten**

Umgesetzt. Ein Administrator kann Benutzer verwalten. Er erhält eine Übersicht über alle Benutzer und ist in der Lage, diese zu aktivieren und zu deaktivieren.

**FA-12 Benutzer anlegen**

Umgesetzt. Ein Admin kann einen Benutzer mit Benutzernamen und Mock-Passwort erstellen. Der Benutzer meldet sich mit dem Mock-Passwort an und erhält anschließend die Möglichkeit, sein Passwort zu ändern. Eine Möglichkeit sein Passwort zu einem späteren Zeitpunkt zu ändern gibt es nicht.

**OFA-01 Fragebogendauer**

Umgesetzt. Das Ausfülldatum und Zeitpunkt eines Fragebogens wird auf dem Server gespeichert (Wenn der Fragebogen zum Server gesendet wird). Zusätzlich wird die benötigte Zeit für jede Fragebogenseite aufgezeichnet und gespeichert.

**OFA-02 Feedback**

Umgesetzt. Am Ende jedes Fragebogens erhält ein Benutzer die Möglichkeit, Feedback abzugeben. Das Feedback ist beim Anzeigen des ausgefüllten Fragebogens einsehbar.

**OFA-03 Modelldaten**

Umgesetzt. Verschiedene Daten zu einem Modell werden automatisch anhand des Modells ermittelt. Dies umfasst die Anzahl verschiedener verwendeter Elemente, die Beschriftung (prozentualer Anteil von beschrifteten Aktivitäten), Schönheit (Verlauf von Sequenzflüssen sollte möglichst grade sein) und der Workflow eines Modells (Elemente sollten eine korrekte Anzahl von eingehenden und ausgehenden Sequenzflüssen besitzen).

**OFA-04 Modell Replay**

Nicht umgesetzt. Auf dieses Feature wurde verzichtet, da pro Replayschritt das jeweilige Modell gespeichert werden müsste. Da das Modell den Hauptanteil des benötigten Speichers auf der Datenbank benötigt, würde sich die Datenbankgröße nur für dieses Anforderung um ein vielfaches vergrößern. Dies würde auf lange Sicht zu Performance Einbußen führen.

**OFA-05 Excel Export**

Umgesetzt. Es ist möglich eine Exceldatei zu einem Fragebogen zu erstellen. In dieser Exceldatei werden alle Frage und Antworten zu dem entsprechenden Fragebogen aufgelistet. Die Seiten werden auf Exceltabellen aufgeteilt. Modelle werden nicht angezeigt, jedoch die jeweiligen Modelldaten zu den Modellen.

## 7.2 Nicht-Funktionale Anforderungen

Alle nicht-funktionalen Anforderungen aus Kapitel 4.3 wurden umgesetzt. Im Nachfolgendem wird die Umsetzung dieser Anforderungen im System beleuchtet und begründet.

**NFA-01 Verfügbarkeit**

Die Webseite ist über den Server verfügbar. Solange der Server läuft und es keine Probleme mit der Hardware gibt, sollten sämtliche Anfragen den Server erreichen. *Dropwizard* ermöglicht das parallele Verarbeiten von mehreren tausend Anfragen (abhängig vom Server) gleichzeitig [33].

**NFA-02 Zuverlässigkeit**

*Dropwizard* ist auf langen Serverbetrieb ausgelegt. In Tests gingen Pakete nie verloren und der Server antwortete immer in der erwarteten Art und Weise.

**NFA-03 Intuitive Bedienbarkeit**

Die Benutzeroberfläche ist simple und selbsterklärend. Interaktionsmöglichkeiten sind ersichtlich und jede Seite zeigt einen erklärenden Text über die Funktionsweise der Seite an.

**NFA-04 Einheitliches Aussehen**

*Bulma* sorgt für ein einheitliches Aussehen auf der gesamten Webseite. Das Aussehen wurde bewusst schlich gehalten und bei den verwendeten Farben wurde sich stark an der Webseite der Universität Ulm und am Bulma Framework gehalten. Ähnliche Seiten sind gleich aufgebaut, zum Beispiel bei Auflistungen oder das Ansehen von Fragebögen (Ausfüllen, Vorschau, Evaluieren).

**NFA-05 Vorhersehbares Verhalten**

Alle Interaktionsmöglichkeiten sind beschriftet. Beim Abbrechen von Tätigkeiten die zu Datenverlust führen wird der Benutzer vorher um Bestätigung gebeten. Das System und der Quellcode ist verständlich dokumentiert.

**NFA-06 Leistungsfähigkeit**

*Dropwizard* verspricht multithreaded Session und zuverlässige Antwortzeiten [18] und die verwendete Datenbank *H2* ist besonders auf Performance ausgelegt [32]. Bei der Wahl der Frameworks und der Struktur des Projekts wurde besonders auf die Performance und Antwortzeiten geachtet. Die Größe der Datenbank soll so klein wie möglich bleiben um auch bei längeren Betrieb schnelle Antwortzeiten zu garantieren.

**NFA-07 Sicherheit**

Das System ermöglicht eine HTTPS-Verbindung um Pakete verschlüsselt über das Internet zu versenden. Es wird eine zweischichtige Authentifizierung verwendet. Die Anmeldung verläuft über *Basic Authentication*. Es wird ein JSON Web Token (JWT) generiert, mit dem sich der Benutzer im System über die *Bearer Token Authentication* schnell authentifizieren kann. Die auf der Datenbank hinterlegten Passwörter werden über *BCrypt* verschlüsselt.

**NFA-08 Datenschutz**

Angemeldete Benutzer (mit Ausnahme von Administratoren) haben nur Zugriff auf eigene Fragebögen. Die Daten liegen somit in der Verantwortung des Fragebogenerstellers. Alle Angaben im Fragebogen sind freiwillig und können übersprungen werden. Der Zugriff auf Daten von außen ist ohne Authentisierung nicht möglich.

**NFA-09 Datenintegrität**

Das Löschen und Bearbeiten von Daten ist im System nur eingeschränkt möglich. Antworten können nicht gelöscht werden. Sämtliche erhaltene Daten werden im System auf Validität überprüft. Besonders die Struktur von Fragebögen muss bestimmten Vorgaben entsprechen.

**NFA-10 Skalierbarkeit**

Das System ist auf langen Betrieb ausgelegt und gleichzeitig sehr leichtgewichtig. Größere Fragebögen können schnell verarbeitet werden.

**NFA-11 Stabilität**

Durch interne Validierung und Fehlerprüfung bleibt das System konsistent. Aufgetretene Fehler werden dem Anwender mitgeteilt und dokumentiert.

**NFA-12 Robustheit**

Es ist nicht möglich, über die Benutzeroberfläche fehlerhafte Daten zum Server zu schicken. Dies wurde durch eingehende Tests überprüft. Bei fehlerhaften Eingaben wird der Benutzer informiert.

**NFA-13 Erweiterbarkeit**

Das System ist durch die verwendeten Frameworks sehr leichtgewichtig und übersichtlich. Durch eine gute Dokumentation wird das Einarbeiten im System erleichtert.

**NFA-14 Portierbarkeit**

Zum Installieren des Servers wird nur Java benötigt. Der Server ist auf eine .jar Datei komprimiert und eine Installationsanleitung führt den Benutzer durch die benötigten Schritte.

# 8

# Zusammenfassung

Das in dieser Masterarbeit entstandene System *ProMoEE* bietet eine Web-Plattform für das Durchführen von Studien im Kontext von Business Process Modeling an. Es ist möglich, Fragebögen zu erstellen, auszufüllen und auszuwerten. Diese Fragebögen bieten zusätzlich eine Modellierungsumgebung für Geschäftsprozesse an. Das ermöglicht empirische Studien mit dem Ziel, die Verständlichkeit von Modellen zu steigern und den Modelliervorgang zu verbessern. Es ist ebenfalls denkbar den Übungsbetrieb für entsprechende Vorlesungen über diese Plattformen ablaufen zu lassen.

Das System erfüllt alle funktionalen und nicht-funktionalen Anforderungen aus Kapitel 4. Es wurde darauf geachtet, das System leichtgewichtig, erweiterbar und performant zu halten. Kapitel 5 bietet einen detaillierten Einblick zum Aufbau des Systems an. Das System wird in Kapitel 6 vorgestellt. Durch verschiedene Evaluationsmöglichkeiten lassen sich Fragebögen eingehend evaluieren. Als Auswertungsmöglichkeit stehen das Betrachten von ausgefüllten Fragebögen, das Anzeigen von allen Antworten zu einer Frage, das Evaluieren von Modellen und das Erstellen einer Excel-Datei mit allen getätigten Antworten und ausgewerteten Modelle zu einen Fragebogen zur Verfügung. Das System ist durch leichtgewichtige Frameworks und guter Dokumentation einfach um weitere Funktionen erweiterbar. *ProMoEE* bietet ein gutes Fundament für die Durchführung von empirischen Studien zum Modellierungsvorgang von Geschäftsprozessen.

## 8.1 Ausblick

Die in dieser Arbeit vorgegebenen funktionalen Anforderungen wurden erfüllt. Es sind im Laufe der Arbeit weitere funktionale Anforderungen entstanden, die in Abschnitt 4.2

als zusätzliche Anforderungen umgesetzt wurden. In diesen Abschnitt werden weitere denkbare Erweiterungen für das System beschrieben.

Die bereits angesprochene Möglichkeit Modelle Schritt für Schritt abzuspielen, wurde aufgrund von Performance verworfen, da für jeden Modellschritt ein zusätzliches Modell gespeichert werden muss. Es wäre beispielsweise möglich die Modellschritte auf eine bestimmte Anzahl zu begrenzen. Eine Implementierung einer Änderungshistorie des Modells ist ebenfalls denkbar.
Es ist für einen Benutzer nur möglich, auf eigene Fragebögen zuzugreifen. Es wäre eine Freigabefunktion wie aus *PEx* nützlich, mit der Benutzer auch auf andere Fragebögen zugreifen können um Ergebnisse besser abzusprechen.
Momentan ist die Benutzerverwaltung rudimentär gehalten. Die Möglichkeit Passwörter zu ändern ist nur bedingt gegeben. Die Möglichkeit, verlorene Passwörter wiederherzustellen ist nicht gegeben. Eine bessere Benutzerverwaltung über E-Mail Validierung ist sinnvoll.
Im Hinblick auf die Performance können bei vielen ausgefüllten Fragebögen die Ladezeiten für das Darstellen von Antworten zunehmen, da alle Antworten ungefiltert angezeigt werden. Eine Filterfunktion wäre sinnvoll um dieses Problem zu beheben, beispielsweise die Anzeige auf hundert Antworten zu beschränken.
Es ist durchaus denkbar die Anzeige und Auswertung von Modellen zu erweitern, beispielsweise um weitere Aspekte in der Tabelle mit prozentualen Fortschritt. Die Möglichkeit, Modelle als Bild-Dateien herunterzuladen wäre ebenfalls sinnvoll.
Momentan ist das System darauf ausgelegt, dass keine Daten verloren gehen. Bei großen Datenmengen wäre eine Löschmöglichkeit eventuell sinnvoll. Es wäre denkbar, das Löschen von Fragebögen grundsätzlich zu ermöglichen, und zudem selektiv Antworten löschen zu können.
Die Fragebögen sind um verschiedenen Aspekte erweiterbar. Neben weiteren Antwortmöglichkeiten, wie beispielsweise Zahleneingaben oder Slider, ist auch eine Aufteilung von Fragebögen denkbar. Es ist denkbar die Fragebogenseiten aus möglichen Alternativen bestehen zu lassen, welche zufällig ausgewählt werden. Dies ermöglicht das Durchführen von Fallstudien bestehend aus Testgruppen und Kontrollgruppen. Das ist momentan nur über das Kopieren von Fragebögen eingeschränkt möglich.

# Literaturverzeichnis


[1] Jeston, J.: Business process management. Routledge (2014)

[2] Scholl, A.: Die Befragung. Volume 2413. Utb (2014)

[3] Maurer, M., Jandura, O.: Masse statt Klasse? Einige kritische Anmerkungen zu Repräsentativität und Validität von Online-Befragungen. In: Sozialforschung im Internet. Springer (2009) 61–73

[4] Gutermuth, J.: Konzeption und Implementierung einer webbasierten Anwendung zur Durchführung von Modellierungsexperimenten. (2016)

[5] Pinggera, J., Zugal, S., Weber, B.: Investigating the Process of Process Modeling with Cheetah Experimental Platform–Tool Paper–. ER-POIS 2010 **13** (2010)

[6] Ulm University: 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/` (2013) letzter Aufruf: 28.01.2019.

[7] Scherle, S.: Konzeption und Evaluierung einer domänenspezifischen Modellierungsumgebung für prozessorientierte Fragebögen. (2014)

[8] Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: Empirical insights and recommendations. Information Systems **35** (2010) 467–482

[9] Friedrichs, J.: Methoden empirischer Sozialforschung. Springer-Verlag (1990)

[10] Mayer, H.O.: Interview und schriftliche Befragung: Grundlagen und Methoden empirischer Sozialforschung. Walter de Gruyter (2013)

[11] statista: Anteil der Internetnutzer nach Altersgruppen in Deutschland im Jahr 2014. `https://de.statista.com/statistik/daten/studie/216710/umfrage/internetnutzer-nach-altersgruppen-in-deutschland/` (2014) letzter Aufruf: 24.02.2019.

[12] ADM: Jahresbericht 2017. https://www.adm-ev.de/wp-content/uploads/2018/08/ADM_Jahresbericht_2017_Web-6.pdf (2017) letzter Aufruf: 24.02.2019.

[13] SurveyMonkey: SurveyMonkey: The Worlds Most Popular Free Online Survey Tool. https://www.surveymonkey.com/ (1999) letzter Aufruf: 24.02.2019.

[14] Survio: Online Survey Software; Create Free and Beautiful Survey; Survio.com. https://www.survio.com/ (2012) letzter Aufruf: 24.02.2019.

[15] Doodle: Doodle. https://doodle.com (2007) letzter Aufruf: 24.02.2019.

[16] LimeSurvey GmbH: LimeSurvey: The online survey tool. https://www.limesurvey.org/ (2003) letzter Aufruf: 24.02.2019.

[17] Allweyer, T.: BPMN 2.0: introduction to the standard for business process modeling. BoD–Books on Demand (2016)

[18] Yammer Inc: Dropwizard. https://www.dropwizard.io/1.3.8/docs/ (2012) letzter Aufruf: 24.02.2019.

[19] Apache Maven Project: Maven. https://maven.apache.org/ (2002) letzter Aufruf: 24.02.2019.

[20] Google: Guice. https://github.com/google/guice/wiki/GettingStarted (2006) letzter Aufruf: 24.02.2019.

[21] The Eclipse Foundation: EclipseLink. https://www.eclipse.org/eclipselink/ (2015) letzter Aufruf: 24.02.2019.

[22] Google: Guice Persist. https://github.com/google/guice/wiki/JPA (2014) letzter Aufruf: 24.02.2019.

[23] Morling, Gunna: mapstruct. http://mapstruct.org/ (2015) letzter Aufruf: 24.02.2019.

[24] leonate: JXLS. http://jxls.sourceforge.net/ (2005) letzter Aufruf: 24.02.2019.

[25] Camunda: Camunda BPM Platform. https://docs.camunda.org/manual/7.8/user-guide/model-api/bpmn-model-api/read-a-model/ (2005) letzter Aufruf: 24.02.2019.

[26] Facebook: React: A JavaScript library for building user interfaces. https://reactjs.org/ (2011) letzter Aufruf: 24.02.2019.

[27] Microsoft: Typescript: JavaScript that scales. https://www.typescriptlang.org/ (2012) letzter Aufruf: 24.02.2019.

[28] Abramoc, Dane: Redux: A predictable state container for JavaScript apps. https://redux.js.org/ (2015) letzter Aufruf: 24.02.2019.

[29] Camunda: bpmn-js. https://github.com/bpmn-io/bpmn-js (2014) letzter Aufruf: 24.02.2019.

[30] Thomas, Jeremy: Bulma. https://bulma.io/ (2016) letzter Aufruf: 24.02.2019.

[31] Masse, M.: REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. O'Reilly Media, Inc. (2011)

[32] The HSQLDB Group: H2 Database Engine. http://www.h2database.com/html/main.html (2005) letzter Aufruf: 24.02.2019.

[33] Babcock, N.: Turning Dropwizard Performance up to Eleven. https://nbsoftsolutions.com/blog/turning-dropwizard-performance-up-to-eleven (2016) letzter Aufruf: 24.02.2019.

# A

# Anhang Quellcode

Listing A.1: Klasse *QuestionnaireMapper* zur Transformation von Objekten, welche Fragebögen auf unterschiedlicher Art repräsentieren

```
1  /**
2   * Mapper used to map objects of questionnaires, its pages and
       questions between
3   * model (used for the database) and transfer objects (used for
       communication)
4   */
5  @Mapper(unmappedTargetPolicy = ReportingPolicy.IGNORE)
6  public interface QuestionnaireMapper {
7
8    QuestionnaireMapper INSTANCE = Mappers.getMapper(
         QuestionnaireMapper.class);
9
10   @Mapping(target = "questionnaireId", source = "id")
11   QuestionnaireResponse map(Questionnaire model);
12
13   @Mapping(target = "pageId", source = "id")
14   QuestionnairePageResponse map(QuestionnairePage model);
15
16   @Mapping(target = "questionId", source = "id")
17   QuestionResponse map(Question model);
18
19   @Mapping(target = "questionnaireId", source = "id")
20   @Mapping(target = "userName", source = "model.user.name")
21   QuestionnaireOverviewListItemResponse mapList(Questionnaire model);
```

```
22
23    Questionnaire map(ModifyQuestionnaireRequest model);
24    QuestionnairePage map(ModifyQuestionnairePageRequest model);
25    Question map(ModifyQuestionRequest model);
26  }
```

Listing A.2: Klasse *ExcelCreator* zur Erstellung und Rückgabe einer Exceldatei

```
1  /**
2   * Class handling the creation of an excel file using a given list of
        completed questionnaires.
3   */
4  public class ExcelCreator {
5
6    private final String questionnaireTitle;
7    private final String questionnaireCode;
8    private final String userName;
9    private final List<ExcelPage> pages;
10
11   /*
12    * Sets different variables representing different informations
        about the questionnaire. Only questionnaires
13    * that have already been completed can have an excel file created.
         Creates a List of ExcelPage class,
14    * each page being a separated sheet on the excel file.
15    */
16   public ExcelCreator(List<CompletedQuestionnaire> completedList) {
17     if (completedList.size() == 0) {
18       throw new ApplicationException("Der Fragebogen besitzt noch
            keine beantworteten Frageb\u00f6gen.", Response.Status.
            NOT_FOUND.getStatusCode());
19     }
20     questionnaireTitle = completedList.get(0).getQuestionnaire().
          getTitle();
```

```
21      questionnaireCode = completedList.get(0).getQuestionnaire().
            getCode();
22      userName = completedList.get(0).getQuestionnaire().getUser().
            getName();
23      List<ExcelPage> unsortedPages = new ArrayList<ExcelPage>();
24      for (QuestionnairePage qp : completedList.get(0).getQuestionnaire
            ().getQuestionnairePages()) {
25        unsortedPages.add(new ExcelPage(completedList, qp));
26      }
27      pages = unsortedPages.stream()
28      .sorted(Comparator.comparing(ExcelPage::getPageNumber))
29      .collect(Collectors.toList());
30    }
31
32    /*
33     * Creates the excel file using jxls, filling the sheet with
           variables of this class and returning an array of bytes
34     * containing the resulting excel file.
35     */
36    public byte[] createExcelFile() {
37
38      try(InputStream is = ExcelCreator.class.getResourceAsStream("/
            excel-templates/Template.xlsx")) {
39          try (ByteArrayOutputStream result = new ByteArrayOutputStream
                ()) {
40              Context context = new Context();
41              context.putVar("questionnaireTitle", questionnaireTitle);
42              context.putVar("questionnaireCode", questionnaireCode);
43              context.putVar("userName", userName);
44              context.putVar("pages", pages);
45              context.putVar("sheetNames", pages.stream().map(iter -> "
                    Seite " + iter.getPageNumber()).collect(Collectors.
                    toList()));
46              JxlsHelper.getInstance().setUseFastFormulaProcessor(false)
47              .processTemplateAtCell(is, result, context, "A1");
```

```
48              return result.toByteArray();
49          }
50      } catch (IOException e) {
51          throw new ApplicationException("Beim Erstellen der Excel-Datei
                ist ein Fehler aufgetreten.", Response.Status.
              INTERNAL_SERVER_ERROR.getStatusCode());
52      }
53    }
54  }
```

Listing A.3: Methode *getAnswerRowForModel* zur Analyse eines BPMN-Prozessmodells

```
1   /**
2     * Function to analyze a given model, retrieving multiple
        informations about the model like number
3     * of elements used, the percentage of labeled activity's, pretty
        sequence flows and correctly connected elements.
4     * Camunda bpmn is used to easily analyze and retrieve bpmn objects
         from the model.
5     * Returns a list of Strings to represent the cell content of the
        row.
6     */
7    private List<String> getAnswerRowFromModel(String model) {
8      if(Strings.isNullOrEmpty(model)) {
9        return Arrays.asList("Modell wurde nicht bearbeitet", "-", "-",
              "-", "-", "-", "-", "-");
10     }
11     // Get all instances of different types of elements used in the
          model
12     BpmnModelInstance modelInstance = Bpmn.readModelFromStream(new
          ByteArrayInputStream(model.getBytes(StandardCharsets.UTF_8)));

13     Collection<ModelElementInstance> activityInstances =
          modelInstance
```

```
14      .getModelElementsByType(modelInstance.getModel().getType(Activity
            .class));
15      Collection<ModelElementInstance> gatewayInstances = modelInstance
16      .getModelElementsByType(modelInstance.getModel().getType(Gateway.
            class));
17      Collection<ModelElementInstance> eventInstances = modelInstance
18      .getModelElementsByType(modelInstance.getModel().getType(Event.
            class));
19      Collection<ModelElementInstance> sequenceInstances =
            modelInstance
20      .getModelElementsByType(modelInstance.getModel().getType(
            SequenceFlow.class));
21      Collection<ModelElementInstance> annotationInstances =
            modelInstance
22      .getModelElementsByType(modelInstance.getModel().getType(
            TextAnnotation.class));
23
24      // Calculate the degree of labeled activitys in percentage
25      int numberOfLabels = 0;
26      for (ModelElementInstance activity : activityInstances) {
27        if (!Strings.isNullOrEmpty(activity.getAttributeValue("name")))
              numberOfLabels++;
28      }
29      String labelPercentage;
30      if (activityInstances.size() == 0) {
31        labelPercentage = "-";
32      } else {
33        labelPercentage = String.valueOf((numberOfLabels * 100) /
             activityInstances.size()) + "%";
34      }
35
36      // Calculate the degree of pretty sequence flows (straight
            vertical and horizontal, only having one turn) in percentage
            using waypoints of the bpmn edges.
37      Collection<ModelElementInstance> edgeInstances = modelInstance
```

```
38      .getModelElementsByType(modelInstance.getModel().getType(BpmnEdge
            .class));
39      int goodFlows = 0;
40      for (ModelElementInstance edge : edgeInstances) {
41        if(!edge.getAttributeValue("id").contains("SequenceFlow"))
              continue;
42        Collection<Waypoint> waypoints = edge.getChildElementsByType(
              Waypoint.class);
43        if(waypoints.size() < 4 && (waypoints.stream().map(Waypoint::
              getX).distinct().count() + waypoints.stream().map(Waypoint::
              getY).distinct().count()) == (waypoints.size() + 1)) {
44          goodFlows++;
45        }
46      }
47      String goodFlowsPercentage;
48      if (sequenceInstances.size() == 0) {
49        goodFlowsPercentage = "-";
50      } else {
51        goodFlowsPercentage = String.valueOf((goodFlows * 100) /
              sequenceInstances.size()) + "%";
52      }
53
54      // Calculate the degree of well connected elements in the model.
55      int goodConnections = 0;
56      // An activity should only have one incoming and one outgoing
            flow
57      for (ModelElementInstance activity : activityInstances) {
58        Collection<Outgoing> outgoing = activity.
              getChildElementsByType(Outgoing.class);
59        Collection<Incoming> incoming = activity.
              getChildElementsByType(Incoming.class);
60        if (outgoing.size() == 1 && incoming.size() == 1)
              goodConnections++;
61      }
```

```
62      // Gateway can have one incoming and multiple outgoing flows, or
            vice versa.
63      for (ModelElementInstance gateway : gatewayInstances) {
64        Collection<Outgoing> outgoing = gateway.getChildElementsByType
              (Outgoing.class);
65        Collection<Incoming> incoming = gateway.getChildElementsByType
              (Incoming.class);
66        if ((outgoing.size() > 1 && incoming.size() == 1) || (outgoing.
              size() == 1 && incoming.size() > 1)) goodConnections++;
67      }
68      // An event can have depending on its type up to one incoming and
            up to one outgoing flow
69      for (ModelElementInstance event : eventInstances) {
70        if(event.getElementType() instanceof StartEvent) {
71          Collection<Outgoing> outgoing = event.getChildElementsByType
                (Outgoing.class);
72          Collection<Incoming> incoming = event.getChildElementsByType
                (Incoming.class);
73          if (outgoing.size() == 1 && incoming.size() == 0)
                goodConnections++;
74        } else if (event.getElementType() instanceof EndEvent) {
75          Collection<Outgoing> outgoing = event.getChildElementsByType
                (Outgoing.class);
76          Collection<Incoming> incoming = event.getChildElementsByType
                (Incoming.class);
77          if (outgoing.size() == 0 && incoming.size() == 1)
                goodConnections++;
78        } else {
79          Collection<Outgoing> outgoing = event.getChildElementsByType
                (Outgoing.class);
80          Collection<Incoming> incoming = event.getChildElementsByType
                (Incoming.class);
81          if (outgoing.size() == 1 || incoming.size() == 1)
                goodConnections++;
82        }
```

```
83      }
84      int allElements = activityInstances.size() + gatewayInstances.
            size() + eventInstances.size();
85      String goodConnectionsPercentage;
86      if (allElements == 0) {
87        goodConnectionsPercentage = "-";
88      } else {
89        goodConnectionsPercentage = String.valueOf((goodConnections *
              100) / allElements) + "%";
90      }
91
92      return Arrays.asList(
93             String.valueOf(activityInstances.size()),
94             String.valueOf(gatewayInstances.size()),
95             String.valueOf(eventInstances.size()),
96             String.valueOf(sequenceInstances.size()),
97             String.valueOf(annotationInstances.size()),
98             labelPercentage,
99             goodFlowsPercentage,
100            goodConnectionsPercentage);
101   }
```

Listing A.4: Klasse *ApiApplication* für das Starten und Initialisieren des Dropwizard Servers

```
1  /**
2   * Dropwizard application class for starting and running the server.
        For more informations use the
3   * official dropwizard documentation
4   */
5  public class ApiApplication extends Application<ApiConfiguration> {
6
7    public static void main(String[] args) throws Exception {
8        new ApiApplication().run(args);
```

```
9     }
10
11    @Override
12    public void initialize(Bootstrap<ApiConfiguration> bootstrap) {
13
14      // Add a bundle for using the database
15      bootstrap.addBundle(new MigrationsBundle<ApiConfiguration>() {
16        public PooledDataSourceFactory getDataSourceFactory(final
              ApiConfiguration configuration) {
17          return configuration.getDataSourceFactory();
18        }
19      });
20      // Add a bundle for using the web application
21      bootstrap.addBundle(new AssetsBundle("/assets/", "/", "index.html
          ","static"));
22    }
23
24  @Override
25  public void run(ApiConfiguration configuration, Environment
      environment) throws Exception {
26    // Create a google guice injector and a jpa persistence module
        for accessing the database
27    Injector injector = Guice.createInjector(new ApiModule(),
        createJpaModule(configuration.getDataSourceFactory()));
28    environment.servlets().addFilter("persistFilter", injector.
        getInstance(PersistFilter.class))
29    .addMappingForUrlPatterns(EnumSet.of(REQUEST, FORWARD), true, "/
        api/*");
30
31    environment.jersey().setUrlPattern("/api/*");
32      environment.jersey().enable(ServerProperties.
          BV_SEND_ERROR_IN_RESPONSE);
33
34      // Register the BasicAuthenticator and JwtTokenAuthenticator
```

```
35      final AuthFilter<BasicCredentials, LoginResponse> basicFilter =
            new BasicCredentialAuthFilter.Builder<LoginResponse>()
36          .setAuthenticator(new BasicAuthenticator(injector.
                getInstance(UserService.class)))
37          .setPrefix("Basic")
38          .buildAuthFilter();
39      final AuthFilter<JwtContext, User> jwtFilter = new
            CustomJwtAuthFilter.Builder<User>()
40          .setJwtConsumer(createConsumer())
41          .setRealm("realm")
42          .setPrefix("Bearer")
43          .setAuthenticator(new JwtAuthenticator(injector.getInstance(
                UserService.class)))
44          .buildAuthFilter();
45      final PolymorphicAuthDynamicFeature<Principal> feature = new
            PolymorphicAuthDynamicFeature<>(
46          ImmutableMap.of(LoginResponse.class, basicFilter, User.class,
                jwtFilter));
47      final AbstractBinder binder = new
            PolymorphicAuthValueFactoryProvider.Binder<>(
48          ImmutableSet.of(LoginResponse.class, User.class));
49      environment.jersey().register(feature);
50      environment.jersey().register(binder);
51
52      // Register resources
53      environment.jersey().register(injector.getInstance(CodeResource.
            class));
54      environment.jersey().register(injector.getInstance(
            QuestionnaireResource.class));
55      environment.jersey().register(injector.getInstance(AnswerResource
            .class));
56      environment.jersey().register(injector.getInstance(UserResource.
            class));
57      environment.jersey().register(injector.getInstance(ExcelResource.
            class));
```

```
58
59      environment.healthChecks().register("APIHealthCheck", new
            ApiHealthCheck(injector.getInstance(UserService.class)));
60      environment.jersey().register(new ApplicationExceptionMapper());
61    }
62
63    /*
64     * Creates a jpa module connecting to the database using the given
           properties given by the config.yml
65     */
66    private JpaPersistModule createJpaModule(final DataSourceFactory
          dbConfig) {
67      final Properties properties = new Properties();
68      properties.put("javax.persistence.jdbc.driver", dbConfig.
            getDriverClass());
69      properties.put("javax.persistence.jdbc.url", dbConfig.getUrl());
70      properties.put("javax.persistence.jdbc.user", dbConfig.getUser()
            );
71      properties.put("javax.persistence.jdbc.password", dbConfig.
            getPassword());
72
73      final JpaPersistModule jpaModule = new JpaPersistModule("d1");
74      jpaModule.properties(properties);
75
76      return jpaModule;
77    }
78
79    /*
80     * Function to create a JwtConsumer used for validating tokens
81     */
82    private JwtConsumer createConsumer() {
83      return new JwtConsumerBuilder()
84              .setAllowedClockSkewInSeconds(30) // allow some leeway in
                    validating time based claims to account for clock skew
```

```
85              .setRequireExpirationTime() // the JWT must have an
                    expiration time
86              .setRequireSubject() // the JWT must have a subject claim
87              .setVerificationKey(new HmacKey(Constants.SECRET)) //
                    verify the signature with the public key
88              .setRelaxVerificationKeyValidation() // relaxes key length
                     requirement
89              .build(); // create the JwtConsumer instance
90    }
91 }
```

Listing A.5: Klasse *ApiModule* zur Erstellung und Binden von Klassenabhängikeiten über Guice

```
1  /**
2   * Class for the main module of this api, used to bind classes in the
        injector to retrieve
3   * them later on using the inject annotation.
4   */
5  public class ApiModule extends AbstractModule{
6
7    @Override
8    protected void configure() {
9
10     bind(CodeResource.class).in(Singleton.class);
11     bind(CodeService.class).in(Singleton.class);
12     bind(QuestionnaireResource.class).in(Singleton.class);
13     bind(QuestionnaireService.class).in(Singleton.class);
14     bind(AnswerResource.class).in(Singleton.class);
15     bind(AnswerService.class).in(Singleton.class);
16     bind(UserResource.class).in(Singleton.class);
17     bind(UserService.class).in(Singleton.class);
18     bind(ExcelResource.class).in(Singleton.class);
19     bind(ExcelService.class).in(Singleton.class);
```

```
20    }
21 }
```

Listing A.6: Klasse *ApplicationException* zur Definition von Antworten bei internen Serverfehlern

```
1  /**
2   * Exception being thrown on internal server errors, returning a error
        message and
3   * a status code to the requesting application using the
       ApllicationExceptionMapper
4   */
5  public class ApplicationException extends RuntimeException {
6
7    private static final long serialVersionUID = 5597675843854389 3L;
8
9    private final Integer status;
10
11   public ApplicationException(final String message, final Integer
         status) {
12     super(message);
13     this.status = status;
14   }
15
16   public Integer getStatus() {
17     return status;
18   }
19 }
```

Listing A.7: Klasse *ApplicationExceptionMapper* zur Verarbeitung von *ApplicationExceptions*

```
1  /**
```

```
2  * Mapper to return a Response upon throwing an ApplicationException
       detailing informations
3  * about the exception and a http status code
4  */
5  @Provider
6  public class ApplicationExceptionMapper implements ExceptionMapper<
       ApplicationException>{
7
8    private static final Logger LOGGER = LoggerFactory.getLogger(
         ApplicationException.class);
9
10   @Override
11   public Response toResponse(ApplicationException exception) {
12     LOGGER.warn(exception.getMessage());
13     return Response.status(exception.getStatus())
14             .entity(exception.getMessage())
15             .type(MediaType.TEXT_PLAIN)
16             .build();
17   }
18 }
```

Listing A.8: Klasse *Questionnaire* zur Repräsentation von Fragebögen in der Datenbank

```
1  /**
2   * Represents a questionnaire on the database
3   */
4  @Entity
5  @Table(name = "QUESTIONNAIRE")
6  @SequenceGenerator(name = "QUESTIONNAIRE_ID_SEQ", sequenceName = "
       QUESTIONNAIRE_ID_SEQ", allocationSize = 1)
7  @NamedQueries({
8     @NamedQuery(
9           name = "Questionnaire.findAll",
10          query = "SELECT q FROM Questionnaire q"),
```

```
11      @NamedQuery(
12              name = "Questionnaire.findByCode",
13              query = "SELECT q FROM Questionnaire q WHERE q.code = :code"
                    )
14  })
15  public class Questionnaire {
16
17      /*
18       * The unique id of the questionnaire used to identify it on the
             database. Auto increments upon
19       * saving it to the database using the sequence generator
20       */
21      @Id
22      @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "
            QUESTIONNAIRE_ID_SEQ")
23      @Column(name = "ID", updatable = false, nullable = false)
24      private Integer id;
25
26      /*
27       * The title of this questionnaire
28       */
29      @Column(name = "TITLE", nullable = false)
30      @Size(min= 3)
31      @NotNull
32      private String title;
33
34      /*
35       * The code of this questionnaire used to request it from the
             server
36       */
37      @Column(name = "CODE", nullable = false, unique = true)
38      @Size(min= 3, max = 50)
39      @NotNull
40      private String code;
41
```

```
42    /*
43     * The list of questionnaire pages this questionnaire has. Each
          page is persisted upon persisting
44     * this questionnaire and deleted upon deleting this questionnaire
45     */
46    @OneToMany(mappedBy = "questionnaire", cascade=CascadeType.ALL,
          orphanRemoval = true)
47     private Set<QuestionnairePage> questionnairePages;
48
49    /*
50     * The user who created this questionnaire
51     */
52    @ManyToOne
53    @JoinColumn(name = "USER_ID", nullable = false)
54    @NotNull
55    private User user;
56  }
```

Listing A.9: Klasse *QuestionnaireResponse* zur Repräsentation des mitgesendeten Fragebogens bei einer entsprechenden Serverantwort

```
1  /**
2   * Server response object for a questionnaire entity
3   */
4  public class QuestionnaireResponse {
5
6    @JsonProperty("questionnaireId")
7    private Integer questionnaireId;
8
9    @JsonProperty("title")
10   private String title;
11
12   @JsonProperty("code")
13   private String code;
```

```
14
15    @JsonProperty("questionnairePages")
16    private List<QuestionnairePageResponse> questionnairePages;
17 }
```

Listing A.10: Klasse *ModelerPage* zur Darstellung einer Modellierseite in einem Fragebogen

```
1  import Modeler from 'bpmn-js/lib/Modeler.js';
2  /**
3   * Due to the fact that bpmn.js doesnt support typescript, we had to
        deactivate the
4   * "no implicit any" feature of typescript
5   */
6  import * as React from 'react';
7  import { QuestionnairePage } from '../../state-management/
       questionnaire';
8  import { PageFooter } from '../views';
9
10 interface Params {
11    readonly page: QuestionnairePage;
12    readonly totalPages: number; // Indicates the total amount of
          pages
13    readonly current: number; // The currently viewed pagenumber
14
15    // Save the current model to state
16    readonly saveModel: (value: string, current: number, timeSpan:
          number) => void;
17    // Switch page
18    readonly setPage: (page: number) => void;
19 }
20
21 interface State {
```

```
22    // The total time the user was on this page in seconds, increments
          every second
23    timeSpan: number;
24 }
25
26 /**
27  * This component will render the BPMN-Modeler from bpmn.io. It is
        used for
28  * displaying a canvas for the user to freely model a bpmn-model if
        the pageType represents
29  * a MODELER. Has timeSpan as internal state
30  */
31 export class ModelerPage extends React.Component<Params, State> {
32    private modeler; // The modeler
33    private interval; // The interval dictating when the model is
          saved to the state
34    private timer; // Timer ticking every secounds to increment
          timeSpan
35
36    constructor(props: Params) {
37       super(props);
38
39       // Initialize timeSpan with the saved timeSpan from the state
40       this.state = {
41          timeSpan: this.props.page.timeSpan,
42       };
43
44       // Initialize Modeler
45       this.modeler = new Modeler({
46          width: '100%',
47          height: '400px',
48       });
49    }
50
51    public render() {
```

```
52
53        return (
54           <div>
55              <div className="box">
56                 <label className="label">
57                    {this.props.page.pageTitle}
58                 </label>
59                 <div id="model">
60                    <div id="canvas"/>
61                 </div>
62              </div>
63              <PageFooter
64                 current={this.props.current}
65                 totalPages={this.props.totalPages}
66                 prevPage={this.prevPage}
67                 nextPage={this.nextPage}
68                 disableLast={false}
69              />
70           </div>
71        );
72     }
73
74     public componentDidMount() {
75
76        // After mounting, it is save to attach the modeler to the html
              element
77        this.modeler.attachTo('#model');
78
79         // Import function for the model
80        function importXML(xml: any, bpmn: any) {
81
82           // Import diagram
83           bpmn.importXML(xml, (err: any) => {
84
85           if (err) {
```

```
86              return console.error('could not import BPMN 2.0 diagram',
                    err);
87          }
88
89          const canvas = bpmn.get('canvas');
90
91          // Zoom to fit full viewport, automatic zoom to best fit the
                model
92          canvas.zoom('fit-viewport');
93        });
94      }
95
96      // Is a model given? If not generate a default model
97      this.props.page.model ? importXML(this.props.page.model, this.
            modeler) : this.modeler.createDiagram();
98
99      // Every 10 secounds, save to model to the current state
100     this.interval = setInterval(this.saveDiagram, 10000);
101
102     // Every second increment timeSpan
103     this.timer = setInterval(() => this.setState(((prevState) => ({
104         ...prevState,
105         timeSpan: prevState.timeSpan + 1,
106       }))), 1000);
107   }
108
109   public componentWillUnmount() {
110     clearInterval(this.interval);
111     clearInterval(this.timer);
112   }
113
114   // Get to the previous page
115   private prevPage = () => {
116     this.props.setPage(this.props.current - 1);
117     this.saveDiagram();
```

```
118     }
119
120     // Get to the next page
121     private nextPage = () => {
122         this.props.setPage(this.props.current + 1);
123         this.saveDiagram();
124     }
125
126     // Save page to model
127     private saveDiagram = () => {
128
129         this.modeler.saveXML({ format: true }, (err, xml) => {
130             this.props.saveModel(xml, this.props.current, this.state.
                    timeSpan);
131         });
132     }
133 }
```

Listing A.11: LoginState zur Darstellung des Zustands für die Verwaltung von Login-Information

```
1  /**
2   * Represents all information about the user logged in, if the user
        is logged in
3   */
4  export interface LoginState {
5      // The name of the user currently logged in
6      readonly userName?: string;
7      // The access token to authenticate to the server as the current
           user
8      readonly token?: string;
9      // Indicates whether this user is an admin and has additional
           permissions
10     readonly isAdmin?: boolean;
```

```
11     // Indicates whether th user was recently created and has not yet
           changed his password
12     readonly isCreated?: boolean;
13     // Indicates whether current login request is already running
14     readonly loginLoading: boolean;
15     // The error text to display upon a failed login, if any
16     readonly loginFailedText?: string;
17 }
18
19 export const initialLoginState: LoginState = {
20     loginLoading: false,
21 };
```

Listing A.12: LoginAction zur Definition von Actions für die Verwaltung von Login-Information

```
1  import { checkResponse, LoginDto, parseJson } from '../../transfer';
2  import { AppDispatch } from '../AppDispatch';
3
4  /**
5   * Actions used in the context of logging in and out and managing the
        login state
6   */
7  export type LoginAction = LoginRequest | LoginResponse | LoginFailure
       | Logout | PasswordWasChanged;
8
9  interface LoginRequest {
10     readonly type: 'LOGIN_REQUEST';
11 }
12
13 function loginRequest(): LoginRequest {
14     return {
15         type: 'LOGIN_REQUEST',
16     };
```

```
}

interface LoginResponse {
   readonly type: 'LOGIN_RESPONSE';
   readonly dto: LoginDto;
}

function loginResponse(dto: LoginDto): LoginResponse {
   return {
      type: 'LOGIN_RESPONSE',
      dto,
   };
}

interface LoginFailure {
   readonly type: 'LOGIN_FAILURE';
   readonly cause: string;
}

function loginFailure(cause: string): LoginFailure {
   return {
      type: 'LOGIN_FAILURE',
      cause,
   };
}

/**
 * Indicates that the password was successfully changed. Upon loging
    in with a new user, if the isCreated flag is true
 * the current user can do no other actions than changing his
    password. On success the isCreated flag in the login state
 * has to be set to false, allowing the user further interactions
    with the system
 */
interface PasswordWasChanged {
```

```
49    readonly type: 'PASSWORD_WAS_CHANGED';
50 }
51
52 export function passwordWasChanged(): PasswordWasChanged {
53    return {
54       type: 'PASSWORD_WAS_CHANGED',
55    };
56 }
57
58 /**
59  * Log in to the server using the given credentials with basic
        authentication
60  */
61 export function login(userName: string, password: string) {
62    return (dispatch: AppDispatch) => {
63       dispatch(loginRequest());
64
65       const headers = new Headers();
66       headers.set('Authorization', 'Basic ' + Buffer.from(userName + '
             :' + password).toString('base64'));
67
68       return fetch('/api/user/login', {
69          headers,
70       })
71       .then(checkResponse())
72       .then(parseJson<LoginDto>())
73       .then (({json}) => {
74          dispatch(loginResponse(json));
75       })
76       .catch((response) => {
77          if (response.status === 401) {
78             response.text()
79             .then((text) => dispatch(loginFailure(
80                'Anmeldung fehlgeschlagen: ' + text)))
```

```
81              .catch(() => dispatch(loginFailure('Anmeldung
                    fehlgeschlagen.')));
82          } else {
83              response.text()
84              .then((text) => dispatch(loginFailure(
85                  'Bei der Verarbeitung der Anfrage ist ein Fehler
                        aufgetreten: ' + text)))
86              .catch(() => dispatch(loginFailure('Bei der Verarbeitung
                    der Anfrage ist ein Fehler aufgetreten.')));
87          }
88      });
89  };
90  }
91
92  /**
93   * Logs out the user from the system, simply clearing the login state
94   */
95  interface Logout {
96      readonly type: 'LOGOUT';
97  }
98
99  export function lougout(): Logout {
100     return {
101         type: 'LOGOUT',
102     };
103 }
104
105 /**
106  * Function that is being dispatches everytime, the authentication
        failed (statuscode 401) for requests using the access
107  * token, most likely meaning that the token has expired and the user
        has to relog to the system
108  */
109 export function invalidToken(dispatch: AppDispatch) {
110     dispatch(lougout());
```

```
111     dispatch(loginFailure('Ihre Session ist abgelaufen. Um Fortzufahren
            muessen sie sich neu einloggen'));
112 }
```

Listing A.13: LoginReducer für die Verarbeitung von Actions die den Zustand für die Verwaltung von Login-Information ändern

```
1  import { LoginAction } from './LoginAction';
2  import { initialLoginState, LoginState } from './LoginState';
3
4  /**
5   * Reducer for LoginActions to update the state given the previous
        state and the given action
6   */
7  export function loginReducer(
8     state = initialLoginState, action: LoginAction): LoginState {
9
10       switch (action.type) {
11          case 'LOGIN_REQUEST':
12             return {
13                ...state,
14                userName: undefined,
15                token: undefined,
16                isAdmin: undefined,
17                isCreated: undefined,
18                loginFailedText: undefined,
19                loginLoading: true};
20          case 'LOGIN_RESPONSE':
21             return {
22                ...state,
23                userName: action.dto.userName,
24                token: action.dto.token,
25                isAdmin: action.dto.isAdmin,
26                isCreated: action.dto.isCreated,
```

```
27                  loginLoading: false};
28          case 'LOGIN_FAILURE': return {...state, loginFailedText:
                action.cause, loginLoading: false};
29          case 'LOGOUT': return {
30              ...state,
31              userName: undefined,
32              token: undefined,
33              isAdmin: undefined,
34              isCreated: undefined,
35              loginFailedText: undefined};
36          case 'PASSWORD_WAS_CHANGED': return {...state, isCreated:
                false};
37      }
38      return state;
39  }
```

# Abbildungsverzeichnis

# Tabellenverzeichnis

Name: Fabian Kreßmann Matrikelnummer: 829963

**Erklärung**

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

Ulm, den .....

Fabian Kreßmann