Ruhr-Universität Bochum

Bachelorarbeit

**Entwicklung einer Plattform für datengetriebene Unterstützung von Interviews**

Schriftliche Abgabe der Abschlussprüfung im Studiengang „Angewandte Informatik“

Vorgelegt von

Dennis Fischer

Am Lehrstuhl für Informations- und Technikmanagement bei

Prof. Thomas Herrmann

Abgabedatum: 17.10.2020

Matrikelnummer: 108017220706

Betreuer/Prüfer: M. Sc. Felix Thewes

**Inhalt**

[1.Einleitung 1](#_Toc59030536)

[1.1 Einführung 2](#_Toc59030537)

[1.2 Relevanz & Ziel der Arbeit 3](#_Toc59030538)

[2.Literatur 4](#_Toc59030539)

[2.1 Heuristiken 4](#_Toc59030540)

[2.2 Usability & User Experience 5](#_Toc59030541)

[2.2.1 Usability 5](#_Toc59030542)

[2.2.2 User Experience 6](#_Toc59030543)

[2.3 Fragebögen (1) 6](#_Toc59030544)

[2.4 Evaluierung 7](#_Toc59030545)

[2.4.1 Heuristic Evaluation 7](#_Toc59030546)

[2.4.2 Cognitive Walkthrough 8](#_Toc59030547)

[2.4.3 Coaching Method 8](#_Toc59030548)

[3.Verwendete Technik (4-5) 9](#_Toc59030549)

[4.1 JavaScript/Typescript 9](#_Toc59030550)

[4.2 Angular 10](#_Toc59030551)

[4.2.1 Das Framework Angular 10](#_Toc59030552)

[4.2.2 Architektur von Angular 10](#_Toc59030553)

[4.3 Node und Express 12](#_Toc59030554)

[4.4 MongoDB 13](#_Toc59030555)

[4.5 Architektur: MEAN-Stack 13](#_Toc59030556)

[4.6 Bootstrap 14](#_Toc59030557)

[4.Entwurf & Konzeptionierung (5-9) 15](#_Toc59030558)

[5.1 Personas (3) 15](#_Toc59030559)

[5.2 Anforderungen (1-2) 19](#_Toc59030560)

[5.3 Use-Case-Diagramm (1-2) 23](#_Toc59030561)

[5.Implementierung (13-17) 24](#_Toc59030562)

[6.1 Datenbank (2-3) 25](#_Toc59030563)

[6.2 Backend (3-4) 27](#_Toc59030564)

[6.3 Frontend (8-10) 27](#_Toc59030565)

[8.Evaluierung: Anwendung (6-?) 28](#_Toc59030566)

[8.2 Durchführung von Methodik A (3-?) 28](#_Toc59030567)

[8.3 Durchführung von Methodik B (3-?) 28](#_Toc59030568)

[9.Ergebnisse (2-3) 28](#_Toc59030569)

[9.1 Vergleich der Resultate (2-3) 28](#_Toc59030570)

[10.Fazit (2) 28](#_Toc59030571)

[10.1 Ausblick (1) 28](#_Toc59030572)

[10.2 Folgerung (1) 28](#_Toc59030573)

[11. Literaturverzeichnis (1-2) 28](#_Toc59030574)

[12.Abbildungsverzeichnis (1-2) **Fehler! Textmarke nicht definiert.**](#_Toc59030575)

[13.Berufung (1) Seiten: 40- ca. 50+ 29](#_Toc59030576)

**Abbildungsverzeichnis**

[Abbildung 1 – Angular allgemeiner Aufbau von Component mit Service Quelle: https://angular.io/generated/images/guide/architecture/overview2.png (später richtig Quelle angeben) 12](#_Toc60270713)

[Abbildung 2 - Der MEAN-Stack im Überblick Quelle: https://kb.heaptrace.com/mean-stack-requirement 14](#_Toc60270714)

[Abbildung 3 - Ablauf der App im Flussdiagramm 24](#_Toc60270715)

[Abbildung 4- Datenbank: Schemata 26](#_Toc60270716)

# **1.Einleitung**

## **Einführung**

Durch die zunehmende Digitalisierung von Industriebereichen, können Betriebe immer flexibler auf Anforderungen innerhalb sowie außerhalb des Unternehmens reagieren. Aufgrund dieser Dynamik sind Vorgehensmodelle für Projekte meist durch Zyklen definiert, sodass diese nicht konstant halten. Dies hat zur Folge, dass sich Arbeitsbedingungen sowie Tätigkeiten von Mitarbeitern enorm und schnell wandeln können (Herrmann & Nierhoff, 2019). Insbesondere kann es daher häufig dazu kommen, dass Mitarbeiter mit ihnen bisher unbekannter Arbeit konfrontiert werden. Dies impliziert, dass die geforderten Tätigkeiten im Grunde nicht effizient, bis gar nicht ausgeführt werden können.

Da Unternehmen im schlimmsten Fall daher zeitliche sowie ressourcentechnische Verluste hinnehmen können, wäre es ratsam entsprechende Situationen vorzubeugen bzw. zu planen, um angemessen darauf zu reagieren.

Das methodische Vorgehen würde hierbei beispielsweise die Anwendung von Heuristiken anbieten, welche verwendet werden, um Problem zu identifizieren und kurzfristige Lösungsansätze bieten. Heuristiken werden allerdings in Folgekapiteln noch genauer beschrieben.

Hierbei wäre interessant mit Hilfe von Mitarbeitern innerhalb eines Interviews Probleme durch bestimmte Heuristiken, welche im Grunde relevante Fragen bestimmen, zu identifizieren. Ein solches Vorgehen wird die Verwendung eines durchdachten Fragebogens bedürfen, welcher in Bezug auf die situationsbedingten gewählten Heuristiken unterschiedliche Ausmaße im Umfang annehmen kann.

In Zusammenhang mit dem erwähnten Umfang eines Fragebogens ergibt sich die Frage, ob es dem Interviewer leicht fällt entsprechende Vorgehensprozesse händisch zu organisieren und zu verwalten. Erwähnenswert ist hierbei vor allem die aufgewendete Zeit, welche schließlich Einfluss auf die Effizienz des gesamten Prozesses nimmt. In diesem Sinne möchte ich zum Ziel sowie der Relevanz dieser Arbeit in folgendem Unterkapitel übergehen.

## **Relevanz & Ziel der Arbeit**

Da der zeitliche Aufwand in Bezug auf die Organisierung, Durchführung sowie Verwaltung eines Interviews mittels eines Fragebogens recht umfangreich sein kann, soweit dies größtenteils händisch stattfindet, ist eine digitale Lösung in diesem Zusammenhang wünschenswert.

Durch den Übergang in einen digitalen Prozess kann sich der Interviewer entsprechend Zeit sparen und produktiver arbeiten, da unteranderem Aufgaben wie die Organisierung sowie Verwaltung der Bögen gänzlich wegfallen würden. Die Relevanz einer solchen Idee besteht also darin eine effizientere sowie komfortablere Lösung zur Durchführung von Interviews zu bieten.

Darauffolgend lässt sich das Ziel dieser Arbeit definieren, welches darauf abzielt eine digitale Plattform zur Unterstützung von Interviews zu entwickeln. Die Rede ist von einer vollfunktionsfähigen Software, welche einen Datenbank-/, Server-/ und Clientteil beinhaltet. Diese soll in der Lage sein zur Verfügung stehende Daten abzurufen, Daten zu verarbeiten sowie eingehende Daten innerhalb einer Datenbank strukturell zu sichern.

Das System einer solchen Plattform dient dem Zweck Fragebögen, anhand optional ausgewählter Heuristiken, zu generieren und entsprechende Vorgehensprozesse eines Interviews zu begleiten sowie funktional zu unterstützen. Eingehende Daten eines Interviews sollen verarbeitet und ausgewertet werden, um dem Interviewer einen statistischen Überblick der Antworten zu geben. Die Antwortmöglichkeiten sollen sich hierbei auf eine festgelegte Skala beziehen, welche schließlich eine passende Auswertung ermöglicht.

Eine solche Arbeit soll mit aktueller Technik entwickelt werden, wobei die Architektur des Systems vorher festgelegt werden soll, um eine strukturelle sowie geplante Entwicklung zu ermöglichen. Hierbei wird der sogenannte „MEAN-Stack“ verwendet, welcher in Folgekapiteln erläutert wird.

Abschließend soll ein methodisches Vorgehen zur Evaluierung der UI (User Interface) stattfinden. Hierbei werden unteranderem Aspekte der „Mensch-Maschine-Interaktion“ wie der „Usability“ sowie „Utility“ aufgegriffen, welche mit Hilfe ausgewählter Testpersonen bewertet werden. Entsprechende Themenbereiche werde ebenfalls in Unterkapiteln vor der Verwendung ausführlich beschrieben. Ziel der Evaluierung ist eine Optimierung der Benutzerschnittstelle bzw. eine Möglichkeit für zukünftige Arbeiten oder Themenbereiche zu schaffen, welche im Ausblick dieser Arbeit als solche dargestellt werden. Die entstehenden Resultate der angewendeten Methoden werden unteranderem zusätzlich verglichen, um Vor- sowie Nachteile zu identifizieren.

# **2.Literatur**

## **2.1 Heuristiken**

Heuristiken beschreiben Methoden bzw. Techniken, welche dazu verhelfen innerhalb eines begrenzten Zeitraumes mindestens befriedigende Resultate in Bezug auf Lösungen von bestimmten Problemstellungen zu erzielen (vgl. James Chen, 2020). In diesem Sinne stellen Heuristiken also eine Möglichkeit da, um schnell Entscheidungen treffen zu können. Wichtig ist hierbei, dass die hervorgehenden Ergebnisse nicht dem Optimum entsprechen müssen, bzw. weit von diesem entfernt sein können und flexibel anwendbar sind. Daraus folgt, dass Heuristiken in erster Linie dafür interessant sind bestimmte Probleme innerhalb eines Prozesses oder einem entsprechendem System zu identifizieren sowie vorzubeugen oder um schnell angemessene Gegenmaßnahmen zu treffen. Darüber hinaus lassen sich Heuristiken auch als Beurteilungskriterien beschreiben, welche auf Grundlage von subjektiven Erfahrungen und überlieferten Verhaltensweisen erdacht werden, wobei dies häufig bei schlecht strukturierten oder schwer überschaubaren Problembereichen der Fall ist (vgl. Jean-Paul & Siepermann).

In Bezug auf die vorhergehende Definition von Heuristiken lassen sich Vorteile darin bestimmen, dass Entscheidungsprozesse erleichtert werden. Als Beispiel lässt sich der bereits in der Einleitung erwähnte Wandel im Industriebereich bezüglich der Digitalisierung heranziehen. Hierbei ist der Einsatz von Heuristiken besonders unter Drucksituationen vorteilhaft, da schnell auf vorhandene Probleme reagiert werden kann, um hinnehmbare Lösungen zu erarbeiten. Nachteile einer solchen Herangehensweise liegen allerdings darin, dass durchaus effizientere Lösungen vorhanden sein können, diese aber aufgrund der zeitlich beschränkten Ressourcen nicht realisierbar sind. Zusätzlich lässt sich aus der Definition ableiten, dass Problembereiche ohne bekannte Vorgehensweisen zur Erarbeitung von Lösungen ebenfalls mit Hilfe von Heuristiken eliminiert werden können.

Ein bekanntes Beispiel für Heuristiken stellt Jakob Nielsens Modell für Heuristiken im Bereich der Usability dar. Dieses definiert zehn Heuristiken, welche unter anderem den Status eines Systems oder die Freiheiten bzw. die Kontroller eines Users in Bezug auf die Anwendung als notwendiges Kriterium für optimale Usability darstellen (Nielsen, 1994). Es geht also darum die Usability des eigenen Produkts frühzeitig oder evtl. während der Entwicklungsphase mit Hilfe solcher Heuristiken zu verbessern, bzw. Probleme schnell erkennen zu können. Der Begriff der Usability wird in Folgekapiteln noch einmal genau definiert und soll in diesem Kontext zunächst die Effizienz sowie den optimalen Nutzen einer Benutzerschnittstelle beschreiben.

Wichtig für den Nutzen von Heuristiken sind Fachkräfte, welche sich gut genug mit dem entsprechenden Fachbereich auskennen, um somit die gegebenen Umstände effizient analysieren zu können und die Anwendung bestimmter Heuristiken in den notwendigen Situationen ermöglichen (Hermann & Nierhoff). Sinnvoll ist es hierbei unter anderem unterschiedliche Arten von Fachkräften betrachten und einsetzen zu können, da sich die Meinung in bestimmten Situationen stark unterscheiden kann und eventuell falsche/unnötige Schlüsse gezogen werden, bzw. die Umsetzung der Heuristiken nicht optimal stattfindet (Hermann & Nierhoff).

Da der Einsatz für Heuristiken sehr breit gefächert ist und diese sehr vielfältig sind, werden im Rahmen dieser Arbeit Heuristiken im Bereich der Evaluierung von Benutzerschnittstellen relevant sein und entsprechende Erwähnung innerhalb der angewendeten Methodik finden.

Da sich diese Arbeit zum Ziel genommen hat eine funktionsfähige Anwendung für die Durchführung von Interviews mit Hilfe von Fragebögen zu schaffen, werden im Folgekapitel die Rahmenbedingungen für Fragebögen erläutert.

## **2.2 Usability & User Experience**

### **2.2.1 Usability**

Usability lässt sich als ein Qualitätsmerkmal beschreiben, welches sich auf die Benutzerfreundlichkeit bzw. die Nutzbarkeit von Benutzerschnittstellen einer Software bezieht. In diesem Sinne heißt eine gute Usability, dass eine Software für einen Benutzer handlich und verständlich gestaltet worden ist, sodass dieser keine größeren Schwierigkeiten hat diese zu nutzen (Nielsen, 2012).

Bei Usability werden fünf Aspekte betrachtet, welche maßgeblichen Einfluss auf die Qualität dieser haben und entsprechend beachtet werden müssen. Hierbei werden die Lernbarkeit, Effizienz, Einprägsamkeit, Fehlerrate sowie die Zufriedenheit in Bezug auf die Anwendung mit dem Nutzer betrachtet. Neben diesen Hauptaspekten gibt es noch weitere wichtige Merkmal wie der Utility, welche sich auf die Funktionalität der Anwendung bezieht (Nielsen, 2012). Mit der Utility ist hierbei im groben gemeint, ob die Software die nötige Funktionalität zum Erreichen des Ziels ohne Umwege zur Verfügung stellt.

Grundsätzlich lässt sich sagen, dass eine gute Usability einer Software eine enorme Bedeutung innehält, da diese entscheidend dafür ist, ob ein Benutzer die Anwendung nutzten will oder nicht. Hierbei kann dies die Benutzerschnittstelle einer Webpage sein, wobei gerade hier eine übersichtliche, angenehme sowie funktional effiziente Darstellung vom Nutzer positiv aufgenommen wird. Es lässt sich entsprechend schlussfolgern, dass eine unorganisierte sowie nicht übersichtlich gestaltete Anwendung einen unattraktiven Eindruck auf den Anwender macht und die Chance erhöht für Vorgehensprozesse mehr Zeit zu benötigen als eingeplant war.

Die Analyse sowie Optimierung der Usability, der im Rahmen dieser Arbeit, werden in nachfolgenden Kapiteln vor allem bei der Evaluierung schwerpunktmäßig betrachtet.

### **2.2.2 User Experience**

Die User-Experience wird als „Die Wahrnehmungen und Reaktionen einer Person, die sich aus der Verwendung oder dem voraussichtlichen Nutzen eines Produkts, Systems oder einer Dienstleistung ergeben“ definiert (ISO O 9241-210, 2010). Aus der Definition lässt sich schlussfolgern, dass somit das gesamte Nutzungserlebnis eines Benutzers gemeint ist und entsprechend das System sowie alle weiteren Aspekte neben der konkreten Anwendung Einfluss auf das Erlebnis des Benutzers haben.

Ein interessantes Beispiel hierzu wäre der Erwerb eines neuen Apple-Smartphones. Die User-Experience würde hierbei Werbung, die Beratung, den Erwerb, den Transport sowie schließlich den Nutzen umfassen. Die gesamte Erfahrung, welche mit dem neuen Smartphone assoziiert wird, beeinflusst das Erlebnis des Benutzers und kann entsprechend positiv oder negativ ausfallen. Daher werden die Gefühle des Benutzers in Bezug auf das Gerät und die dazugehörigen Leistungen als Teil des Erlebnisses betrachtet, wobei selbst die Marke eines Produkts eine Wirkung haben kann, da ein Zusammengehörigkeitsgefühl ebenfalls Teil der Erfahrung ist. Die psychologischen Aspekte seien hierbei zunächst dahingestellt, wobei als logische Konsequenz jedoch behauptet werden kann, dass ein Benutzer ein positiveres Erlebnis beim Erwerb eines neuen Gerätes empfindet, wenn alle mit diesem Gerät assoziierten Leistungen ebenfalls positiv ausfallen.

Das Ziel wäre hierbei also eine möglichst gute User-Experience zu schaffen, da der Benutzer nicht nur zufrieden mit der Leistung sein soll, sondern durch positive Impulse der Erfahrung auch zukünftig motiviert wird entsprechende Leistungen in Anspruch zu nehmen.

## **2.3 Fragebögen**

Fragebögen stellen ein Mittel zur Datenerhebung dar und können je nach Ziel unterschiedlich dargestellt und verwendet werden. Hierbei müssen mehrere Aspekte beachtet werden, um eine effektive sowie nützliche Datenerhebung zu realisieren. Da sich diese Arbeit mit der Entwicklung einer App sowie dem Layout der Benutzerschnittstelle beschäftigt, wird die Wahl eines Fragebogens sowie die Konstruktion passender Fragen nicht beachtet, weil diese zu Beginn bereits gegeben waren. In diesem Sinne ist das Design der Fragebögen innerhalb der App relevant, da diese schließlich Einfluss auf die Usability und UX hat.

Das Layout eines Fragebogens sollte so gestaltet werde, dass alle Fragen von der Verantwortlichen Person beantwortet werden. Hierbei ist es wichtig zu beachten, dass das Design den Nutzer ansprechen soll sowie übersichtlich gestaltet sein muss. Das Interesse des Nutzers soll sich bei Ausführung auf den Fragebogen beschränken, weshalb ein beispielsweise überfüllter Fragebogen, aufgrund einer zu kleinen Schriftgröße der Fragen, unattraktiv wirken kann (Williams, 2003). Es ist daher auch empfehlenswert die Fragen in passende Kategorien zu ordnen sowie die Komplexität der Fragen bzw. der Antworten gering zu halten.

In Bezug auf die Antwortmöglichkeiten des Fragebogens, sollen die nach dem US-amerikanischen Sozialforscher Rensis Likert benannten Likert-Skalen verwendet werden. Hierbei wird in Bezug auf eine Frage eine Antwort innerhalb der Skala gegeben, welche symbolisiert inwieweit der Nutzer etwas einschätzt oder einer Aussage zustimmt (Williams, 2003).

## **2.4 Evaluierung**

Als nächstes soll die verwendete Methodik Erwähnung finden, wobei in diesem Fall drei unterschiedliche Verfahren beschrieben werden. Die Evaluierung soll hierbei dem Zweck dienen die Benutzerschnittstelle der Software zu optimieren oder auf zukünftige Arbeiten hinzuweisen, wobei die Resultate in späteren Kapiteln nochmal präsentiert und verglichen werden.

### **2.4.1 Heuristic Evaluation**

Die heuristische Evaluierung bezieht sich auf die zuvor kurz angeschnittenen Heuristiken von Nielsen. Diese wurde im Jahr 1990 eingeführt und wird generell verwendet, um Benutzerschnittstellen zu evaluieren. Hierbei wird zunächst eine kleine Anzahl von Usability-Experten herangezogen, welche die Benutzeroberfläche einer Anwendung anhand der von Nielsen beschriebenen Heuristiken evaluieren sollen. Hierbei wird jedes Problem, das den Experten auffällt, notiert sowie die entsprechende Stelle innerhalb der Anwendung vermerkt (Nielsen & Molich, 1990).

Wichtig bei dieser Methode ist die Erfahrung der Experten auf dem entsprechenden Bezugsbereich. Da es sich hierbei um die Evaluierung der Usability einer Anwendung Handeln soll, sollten die Experten genaues Verständnis besitzen, um möglichst viele potenzielle Probleme ausfindig zu machen. Um die Effizienz der Methode zu steigern, werden laut Nielsen drei bis fünf Experten empfohlen, da hierbei die größte Anzahl an möglichen Problemen im Schnitt abgedeckt wird.

Interessant ist, dass sich die Methode im Allgemeinen gut bewährt und in vielen Projekten Einsatz fand (Novick & Hollingsed, 2007), wobei dies unter anderem auf die Vorteile des Verfahrens zurückführbar ist. Die Methode ist kostenmäßig günstig und recht intuitiv, wodurch Leute leicht zur Durchführung motiviert werden können und keine besondere Planung von Nöten ist. Zusätzlich kann das Verfahren bereits früh im Entwicklungsprozess angewendet werden, was ebenfalls Kosten für die Entwicklung geringhalten kann (vgl. Nielsen & Molich, 1990).

### **2.4.2 Cognitive Walkthrough**

Bei dem „Cognitive Walkthrough“ (Wharton et. al., 1990/1992) handelt es sich um ein Evaluierungsverfahren, welches mit Hilfe einer kleinen Menge an Experten angewendet wird, um die Benutzeroberfläche einer Anwendung zu inspizieren. Hierbei unterscheidet sich die Methode zur „Heuristic Evaluation“, indem die Experten vorher definierte Aufgabenstellungen frei absolvieren sollen und die Leichtigkeit sowie Komplexität dieser bewerten. Da es viele Nutzer grundsätzlich bevorzugen eigenständig ohne Einschränkungen eine Software zu lernen, besteht die Motivation dieser Methode eben darin die Experten uneingeschränkt Aufgaben allein durchführen zu lassen (Wharton et. al., 1990/1992).

Die Methode selbst wird in zwei Phasen unterteilt, wobei als Erstes die Vorbereitung im Fokus steht und im Anschluss die Durchführung auf dieser aufbaut. Hierbei wird innerhalb der Vorbereitungsphase der Input für die Durchführung der Methode bestimmt, welcher die Nutzer sowie deren Wissen beschreibt, das schließlich den Experten ermöglichen soll, die Benutzerschnittstelle aus unterschiedlichen Sichtweisen zu betrachten und entsprechend zu evaluieren. Darüber hinaus werden die durchzuführenden Aufgaben sowie das Ziel bestimmt und beschrieben. Innerhalb der Durchführung wird die Software auf Basis der Vorbereitung von den Experten analysiert, Aufgaben durchgeführt und schließlich evaluiert (Wharton et. al., 1990/1992). Wichtig bei der Durchführung ist, dass der Nutzer bzw. Experte ein Ziel vor Augen hat und zur Lösung innerhalb des Systems hinarbeitet, die aktuell vorhandenen Aktionen klar und eindeutig erkannt werden, der Nutzer bzw. Experte in der Lage ist eine passende zum Ziel führende Aktion auszuwählen sowie das Feedback des Systems nach einer Aktion erkannt und verständlich aufgenommen wird (Novick & Hollingsed, 2007).

Die Methode ist für diese Arbeit darin vorteilhaft, dass zunächst der Aufwand an Leuten nicht sehr groß ist und Experten intuitiv versuchen aus Sicht unterschiedlicher Nutzerarten zu evaluieren. Gerade dadurch, dass die heuristische Evaluierung eine Reihe an Heuristiken vorgibt, kann es sein, dass der „Cognitive Walkthrough“ Aspekte wie Lernbarkeit oder Komplexität aus der uneingeschränkten Intuitionen der Experten heraus besser analysiert.

### **2.4.3 Coaching Method**

Hierbei handelt es sich um eine sogenannte „Thinking Aloud“ Methode zum Testen der Usability einer Benutzerschnittstelle. Die Idee solcher Methode ist, dass Testpersonen herangezogen werden um beim Testen alles was ihnen in den Sinn kommt, in Bezug auf die Anwendung, verbal wiederzugeben (Nielsen, 1993).

Bei der Coaching Methode werden entsprechend Testsubjekte herangezogen, um die Usability einer Anwendung zu testen, indem kontinuierlich Fragen zu dieser an einen Coach gestellt werden. Der Coach kann hierbei einen Experten oder den Tester selbst darstellen, wobei die Person schlussendlich in der Lage sein muss unvorhersehbare Fragen zur Software zu beantworten (Nielsen, 1993).

Vorteile der Methode liegen darin, dass direkte Benutzer angesprochen werden und somit deren Gedankengänge bzw. Interessen durch die gestellten Fragen geschildert werden können. Hierbei resultiert schließlich die Möglichkeit die Usability mit direktem Einfluss potenzieller Nutzer in Richtung ihrer Interessen zu optimieren, was neben der Methodik mit Expertenbezug eine weitere Ebene an Sichtweisen hinzufügt. Für diese Arbeit ist es daher relevant diese Methoden zu verwenden, um zusätzlichen Benutzerbezug bei der Usability herzustellen.

## **2.5 Personas**

Personas werden als wichtiges Design-Tool verwendet um Software, angelehnt auf die Bedürfnisse potenzieller Benutzer, zu entwickeln. Hierbei werden fiktionale Personen so präzise formuliert, dass diese in den Gedanken der Entwickler/Designer als echte Personen wahrgenommen werden und somit die Erschließung sinnvoller Anforderungen erleichtern.

Die Personas sollen also eine bestimmte Zielgruppe für die Anwendung darstellen, wobei diese durch einen Namen, einem Bild zur Person und einer detailreichen Beschreibungen zur Lebensweise, der Arbeit sowie deren Motivationen präsentiert werden (Cooper, 1999). Laut Cooper ist hierbei wichtig, dass mit Hilfe der Personas die Fähigkeiten des Benutzers, dessen Interessen sowie Ziele erkannt werden.

Vorteilhaft ist der Nutzen von Personas vor allem darin, dass man nicht auf echte Personen angewiesen ist. Indem stereotypische Darstellungen von Menschen sowie optional statistische Daten betrachtet werden, lassen sich realistische Anforderungen für eine Software erstellen.

## **2.6 REST**

REST steht für „**RE**presentational **S**tate **T**ransfer“ und stellt einen Architekturstil bzw. ein Paradigma dar, um beispielsweise Webservices zu errichten. Dieses wurde entworfen, um Schnittstellen im WEB nach bestimmten Grundsätzen zu vereinheitlichen (Rohloff et. al., 2014).

Um in diesem Sinne eine Schnittstelle zu implementieren, müssen bestimmte Prinzipien beachtet werden, wovon ausgewählte im Folgenden kurz erläutert werden.

**Adressierbarkeit:**

Ressourcen müssen eindeutig adressierbar sein, wodurch ein konsistenter und einfacher Zugriff ermöglicht wird. Eine Ressource ist hierbei eine Menge von Repräsentationen, welche mit Objekten aus der objektorientierten Programmierung verglichen werden können (Rohloff et. al., 2014).

**Repräsentation:**

Ein Dienst liefert Ressourcen in bestimmten Darstellungen zurück, welche UTF-8 Zeichensätze oder Strukturen wie JSON-Objekte sein können (Rohloff et. al., 2014).

**Operationen:**

Über die Schnittstelle können gewisse Aktionen aufgerufen werden, welche dem Zweck dienen Daten zu lesen oder zu manipulieren. Diese werden in Anlehnung auf das HTTP-Protokoll (Hypertext Transfer Protocol) realisiert und stellen folgende Methoden dar:

* GET: eine Aktion, um Ressourcen zu lesen bzw. abzufragen. Wichtig ist hierbei, dass dieser Zugriff Daten niemals manipulieren, erstellen oder löschen darf (Rohloff et. al., 2014).
* PUT: eine Aktion, um Ressourcen zu aktualisieren. Hierbei wird die Repräsentation einer Ressource durch eine neue, aus der entsprechende Anfrage, ersetzt (Rohloff et. al., 2014).
* POST: eine Aktion, um an eine Ressource Daten zur Verarbeitung zu übergeben. Diese Aktion wird in den meisten Fällen verwendet, um neue Ressourcen anzulegen (Rohloff et. al., 2014).
* DELETE: eine Aktion, um vorhandene Ressourcen zu löschen (Rohloff et. al., 2014).

Wichtig für die auszuführenden Aktionen ist, das notwendige Informationen aus den Parametern der gelieferten URL bzw. dem Body einer Anfrage entnommen werden, sodass Daten verarbeitet oder zur Verfügung gestellt werden können.

**Zustandslosigkeit:**

Während der Kommunikation zwischen einem Server und Client sind keine Informationen untereinander bekannt. Dies hat zur Folge, dass beide Parteien nicht abhängig voneinander sind und Anpassungen nur auf die geänderte Partei Einfluss nehmen. Alle notwendigen Informationen werden bei einer Anfrage mitgeschickt (Rohloff et. al.).

# **3.Verwendete Technik**

Dieses Kapitel dient dem Zweck die verwendete Technik für die Entwicklung der Anwendung zu präsentieren. Hierbei findet eine kurze Erläuterung zu den relevanten Aspekten statt, welche schließlich bei der Umsetzung genutzt werden.

## **3.1 Sprachen**

### **3.1.1 JavaScript**

JavaScript ist eine Skriptsprache, welche entwickelt worden ist, um dynamisches HTML innerhalb von Webbrowsern ausführbar sowie darstellbar zu machen. Hierbei wurde beispielsweise ermöglicht statisches HTML zu erweitern sowie Benutzeraktionen interpretierbar zu machen (Luber & Augsten, 2017). Erwähnenswert ist zusätzlich, dass JavaScript wenig Gemeinsamkeiten mit der Sprache Java hat und man diese entsprechend nicht gleichsetzen darf. Die Sprache besitzt darüber hinaus objektorientierte Eigenschaften, was sie sehr vielseitig macht da objektorientierte Programmierung möglich ist. Darüber hinaus lässt sich JavaScript im Client sowie im Server nutzen, weshalb die Sprache für bestimmte Modelle der Web-Entwicklung, wovon eines in späteren Kapiteln Erwähnung findet, besonders interessant ist.

### **3.1.2 TypeScript**

Die Entwicklung kann sich mit JavaScript innerhalb von komplexeren Projekten schwierig gestalten, da die Sprache nicht typsicher ist, weshalb Fehlererkennungsprozesse erschwert werden. Um diesem Problem entgegenzukommen, wurde von Microsoft die Programmiersprache TypeScript entwickelt. Wichtig ist hierbei, dass der Compiler von TypeScript den Code zu JavaScript kompiliert, weshalb jeder gültige JavaScript-Code auch in TypeScript gültig ist. Die Sprache selbst ist entsprechend typsicher und erleichtert somit die Entwicklung auch im komplexeren Umfeld (typescriptlang.org).

## **3.2 Angular**

### **3.2.1 Das Framework Angular**

Angular stellt ein Framework dar, welches verwendet wird um sogenannte „single-page applications“ (kurz SPA) zu bauen. SPAs sind Webanwendungen, welche dadurch charakterisiert werden, dass eine einzelne Page ein breites Spektrum an Benutzermöglichkeiten zur Verfügung stellt. Die Idee ist hierbei, dass HTML-Elemente der Page asynchron sowie dynamisch geladen oder verändert werden können, wodurch bei Aktionen des Benutzers die gesamte Anwendung nicht neu geladen werden muss. Hieraus folgt, dass Aspekte der UX sowie die Usability einer Page positiv erweiterbar sind und Kommunikationen mit dem Serverteil für CRUD-Aktionen (**C**reate, **R**ead, **U**pdate, **D**elete) erleichtert werden (Monteiro, 2014).

### **3.2.2 Architektur von Angular**

Die Architektur von Angular basiert auf unterschiedlichen Konzepten, welche zunächst kurz erläutert werden.

**Modules**:

Angular-Module stellen kurz gesagt Container dar, welche unterschiedliche Funktionalitäten in Form von Components, Services sowie weiteren Dateien beinhalten können. Im Durchschnitt werden für unterschiedliche Bereiche einer Anwendung Module angelegt, wobei diese z.B. die Kopfzeile, der Body oder die Fußzeile einer Website sein können. Darüber hinaus besitzt Angular ein eigenes System an Modulen „NgModules“, welche Funktionalitäten importieren sowie exportieren können (<https://angular.io/guide/architecture-modules>).

Da die Anwendung im Rahmen dieser Abschlussarbeit klein gehalten wird, gibt es ein einziges Module für jegliche Components, Services sowie restlichen Dateien.

**Components**:

Angular- Components sind für die Views verantwortlich, welche die visuelle Darstellung einer Anwendung beschreiben. Hierbei werden Views durch Components definiert und kontrolliert. Sie stellen somit das Kernfeature von Angular dar, da diese bestimmen was innerhalb einer Page gerendert wird. Wie bereits erwähnt sind Components innerhalb von Modulen vorhanden und können sich somit auf bestimmte Funktionalitäten und dessen Darstellung beziehen, sodass die Gesamtheit dieser eine Anwendung oder einen Teil dessen bildet. Vorteilhaft ist hierbei, dass unterschiedliche Aspekte einer Anwendung voneinander gekapselt werden und somit leicht erweiterbar sowie wiederverwendbar sind.

Components bestehen standardmäßig aus einem Template für den HTML-Teil, einer TypeScript-Datei für die Funktionalität und einer Design-Datei für die Gestaltung der Darstellung. Zusätzlich können Components ebenso Child-Components beinhalten, was eine Verschachtelung mehrere Components ermöglich. Kommunizieren können diese allerdings außerhalb sowie innerhalb einer Verschachtelung miteinander (<https://angular.io/guide/architecture-components>).

In Bezug auf die zu entwickelnde Anwendung, werden Components grundsätzlich eine ganze Seite darstellen, da die Funktionalität im Rahmen dieser Arbeit nicht zu komplex ist und sich dies entsprechend anbietet.

**Services & Depedency Injection**:

Services stellen standardmäßig Klassen dar, welche zur Erreichung eines bestimmten Ziels entsprechende Funktionalitäten zur Verfügung stellen (https://angular.io/guide/architecture-services). Die Idee ist hierbei die Wiederverwendbarkeit bestimmter Funktionen, wobei diese von unterschiedlichen Components durch „Dependency Injektion“ (kurz DI) verwendet werden können. DI ist hierbei ein Entwurfspattern, welches einer Klasse ermöglicht Abhängigkeiten bzw. Funktionalitäten außerhalb dieser abzurufen, ohne dabei ein Objekt aus entsprechender Quelle instanziieren zu müssen, indem die Abhängigkeit in den Konstruktor der Klasse injiziert wird (https://angular.io/guide/dependency-injection). In Bezug auf Services heißt es, dass diese innerhalb eines Components entsprechend im Konstruktor injiziert werden, ohne ein Objekt davon zu erzeugen. Dadurch wird auf eine Instanz an zentraler Stelle zugegriffen, welche entsprechende Methoden der Serviceklasse zur Verfügung stellt.

Im Rahmen dieser Arbeit wird die Anwendung Services hauptsächlich nutzen, um Anfragen an den Server zu schicken, welche schließlich Daten abrufen oder sichern sollen.

**Directives**:

Directives stellen Instruktionen dar, welche spezifizieren wie Components angezeigt werden sollen oder wann HTML-Logik ausgeführt wird. Man unterscheidet hierbei zwischen den Components selbst, Structural Directives sowie Attribute Directives. Für diese Arbeit sind neben den Components die Structural Directives sehr relevant, da diese verhelfen HTML-Logik zu manipulieren. Dies kann der Einsatz von Schleifen sein, um bestimmte Mengen von DOM-Objekten zu erzeugen oder die Verwendung von Konditionen für die Anzeige bestimmter Elemente (<https://angular.io/guide/built-in-directives>). Entsprechend werden die Directives zur Manipulation von HTML-Logik innerhalb der zuständigen Templates angewendet.

**Metadaten**:

Klassen werden in Angular teilweise mit bestimmten Annotationen (Decorator) versehen. Diese bestimmen was für eine Art von Klasse vorliegt, was beispielsweise ein Component, Service oder Module sein kann, und versorgen diese mit bestimmten Metadaten. Die Daten sind notwendig, damit Angular versteht wie es mit den Klassen umgehen soll bzw. als was diese identifiziert werden (<https://angular.io/guide/architecture>).



Abbildung – Angular allgemeiner Aufbau von Component mit Service  
Quelle: <https://angular.io/generated/images/guide/architecture/overview2.png> (später richtig Quelle angeben)

## **3.3 Node und Express**

## 3.3.1 Node.js/Express.js

Node.js stellt eine Open-Source Laufzeitumgebung für JavaScript Anwendungen dar, welche es ermöglicht JavaScript außerhalb von Webbrowsern zu nutzen. Hierdurch können beispielsweise Webserver in JavaScript entwickelt werden. Da Node.js ursprünglich für Google Chrome entwickelt worden ist, läuft die Laufzeitumgebung über die V8 JavaScript Engine, was eine hohe Performanz ermöglicht (<https://nodejs.dev/learn>).

Bei der Anwendung von Node.js besteht ein Vorteil darin, dass sowohl der Client wie nun auch der Server in der gleichen Sprache geschrieben werden können.

Express.js stellt ein Node.js Framework dar, welches die Möglichkeit erleichtert einen Webserver einzurichten sowie Anfragen an diesen anzunehmen und entsprechende Antworten zurückzuliefern (Holmes, 2016).

## **3.4 MongoDB**

MongoDB stellt eine dokumentbasiertes Datenbankmanagementsystem (NoSQL) dar. Im Gegensatz zu relationalen Datenbanken, welche auf dem Konzept von Tabellen mit entsprechenden Reihen und Spalten basieren, werden in MongoDB Collections erstellt. Ähnlich wie Reihen in relationalen Datenbanken werden nun statt dieser Dokumente in Collections gespeichert. Dokumente unterstehen hierbei einer flexibleren Struktur (Schemafreiheit) als es in SQL-Datenbanken der Fall ist und können somit in ihrem Inhalt stark variieren. Diese werden im BSON-Format gespeichert, welches binäres JSON (JavaScript Serialized Object Notation) darstellt und entsprechend vorteilhaft bei JavaScript-Anwendungen ist. JSON ist hierbei in JavaScript eine Möglichkeit Daten in Objekten zu speichern (Holmes, 2016).

Ein großer Vorteil von MongoDB sind die dynamischen Schemata. Wenn man innerhalb von SQL-Datenbanken neue Daten zu bereits existierenden Tabellen hinzufügen will, müssen häufig viele Stellen im Code angepasst werden. Besonders bei großen Datenbanken stellt dies einen langwierigen Prozess dar, welcher nicht bei laufendem Betrieb erfolgen kann. Bei NoSQL-Datenbanken ist es hingegen möglich Daten ohne vordefiniertes Schema zu ergänzen. Die Entwicklung profitiert hierbei darin, dass dies während des Betriebs realisiert werden kann, der Zeitaufwand des Datenmanagements verringert wird sowie die Entwicklung beschleunigt und flexibler wird (https://www.mongodb.com/de/nosql-explained).

## **3.5 Architektur: MEAN-Stack**

Der MEAN-Stack stellt einen Softwarestack dar, welcher dem Zweck dient komplette Webseiten bzw. Software in JavaScript zu realisieren. MEAN ist hierbei ein Akronym, welches sich auf die bereits vorgestellte Technik bezieht (**M**ongoDB **E**xpress **A**ngular **N**ode) (Holmes, 2016). Ein Stack bezeichnet in diesem Sinne eine Reihe von Softwarekomponenten, welche miteinander angewendet werden, um eine vollfunktionsfähige Anwendung inklusive des Server- sowie Clientteils zu entwickeln.

Der Hauptvorteil bei der Anwendung einer solchen Architektur besteht darin, dass es ermöglicht wird durchgehend in einer Programmiersprache zu entwickeln. Konkret bedeutet dies beim MEAN-Stack, dass der Client- sowie Serverteil in JavaScript realisiert werden kann.

Aus der vorgestellten Technik lässt sich entsprechend herausleiten, dass Node, Express sowie MongoDB für das Backend verwendet werden sowie Angular für das Frontend zuständig sein wird. Das Backend bezeichnet in diesem Sinne jegliche server- sowie datenbankbezogene Logik. Entsprechend steht das Frontend für den Client bzw. die Benutzerschnittstelle der Anwendung, welche schließlich auch bei der Evaluierung eine wichtige Rolle einnimmt.

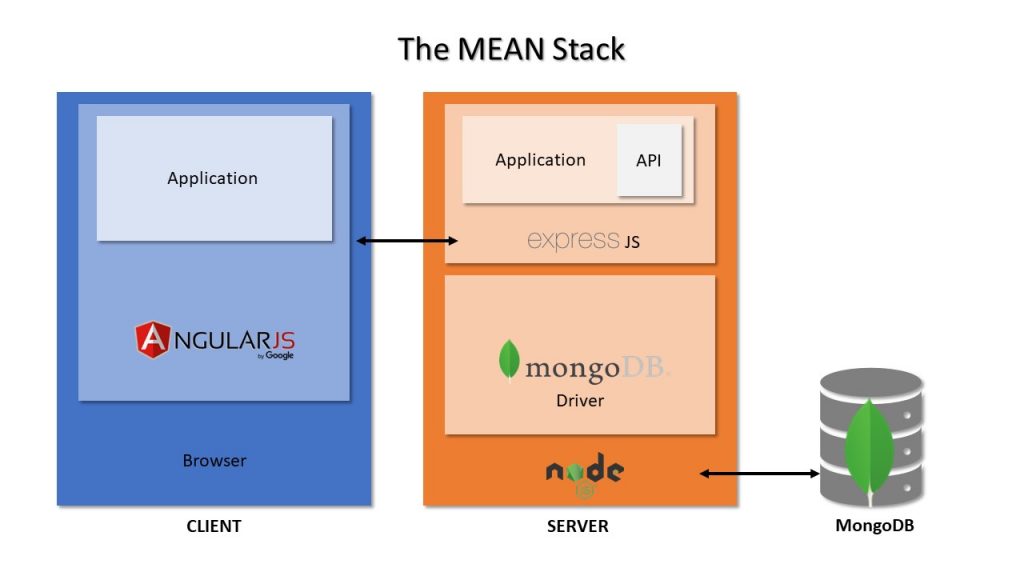


Abbildung - Der MEAN-Stack im Überblick  
Quelle: https://kb.heaptrace.com/mean-stack-requirement

## **3.6 Bootstrap**

Bootstrap stellt ein Open-Source Frontend-Framework dar, welches verwendet wird, um Webseiten zu gestalten. Hierbei wird vor allem die Arbeit innerhalb der Entwicklung erleichtert, indem HTML- und CSS-Vorlagen zur Verfügung gestellt werden sowie Elemente, mit entsprechendem Bezug auf das Framework, bereits responsive dargestellt werden (Shenoy & Sossou, 2014).

Responsive bedeute hierbei, dass sich je nach Hardware die Software auf unterschiedlichen Geräten nach deren Auflösungen richtet und entsprechend dargestellt wird.

Der größte Vorteil bei Bootstrap besteht darin, dass zeitliche Ressourcen bei der Entwicklung effektiver genutzt werden können, da die Gestaltung durch das Framework größtenteils abgedeckt wird.

# **4. Anforderungsanalyse**

## 

Zunächst soll eine grundsätzliche Herausarbeitung notwendiger Anforderungen der Software erschlossen werden, indem die Methodik der Personas verwendet wird, sowie der Ablauf der Anwendung durch ein Flussdiagramm beschrieben werden soll. Die Anforderungen werden schließlich tabellarisch verschriftlicht und in die Entwicklung entsprechend einbezogen.

## **4.1 Personas**

Wie im Literaturkapitel bereits erwähnt sind Personas eine effiziente Möglichkeit zusätzliche Anforderungen für die Entwicklung einer Software zu erstellen. Im Rahmen dieser Arbeit eignet sich der Einsatz auch in Hinblick auf die Evaluierung sehr gut. Hierbei sorgen die unterschiedlichen Persona für verschiedene Perspektiven auf die Anwendung, aus der schließlich evaluiert werden kann. Darüber hinaus hilft diese Methode dabei passende Anforderungen zu erstellen, da man sich als Entwickler mit den möglichen/voraussichtlichen Benutzertypen beschäftigt.

Bei dieser Erstellung der Personas werden jeweils drei verschiedene Typen betrachtet. Zum einen soll die Anwendung für Experten relevant sein und Zusatzfunktionen für die Optimierung von Interviews anbieten. Es soll also eine passende Software für jemanden sein, der diese tagtäglich verwendet. Zum anderen soll die Anwendung aber auch für Menschen relevant sein, die gelegentlich damit arbeiten. Hierbei ist zum Beispiel die Einprägsamkeit der Arbeitsvorgänge besonders wichtig, damit sich die Benutzer auch nach längerer Zeit immer wieder schnell in die Software einarbeiten können. Zuletzt sollen aber auch Benutzer ohne jegliche Vorkenntnisse in der Lage sein ein Interview mittels der Anwendung durchführen zu können. In dieser Hinsicht ist es wichtig, dass Arbeitsprozesse selbsterklärend sind und keine verbale Erläuterung benötigen.

Im Allgemeinen lässt sich hierbei auch sagen, dass durch eine solche Vorgehensweise indirekt Aspekte der Usability sowie User Experience betrachtet und optimiert werden.

**Primäre Persona:**

Hierbei bezieht sich die vorgestellte Persona auf die primäre Zielgruppe, welche gelegentlich bis täglich mit der Anwendung arbeiten wird. Diese werden grundsätzlich Menschen sein, deren Arbeit Erhebung von Daten beinhaltet und diese in Projekte, Methoden, etc. einbezieht.

**A: Willhelm Franck**



**Allgemeine Infos  
Alter: 30**

**Abschluss: Promoviert, Master of Science**

**Job: Wissenschaftlicher Mitarbeiter, Institut für Wirtschaftsinformatik Münster**

**Familienstand: Ledig**

**Charakter: Ambitioniert/Offen/Kritisch**

**Beschreibung:** *Herr Frank ist ländlich aufgewachsen und ist nach der Schulzeit in die Stadt gezogen um Wirtschaftsinformatik zu studieren. Hierbei hat er seinen Master gemacht und neben der Promotion bereits am Institut für Wirtschaftsinformatik gearbeitet.*

**Interessen:** „In meiner Freizeit verfolge ich regelmäßig aktuelle Beiträge in der Informatik sowie Wirtschaft, da mich dieses Wissen für meinen Job auf dem Laufenden hält. Außerdem gehe ich gerne Joggen, lese Bücher und sammele Wertsachen unterschiedlichster Arten.“

**Tätigkeiten im Job:** Konzipierung und Planung von Projektarbeiten sowie Durchführung von Methoden zur Datenerhebung für Forschungszwecke mit Hilfe von gegebenen Heuristiken/Leitfäden.

**Probleme:** „Für meine Arbeit führe ich zur Datenerhebung regelmäßig Interviews mit passenden Bezugspersonen durch. Die Bögen muss ich allerdings selbst drucken, verwalten und analysieren, was eine Menge Zeit in Anspruch nimmt. Darüber kann die Menge an notwendigem Schreibmaterial sehr hoch sein, da unterschiedliche Themenbereiche optional abgefragt werden sollen.“

**Bedürfnisse:** „Ich möchte die Arbeit größtenteils digitalisiert durchführen können, sodass ich mich weniger um Verwaltung und Vorbereitung kümmern muss. Außerdem möchte ich bei der Durchführung von Interviews zusätzliche Funktionen wie dynamisch erzeugter Statistiken oder ergänzender Likert-Skalen haben.“

**Motivation:** „Ich möchte mit meiner Arbeit etwas bewirken können. Hierbei will ich besonders den Bereich der Digitalisierung im wirtschaftlichen Sektor vorantreiben.“



**Allgemeine Infos  
Alter: 23**

**Abschluss: Bachelor of Engineering**

**Job: Student, Werkstudent**

**Familienstand: Ledig**

**Charakter: Freundlich/Ruhig**

**B: Felix Gadow**

**Beschreibung:** *Herr Gadow ist in einer Kleinstadt aufgewachsen und hat nach seinem Abitur ein Bauingenieurs-Studium an der Uni Bochum begonnen. Neben der Uni arbeitet Felix als studentische Hilfskraft.*

**Interessen:** Videospiele, Spieleenwicklung, Sport, Musik

**Tätigkeiten im Job:** Gelegentliche Erhebung von Daten für Forschungszwecke mit Hilfe gegebener Heuristiken/Themenbereiche und vordefinierter Antwortmöglichkeiten (meistens Likert-Skalen), Dokumentation sowie Hilfe bei Projektarbeiten.

**Probleme:** „Die Uni nimmt eine Menge Zeit ein und schränkt mein Privatleben ein oder verursacht zusätzlichen Stress auf der Arbeit.“

**Bedürfnisse:** „Ich möchte mich durch die Arbeit nicht zusätzlich gestresst fühlen und vernünftige Tätigkeiten durchführen können, bzw. die Möglichkeit haben mich in etwas leicht einzuarbeiten. Die Arbeit soll daher nicht zu komplex, übersichtlich und schnell durchführbar sein, da ich nicht auf Vollzeit arbeite und entsprechend wenig Zeit habe, um produktiv zu sein. Wichtig für mich ist, dass ich den Großteil meiner Arbeit digital durchführen kann, da ich so strukturierte arbeite“

**Motivation:** „Neben der Uni möchte ich zusätzliche Arbeitserfahrung sammeln und später bessere Jobmöglichkeiten zu haben“

**Sekundäre Persona:**

Hierbei bezieht sich die vorgestellte Persona auf die sekundäre Zielgruppe, welche eher selten mit der Anwendung arbeiten wird. Dies werden grundsätzlich Menschen sein, die evtl. für eine einmalige Umfrage oder Befragung Daten erfassen müssen, um ihre gewohnten Tätigkeiten durchführen zu können. Hierbei kann es auch beispielsweise der Fall sein, dass diese Persona für andere Menschen einspringen um wichtige Arbeit erledigt zu bekommen.



**Allgemeine Infos  
Alter: 20**

**Abschluss: Abitur**

**Job: Student**

**Familienstand: Ledig**

**Charakter: Offen/Gesprächig/Unsicher**

**C: Jennifer Theilmann**

**Beschreibung:** Frau Theilmann kommt aus der Dortmunder-Innenstadt und hat nach ihrem Abitur ein Studium der Wirtschaftswissenschaften begonnen. Da sie sich noch im Bachelor befindet und keine Berufserfahrung hat, müssen ihr viele Abläufe erklärt werden.

**Interessen:** Musik, Feiern, Shoppen, Reiten

**Tätigkeiten im Studienprojekt:** Datenerhebungen durch Befragungen von verschiedenen Themenbereichen/Heuristiken.

**Probleme:** „Demnächst steht meine erste Projektarbeit im Studium an, für welche ich Interviews mit Testpersonen durchführen muss, da sich der Rest mein Gruppe nicht mit unserem Thema auskennt. Ich weiß allerdings noch nicht wie ich da genau vorgehen soll und ob es evtl. digitale Möglichkeiten für Befragungen gibt.“

**Bedürfnisse:** „Ich möchte möglichst viele Informationen und Erklärungen für meine Aufgaben im Projekt bekommen, da mir eine gute Note wichtig ist. Dabei hoffe ich, dass ich eine sehr selbsterklärende und intuitive Anwendung für die Durchführung meiner Interviews finde. Wichtig ist für mich nur die Durchführung an sich und die Entlastung der Vorbereitung, wobei die Möglichkeit von Notizen bei der Befragung super wären.“

**Motivation:** „Für mich ist es wichtig, dass ich mein Studium gut abschließe, um später einen sicheren Job zu bekommen.“

## **4.2 Anforderungen**

Mit Hilfe der konzipierten Personas sowie dem allgemeinen Ablauf der Software lassen sich nun relevante Anforderungen der App erstellen. Wichtig ist hierbei, dass der Fokus auf den Ablauf der Interviews sowie Funktionen innerhalb der Fragebögen gelegt wird. Zusätzliche Funktionalitäten werden niedriger priorisiert aber dennoch in die Entwicklung einbezogen.

### **4.2.1 Technische Anforderungen**

**Grundanforderungen:**

Zunächst werden die Grundanforderungen für die Software festgelegt, welche entsprechend für jeden Benutzertypen relevant sind. Da mit Hilfe der Anwendung Interviews durchgeführt werden sollen, muss die App in der Lage sein digitale Fragebögen abzubilden. Hierbei ist es notwendig, dass Antworten verarbeitet und in einer Datenbank gesichert werden. Um Fragebögen zu identifizieren, müssen allgemeine Informationen über diese ebenfalls aufgenommen und gesichert werden, sodass später eine Zuordnung verschiedener Informationen stattfinden kann. Des Weiteren wird die Anwendung verwendet, um Interviews zu verschiedenen Heuristiken durchzuführen. Entsprechend wäre es ratsam vor der Durchführung eine Auswahl aus gegebenen Heuristiken zu treffen, sodass keine unnötigen Abschnitte im Fragebogen entstehen. Innerhalb der Bögen sollen einzelne Fragen mit Hilfe einer Skala beantwortet werden können, wobei in diesem Fall Likert-Skalen eingesetzt werden.

Zusammengefasst werden die Grundfunktionalitäten einer Anwendung für digitale Fragebögen festgehalten, wobei hierbei hauptsächlich Datenerfassung, Datensicherung sowie Abbildung der Bögen eine Rolle spielen.

|  |  |  |
| --- | --- | --- |
| **Relevant für Persona:** | **Anforderung** | **Priorität (1=wenig, 3=hoch)** |
| A, B, C | Das System soll in der Lage sein einen digitalen Fragebogen anzuzeigen, wobei Daten abgerufen, gespeichert sowie verarbeitet werden sollen. | 3 |
| A, B, C | Das System soll dem Nutzer die Möglichkeit geben einen Fragebogen aus individuell ausgewählten Heuristiken zu generieren. | 3 |
| A, B, C | Das System soll dem Nutzer ermöglichen Fragen innerhalb einer Skala (1-8, einschließlich Enthaltung) beantworten zu können. (Likert-Skalen) | 3 |
| A, B, C | Das System soll vor der Erzeugung eines Bogens allgemeine Daten abspeichern können, welche generellen Mehrwert beinhalten (Titel, Interviewer, Teilnehmeralter/-geschlecht) und unteranderem zur späteren Identifikation oder Verarbeitung beisteuern sollen. | 3 |

**Primäre Zusatzanforderungen:**

Neben den Grundanforderungen lassen sich zusätzliche Anforderungen erstellen, welche für bestimmte Zielgruppen relevant sind. Persona A, welche als Zielgruppe Experten symbolisiert, erwartet innerhalb der Anwendung Zusatzfunktionen. Diese sollen die alltägliche Arbeit erleichtern und effizienter machen, um somit evtl. für andere Aufgabenbereiche mehr Zeit zu schaffen. Diese Funktionalitäten sollen also optional anwendbar sein und andere Benutzer nicht in ihrer Arbeit einschränken bzw. beeinflussen. Hierbei wird gefordert, dass Fragen neben einer Standardskala zusätzliche Skalen erhalten können. Diese dienen dem Zweck eine Frage aus unterschiedlichen Perspektiven zu betrachten, um somit den Mehrwert der Antworten zu erhöhen. Des Weiteren ist eine visuelle Auswertung der Antworten für Experten relevant, da diese somit in der Lage sind ein Gesamtbild des Interviews zu schaffen und kritische Punkte besser zu erkennen. Entsprechend soll die Anzahl der unterschiedlichen Skalenantworten statistisch dargestellt werden, sodass ein Verteilung der Werte deutlich wird. Da Experten Interviews regelmäßig durchführen, lässt sich eine kritischere sowie tiefere Arbeitsweise folgern. Entsprechend soll die Möglichkeit geboten werden optionale Detaildialoge zu jeder Frage eines Bogens zu öffnen, sodass eine detailliertere Befragung realisiert werden kann. Diese Dialoge sollen daher zusätzliche Fragen zur Vertiefung beinhalten.

Interessant für jeden Benutzertypen ist es Gedankengänge zu verschiedenen Fragen zu verschriftlichen. Daher ergibt sich die Anforderung innerhalb der Fragebögen zu jeder Frage Notizfelder zur Verfügung zu stellen.

|  |  |  |
| --- | --- | --- |
| **Relevant für Persona:** | **Anforderung** | **Priorität (1=wenig, 3=hoch)** |
| A | Das System soll in der Lage sein, zusätzliche Skalen dynamisch zu einer Frage hinzuzufügen, soweit dies benutzerdefiniert der Wunsch ist | 2 |
| A | Das System soll nach der Durchführung eines Interviews neben der Speicherung der Daten diese verarbeiten können und die Ergebnisse innerhalb einer Auswertung grafisch darstellen. | 2 |
| A | Das System soll die Möglichkeit bieten innerhalb der Befragung/Auswertung fragenspezifische Detaildialoge zu öffnen, welche zusätzliche Fragen zur Vertiefung eines bestimmten Gesprächspunktes bieten. | 2 |
| A | Das System soll dem Benutzer innerhalb der Detaildialoge die Möglichkeit geben zusätzliche Notizen machen zu können, wobei diese ebenfalls gesichert werden. | 2 |

**Sekundäre Zusatzanforderungen:**

Neben der Durchführung eines Interviews wäre es interessant, wenn die Anwendung in der Lage wäre alte Fragebogen tabellarisch auswählbar zu machen. Zunächst soll die Auswertung dieser wieder darstellbar gemacht werden, indem der gewünscht Eintrag der Tabelle ausgewählt wird. Des Weiteren soll die Möglichkeit bestehen, einen alten Fragebogen durch die Benutzeroberfläche zu löschen oder zu bearbeiten. Diese Anforderungen sollen dem Zweck dienen dem Benutzer alte Ergebnisse bei Notwendigkeit zu präsentieren oder zu editieren. Da hierbei die Grundfunktion der Anwendung nicht im Mittelpunkt steht, werden diese sekundär betrachtet. Ebenfalls wäre es interessant eine dynamische Verteilung der Antworten innerhalb der Befragung zu visualisieren. Diese soll es dem Benutzer ermöglichen den aktuellen Stand der Dinge zu betrachten.

|  |  |  |
| --- | --- | --- |
| **Relevant für Persona:** | **Anforderung** | **Priorität (1=wenig, 3=hoch)** |
| A, B, C | Das System soll dem Benutzer die Möglichkeit geben durch eine tabellarische Suchmaske auf alte Auswertungen zugreifen zu können, falls die Darstellung noch einmal benötigt wird. | 1 |
| A, B, C | Das System soll dem Benutzer die Möglichkeit geben innerhalb der Suchmaske alte Fragebögen manuell zu löschen. | 1 |
| A, B, C | Das System soll dem Benutzer die Möglichkeit geben alte Fragebögen auf Wunsch zu bearbeiten, wodurch schließlich alte Daten überschrieben werden sollen. | 1 |
| A, B, C | Die aktuelle Statistik der Antworten soll im Fragebogen bereits dynamisch betrachtbar gemacht werden. | 2 |

### **4.2.2 Nichtfunktionale Anforderungen**

Neben den Anforderungen auf technischer Ebene sind einige Zusatzaspekte ebenfalls erwähnenswert. Da im Rahmen dieser Arbeit zukünftige Benutzer der Anwendung grundsätzlich Deutsch sprechen werden, wird die Führung der Software in entsprechender Sprache dargestellt. Darüber hinaus soll die Anwendung mit aktueller Technik entwickelt werden, sodass die Anwendung bei Bedarf angenehm erweiterbar ist. Besonders für Persona B und C ist es wichtig, dass die Anwendung übersichtlich und verständlich gestaltet wird. Dadurch soll die Software zum einen selbsterklärend sowie leicht erlernbar sein und zum anderen einprägsam sein. Hierbei werden entsprechend Aspekte der Usability sowie UX in den Schwerpunkt gesetzt.

|  |  |  |
| --- | --- | --- |
| **Relevant für Persona:** | **Anforderung** | **Priorität (1=wenig, 3=hoch)** |
| A, B, C | Das System soll mit aktueller Technik entwickelt werden. | 3 |
| A, B, C | Die Benutzerführung erfolgt in Deutsch | 3 |
| A, B, C | Die UI soll so gestaltet werden, dass die Usability sowie UX möglichst hoch sind (intuitive Vorgehensprozesse, eindeutig, einprägsam, simpel, übersichtlich) | 3 |
| A, B, C | Die UI soll zusätzlich angenehm gestaltet werden, sodass sich der Benutzer weder überfordert noch uninteressiert fühlt (allgemein „hübsch“, nicht zu komplex). | 3 |

## 

## **4.3 Entwurf der Anwendung**

Mit Hilfe der erarbeiteten Anforderungen lässt sich ein Entwurf der Anwendung erschließen, welcher den grundsätzlichen Ablauf dieser darstellt. Da es ermöglich werden soll entweder alte Ergebnisse zu betrachten oder ein neues Interview zu starten, wird zunächst nach Start der App ein Menü dargestellt. Hierbei soll die Option angeboten werden, eine neue Befragung durchzuführen, alte Ergebnisse zu betrachten oder die App wieder zu schließen.

Entscheidet man sich für ein neues Interview, wird man zur nächsten Maske weitergeleitet, welche allgemeine Informationen zum Interview verlangt. Diese stellen, wie in den Anforderungen festgelegt, Informationen zum Interviewer und dem Teilnehmer dar. Darüber hinaus wird hierbei festgelegt, welche Heuristiken befragt werden, in dem eine Auswahl aller möglichen Kategorien präsentiert wird. Nachdem die notwendigen Informationen angegeben worden sind, wird man nach Bestätigung zur nächsten Maske weitergeleitet. Diese stellt den Fragebogen an sich dar. In Anlehnung zu den ausgewählten Heuristiken sollen hier die entsprechenden Fragen mit Antwortmöglichkeiten dargestellt werden. Der aktuelle Stand sowie weitere Zusatzfunktionen werden während der Durchführung des Interviews zur Verfügung gestellt. Hierbei werden Zusatzfunktionen, welche vor allem für Experten interessant sind, nutzbar gemacht. Nach Abschluss einer Befragung, wird nach Bestätigung eine Gesamtansicht der Ergebnisse präsentiert, wobei ebenfalls die Verteilung der Antworten statistisch abgebildet wird.

Werden alte Ergebnisse benötigt, entscheidet man sich für die entsprechende Option im Menü. Hierbei sollen alle Fragebögen aus der Datenbank tabellarisch dargestellt werden und auswählbar sein. Durch eine Auswahl entsprechender Möglichkeiten soll es ermöglicht werden alte Ergebnisse innerhalb der Auswertung, also dem letzten Schritt der Durchführung, wieder darzustellen oder den Bogen zu löschen. Darüber hinaus soll die Bearbeitung alter Bögen ebenfalls zur Verfügung gestellt werden.

Auf der nächsten Seite befindet sich ein Flussdiagram, welches den Ablauf der Anwendung nochmal in seiner Gesamtheit verdeutlichen soll.

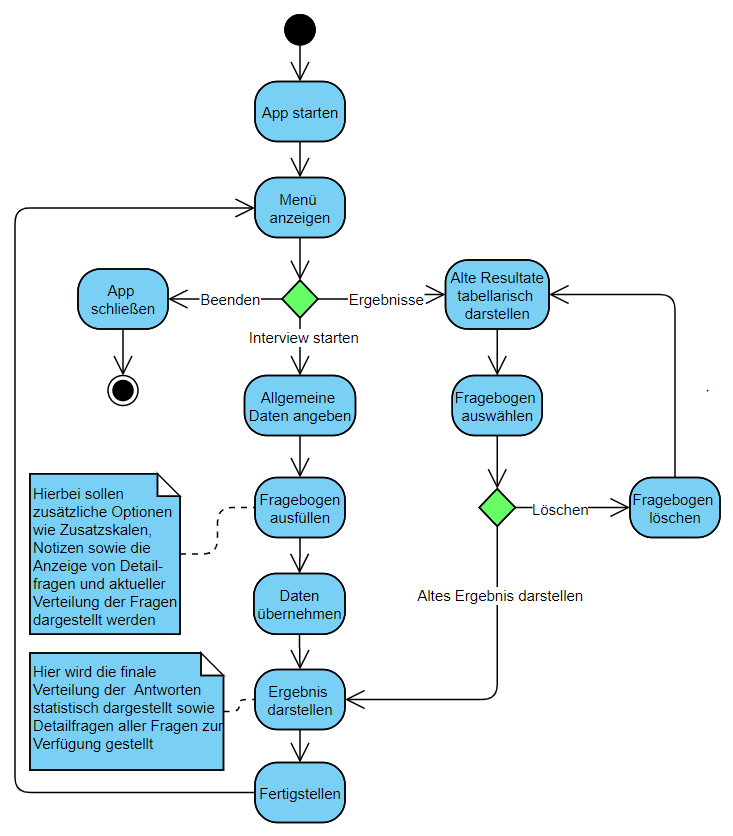


Abbildung - Ablauf der App im Flussdiagramm

# **5. Implementierung (13-17)**

Dieses Kapitel widmet sich der Umsetzung des Entwurfs sowie der herausgearbeiteten Anforderungen für die Software. Hierbei wird die Vorgehensweise in drei Hauptabschnitte eingeteilt, welche die Datenbank, den Server sowie das Frontend bzw. den Client darstellen. Zu Beginn wird der Aufbau der Datenbank besprochen, um ein Verständnis zu schaffen, welche Daten die Anwendung benötigt, bzw. sichern muss. Aus dieser Erkenntnis lässt sich schließlich eine passende Kommunikationsschnittstelle zum Server bis hin zum Client entwerfen. Darüber hinaus ergibt sich mit der Konstruktion einer passenden Datenbank ein Bild vom Frontend. Hierbei ist beispielsweise gemeint, welcher User-Input benötigt wird und wo und wie dieser angegeben wird.

Des Weiteren werden die Anforderungen im Hinblick auf die Prioritäten abgearbeitet. Das Hauptziel ist die Grundfunktionalität der Anwendung zu realisieren, sodass Interviews mit Hilfe der Software ermöglicht werden können. In diesem Sinne werden Zusatzfunktionen nach Abschluss der Grundanforderungen in den Fokus genommen.

Die Projektstruktur wird in das Backend sowie Frontend eingeteilt. Das Backend beinhaltet hierbei den Code zur Datenbank sowie dem Server. Das Frontend enthält entsprechend den Code für den Webclient, wobei ebenfalls das Framework Angular sowie zusätzliche Module enthalten sind.

## **5.1 Datenbank**

### **5.1.1 Allgemeiner Entwurf & Collections:**

Wie bereits im Literaturkapitel erwähnt gibt es bei MongoDB keine Tabellen, wie bei relationalen Datenbanken, sondern Collections. Diese Collections werden mit passenden Dokumenten gefüllt, welche schließlich Daten für unterschiedliche Zwecke beinhalten. Um nun den Bezug zum Projekt herzustellen, müssen passende Collections für die Anwendung entworfen werden. Da die Anwendung allgemeine Informationen zu einem Fragebogen sichern soll sowie Antworten, Notizen, etc. bietet sich die Möglichkeit an zwei unterschiedliche Collections zu erstellen.

Die erste Collection soll allgemeine Daten wie die Fragebogen-ID, den Titel, Informationen zum Interviewer, dem Teilnehmer sowie den ausgewählten Heuristiken beinhalten. Diese Collection kann, bildlich gesprochen, als Titelblatt eines Fragebogens agieren, wobei mit Hilfe dieses Blattes dazugehörige Daten identifizierbar werden. Wichtig ist hierbei die Fragebogen-ID, da mit dieser dazugehörige Heuristiken mit Fragen und Antworten ausfindig gemacht werden sollen.

Die zweite Collection stellt die Heuristiken, bzw. die Kategorie mit Fragen und Antworten, dar. Je nachdem welche Heuristiken ausgewählt worden sind, wird jeweils ein Dokument für eine Heuristik erstellt und dem Fragebogen mit der Fragebogen-ID zugeordnet. Diese Dokumenten sollen schließlich alle relevanten Informationen zu Fragen sowie Antworten der entsprechenden Heuristiken beinhalten.

Da für Fragen optional eine Ansicht von Detailfragen zur Verfügung gestellt werden soll, wird eine dritte Collection benötigt, welche bei Initialisierung mit den vorher festgelegten Detailfragen gefüllt wird und dem Zweck dient über einen Serveraufruf diese zur Verfügung zu stellen. Diese könnte man theoretisch auch im Client hartkodieren, in diesem Fall wird es aber über einen Serveraufruf gemacht, da bereits die allgemeinen Fragen im Client hartkodiert werden.

### **5.1.2 Schemata der Collections**

Um in einer Collection Dokumente zu speichern, muss für diese ein entsprechendes Schema entworfen werden. Mit Hilfe der Schemata lassen sich entsprechende Objekte instanziieren und mit den notwendigen Daten füllen, sodass diese in der dazugehörigen Collection gesichert werden. Um passende Schemata zu erstellen wird Mongoose verwendet, was eine JavaScript-Bibliothek darstellt, die als ODM (Object Document Mapper) agiert. Mit Hilfe von Mongoose können Schemas mit stark typisierten Daten definiert werden (https://www.accentsconagua.com/articles/code/an-introduction-to-mongoose-for-mongodb-and-node-js.html).

Für die erste Collection soll das Schema einen Titel, den Namen des Interviewers, die ausgewählten Heuristiken sowie Informationen zum Teilnehmer beinhalten. Alle Attribute, abgesehen vom Teilnehmer, sind vom Typ String, wobei die Heuristiken, bzw. deren IDs, in einem Array gespeichert werden. Da der Teilnehmer eigene Attribute wie dem Alter und einem Geschlecht beinhaltet, wird für diesen ein eigenes Schema erstellt, welches als Objekt im Fragebogen-Schema eingebettet wird. Das Alter ist entsprechend vom Typ Number sowie das Geschlecht vom Typ String. Darüber hinaus gibt es eine Fragebogen-ID, welche automatisch erzeugt wird und zur Identifizierung verwendet wird.

Das Schema der zweiten Collection, für die Heuristiken, beinhaltet als Attribute eine Heuristik-ID, die dazugehörige Fragebogen-ID, die Fragen an sich sowie den Titel der Heuristik. Die Heuristik-ID wird als String manuell zugewiesen, der Titel ist ebenfalls vom Typ String und die Fragen stellen ein eingebettetes Array von Objekten dar. Diese Objekte besitzen eine Frage-ID, die konkrete Frage, eine allgemeine Notiz sowie eine Detailnotiz, wobei all diese Attribute des Typen String sind. Die Antworten sind schließlich ebenfalls ein eingebettetes Array von Objekten, welche einen Antwortwert, eine Antwort-ID sowie eine Beschreibung besitzen. Abgesehen vom Antwortwert, welcher den Typ Number hat, sind die restlichen Attribute vom Typ String.

Das Letzte Schema für die Detailfragen beinhaltet jeweils die Details der Frage, welche ein Array von Strings darstellen, die Frage-ID sowie die Heuristik-ID. Alle Attribute sind hierbei vom Typen String.

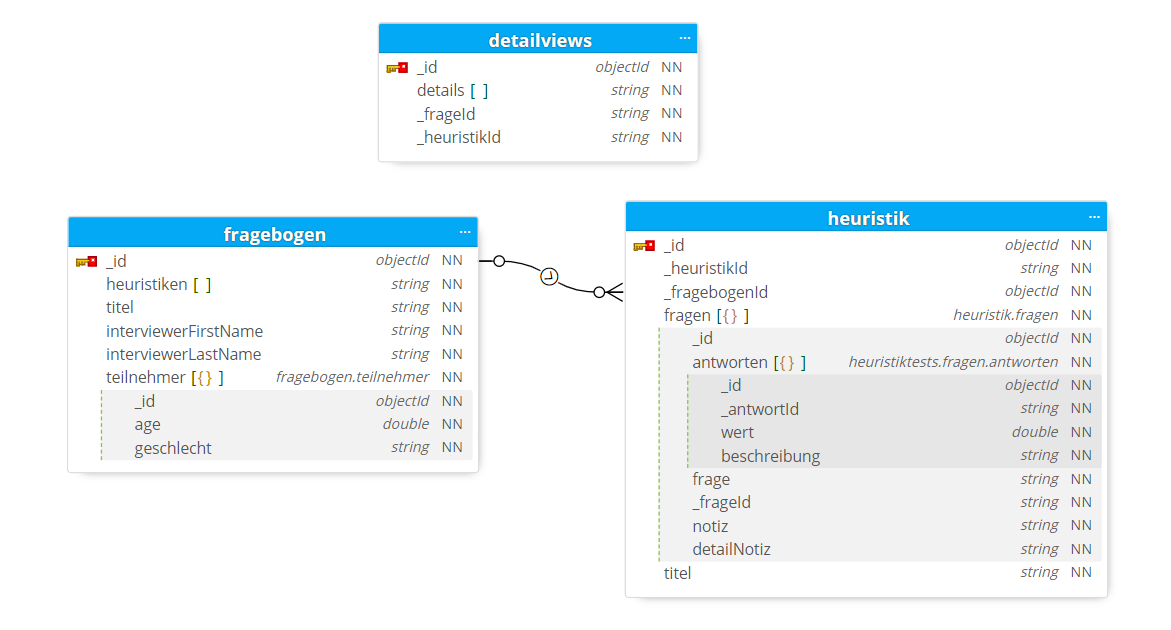


Abbildung - Datenbank: Schemata

## **5.2 Server**

Um über den Client mit der Datenbank zu kommunizieren, muss über einen Server eine Schnittstelle (API bzw. Application Programming Interface) entworfen werden, welche passende Methoden für die jeweils benötigte Aktion zur Verfügung stellt. Diese soll es schließlich ermöglichen, dass Daten von Fragebögen gesichert oder abgerufen werden können, indem sich der Client mit dem Backend über die API durch Anfragen austauscht. Aus dem Literaturkapitel lässt sich zusätzlich entnehmen, dass die Schnittstelle in Anlehnung auf REST realisiert werden soll und somit in diesem Fall mindestens die erwähnten Prinzipien befolgen muss.

Damit das genannte Vorhaben realisierbar wird, muss zunächst ein Server für das Backend eingerichtet werden, welcher schließlich die Schnittstelle zur Kommunikation zwischen der Datenbank und dem Client zur Verfügung stellen wird. Im Literaturkapitel wurde hierbei als Plattform Node.js sowie als Framework Express.js vorgestellt. Im Grunde lässt sich mit Node.js bereits ein Server einrichten, allerdings müssen hierbei grundsätzliche Methoden, welche aus dem REST-Paradigma entnommen werden, selbst implementiert werden. In diesem Fall findet Express.js seinen Einsatz, da das Framework die Implementierung einer API erleichtert, indem die grundsätzlichen REST-Operationen bereits zur Verfügung gestellt werden und lediglich definiert werden muss, wie diese agieren sollen.

//hier passende Abbildung zur Kommunikation einfügen

### **5.2.1 API**

Da nun mit Hilfe von Express ein Server zur Verfügung gestellt wird, muss als nächstes geklärt werden, welche Aktionen die API für die Anwendung bereithalten soll.

**Fragebögen:**

Um einen Fragebogen zu erstellen, müssen zunächst allgemeine Daten angegeben werden, welche schließlich in der Collection für Fragebögen gesichert werden. In Bezug auf die Anforderungen der Anwendung müssen diese Fragebögen erstellt, abgerufen sowie bearbeitet werden können. Daraus folgt, dass grundsätzlich die GET, DELETE sowie POST Operationen benötigt werden. Des Weiteren ist hierbei noch erwähnenswert, dass die Objekte im JSON-Format an die API gesendet werden, bzw. die Antworten der Anfragen ebenfalls im entsprechenden Format zurückgeliefert werden.

Die GET-Methode wird verwendet, um alle Fragebögen innerhalb der tabellarischen Ansicht für alte Ergebnisse anzuzeigen oder um einzelne Fragebögen abzurufen. Da Express die grundsätzlichen Methoden zur Verfügung stellt, wird hierbei für die jeweilige Situation eine GET-Methode aufgerufen. Um alle Fragebögen abzurufen, wird lediglich die passende URL für alle Fragebögen übergeben und mit Hilfe von Mongoose die Datenbank bzw. die dazugehörige Collection über das Schema durchsucht. Um einen bestimmten Fragebogen aufzurufen wird die Fragebogen-ID benötigt, welche als Parameter der URL entnommen wird. Soll ein Fragebogen gelöscht werden, wird hierbei die DELETE-Methode verwendet, welche über die Fragebogen-ID der URL den entsprechenden Bogen aus der Datenbank löschen kann. Wenn ein Interview durchgeführt wird und die allgemeinen Daten erstmals angegeben werden, muss die POST-Operation durchgeführt werden. Hierbei wird bei der Anfrage ein Body mitgeliefert, welcher die neuen Informationen bereitstellt. Mit Hilfe der neuen Daten wird über Mongoose, in Bezug auf das passende Schema, ein neues Objekt erstellt, welches schließlich für die Initialisierung die notwendigen Informationen aus dem Body der Anfrage entnimmt. Zum Schluss wird das neue Objekt in der Datenbank als Dokument in der passenden Collection gesichert.

**Heuristiken:**

Nachdem allgemeine Informationen zu einem Fragebogen angegeben worden sind, wird das eigentliche Interview mit den entsprechenden Fragen der ausgewählten Heuristiken durchgeführt. Entsprechend wird nach Abschluss eine POST-Operation für die Heuristiken benötigt, um diese in der Datenbank zu sichern, bzw. eine neue Ressource anzulegen. Wie bereits erwähnt sollen Fragebögen manuell gelöscht werden können, weshalb auch alle mit einem Fragebogen assoziierten Heuristiken gelöscht werden sollen. Daraus folgt entsprechend, dass die DELETE-Operation ebenfalls von Nöten ist. Um alte Ergebnisse wird darzustellen, müssen zu einem ausgewählten Fragebogen auch die entsprechenden Heuristiken abrufbar sein, weshalb zusätzlich die GET-Methode verwendet wird, um diese entsprechend zu erfragen. Des Weiteren werden Notizen für Detailfragen ebenfalls ermöglicht. Diese werden bei der Durchführung selbst oder der Auswertung optional angegeben. Werden die Notizen bei der Durchführung angegeben, gehören diese entsprechend zur POST-Methode. Ist dies nicht der Fall, wird eine PUT-Operation benötigt, um die Heuristiken in diesem Aspekt zu aktualisieren.

Die GET-Methoden funktionieren wie bei den Fragebögen, wobei für die Identifizierung einer bestimmten Heuristik eines Fragebogens die Fragebogen-ID sowie Heuristik-ID aus den Parametern der Anfrage entnommen werden müssen. Die POST-Operation findet nach der Durchführung eines Interviews statt, indem hierbei die Informationen aus dem Body, wie vorher auch, entnommen werden und ein passendes Objekt mit Mongoose erstellt wird, welches schließlich in der Datenbank als Dokument gesichert wird. Wird ein Fragebogen gelöscht, werden automatisch mit Hilfe der Fragebogen-ID alle dazugehörigen Heuristiken aus der Datenbank entfernt. Soll die Notiz einer Heuristik aktualisiert werden, findet dies mit Hilfe der Fragebogen-ID, der Heuristik-ID sowie der passenden Frage-ID statt.

**Detailfragen:**

Da die Fragen der Detail-Ansicht in der Datenbank verweilen, ist hierbei eine GET-Operation notwendig, um die passenden Fragen vom Client aus zu erfragen. Hierbei wird die Heuristik-ID und Frage-ID benötigt, um schließlich die richtigen Fragen aus der Datenbank bereitzustellen.

## **5.3 Client (8-10)**

Dieses Kapitel beschreibt den Aufbau des Clients. Hierbei wird die Funktionsweise jeder Seite erklärt, wobei die zunächst die grundlegende Funktionalität der Anwendung im Fokus steht. Wie bereits im Technikkapitel erwähnt, nehmen Components in Angular ein wichtige Rolle ein. Im Rahmen dieser Arbeit wird jede Seite oder ein Dialog, dem Diagramm aus Kapitel 4 entsprechend, ein Component darstellen und in seiner Notwendigkeit erklärt, um das Verständnis für dessen Nutzen im Hinblick auf die zu entwickelnde Anwendung zu verdeutlichen. Lediglich die Kopf- sowie Fußzeile der Anwendung werden ebenfalls eigene Components darstellen, welche innerhalb der Pages wiederverwendet werden und zukünftige Erweiterungen, wie evtl. einer Navigation oder anderer Funktionen, beinhalten können.

Die Components werden in eigenen Ordnern in der Projekthierarchie verwaltet, wobei in diesen die dazugehörigen Dateien zur Bearbeitung der Components bereitgestellt werden. Darüber hinaus befinden sich alle Components im Standardmodul eines Angular-Projekts, da im Rahmen dieser Arbeit weitere Module schlicht und ergreifend nicht notwendig sind. Neben den Components werden Services für Anfragen an die API der Anwendung erstellt.

Des Weiteren wird ein Modells Ordner erstellt, welcher benutzerdefinierte Klassen beinhaltet, um Objekte in Anlehnung zu den Schemata der Datenbank zu erstellen. Diese Objekte können somit zur Kommunikation mit der API verwendet werden.

Grundsätzlich lässt sich eine allgemeine Datei für die optische Gestaltung der Views verwenden, allerdings beinhaltet jede Component eine eigene Datei, welche hierbei auch verwendet wird. Des Weiteren wird für die Gestaltung das im Technikkapitel erwähnte Framework Bootstrap verwendet, da dieses den Prozess erleichtert und besonders beschleunigt.

Um später passende Anfragen an den Server ausführen zu können, wird mit Angular ein Routing verwendet, welches je nach Page die URL der Seite anpasst. Hierbei wird ermöglicht, dass Informationen wie die Fragebogen-ID aus den Parametern der aktuellen URL entnommen werden können.

### **5.3.1 Services & Modells**

**Services:**

Bevor die einzelnen Views erarbeitet werden, wird die Kommunikation zwischen Client und API erläutert, um somit das Verständnis zum Datenaustausch zu festigen. Zunächst wird eine allgemeine Klasse „WebService“ erstellt. Diese importiert den von Angular zur Verfügung gestellten „HttpClient“-Service, welcher dem Zweck dient HTTP-Anfragen durchzuführen. Durch DI werden die notwendigen Methoden von dem Angular-Service bereitgestellt, in dem dieser in den Konstruktor des benutzerdefinierten Services injiziert wird. Die Methoden können nun verwendet werden, indem die passende URL als Parameter für die Funktionen übergeben wird. Dieser Service wird im Allgemeinen verwendet, um die API anzusprechen und dient somit anderen benutzerdefinierten Services als Anlegestelle, indem dessen Methoden verwendet werden.

Um nun benutzerdefinierte Anfragen zu implementieren, wird ein weiterer Service „FragebogenService“ erstellt, welcher ebenfalls durch DI die „WebService“-Klasse verwendet. Da „WebService“ die notwendigen Methoden zur Verfügung stellt, um mit der API zu kommunizieren, werden als nächstes konkrete Anfragen für die Anwendung implementiert. Hierbei werden Anfragen erstellt, welche die POST-Methode nutzen, um einen Fragebogen oder dazugehörigen Heuristiken zu erstellen. Die Anfragen nehmen als notwendige Parameter die URL, um beispielsweise die richtige Collection anzusprechen, sowie ein Objekt mit den notwendigen Informationen, welches schließlich aus dem Body der Anfrage im Server entnommen werden kann. Des Weiteren müssen Detailfragen von der Datenbank abgefragt werden, weshalb eine entsprechende Methode erstellt werden muss. Diese verwendet die GET-Operation, um die notwendigen Fragen aus der Datenbank zu bekommen, indem aus der URL Parameter wie der Heuristik-ID sowie Frage-ID entnommen werden. Da die Notizen einer Detailansicht aktualisierbar sein sollen, aufgrund der Vorkommen im Interview sowie der Auswertung, wird eine passende Methode benötigt. In diesem Fall werden aus der URL Informationen wie der Fragebogen-ID, Heuristik-ID und Frage-ID entnommen sowie die Daten zur Aktualisierung aus dem Body der Anfrage verwendet.

**Modells:**

Um passende Objekte mit notwendigen Informationen zur Weiterverarbeitung instanziieren zu können, wird in der Projektstruktur ein weiterer Ordner „Modells“ erstellt, welcher benutzerdefinierten Klassen in Anlehnung zu den Collections der Datenbank beinhaltet. Dieses Klassen werden schließlich verwendet, um z.B. passende Objekte zu erstellen, diese mit Daten zu füllen und zur Verarbeitung an den Server zu leiten. In diesem Fall gibt es ein Modell für Fragebögen, Heuristiken und den dazugehörigen eingebetteten Objekte, welche den Schemata aus Kapitel 5.1.2 entsprechen. Um die Weiterverarbeitung von Informationen innerhalb des Clients zu erleichtern, wurde eine zusätzliche benutzerdefinierte Klasse „Detailview“ erstellt. Diese wird hierbei hauptsächlich dafür verwendet notwendige Informationen an Dialoge, bei Aufrufen, weiterzugeben.

### **5.3.2 Menü**

In Anlehnung zum Flussdiagramm aus Kapitel 4 wird ein Menü erstellt, welches dem Benutzer die Möglichkeit gibt, ein Interview zu starten oder alte Ergebnisse aufzurufen. Hierbei werden drei Button zentriert in einer Box gestapelt. Der erste Button beschriftet mit „Starte Interview“ leitet den Benutzer zur Startpage eines Interviews weiter, wo allgemeine Daten zu diesem angegeben werden. Der Nächste Button beschriftet mit „Ergebnisse“ leitete den Benutzer entsprechend zu einer tabellarischen Ansicht aller Fragebögen weiter. Zuletzt gibt es den Button beschriftet mit „Beenden“, welcher dem Zweck dienen soll die Anwendung zu schließen. Darüber hinaus findet sich eine Sidebar auf der rechten Seiten wieder, welche eine kurze jedoch detailreiche Beschreibung zum Nutzen und der Bedienung dieser App wiedergeben soll. Durch Bootstrap werden die einzelnen Bereiche der Seite in einem Gitter (Grid) angeordnet, sodass die Elemente responsive sind und entsprechend ihre Position und Größe, in Bezug auf die Auflösung des Bildschirms, anpassen. Hieraus folgt, dass die Ansicht auf unterschiedlichen Geräten angepasst wird und die App sich passend bedienen lässt.

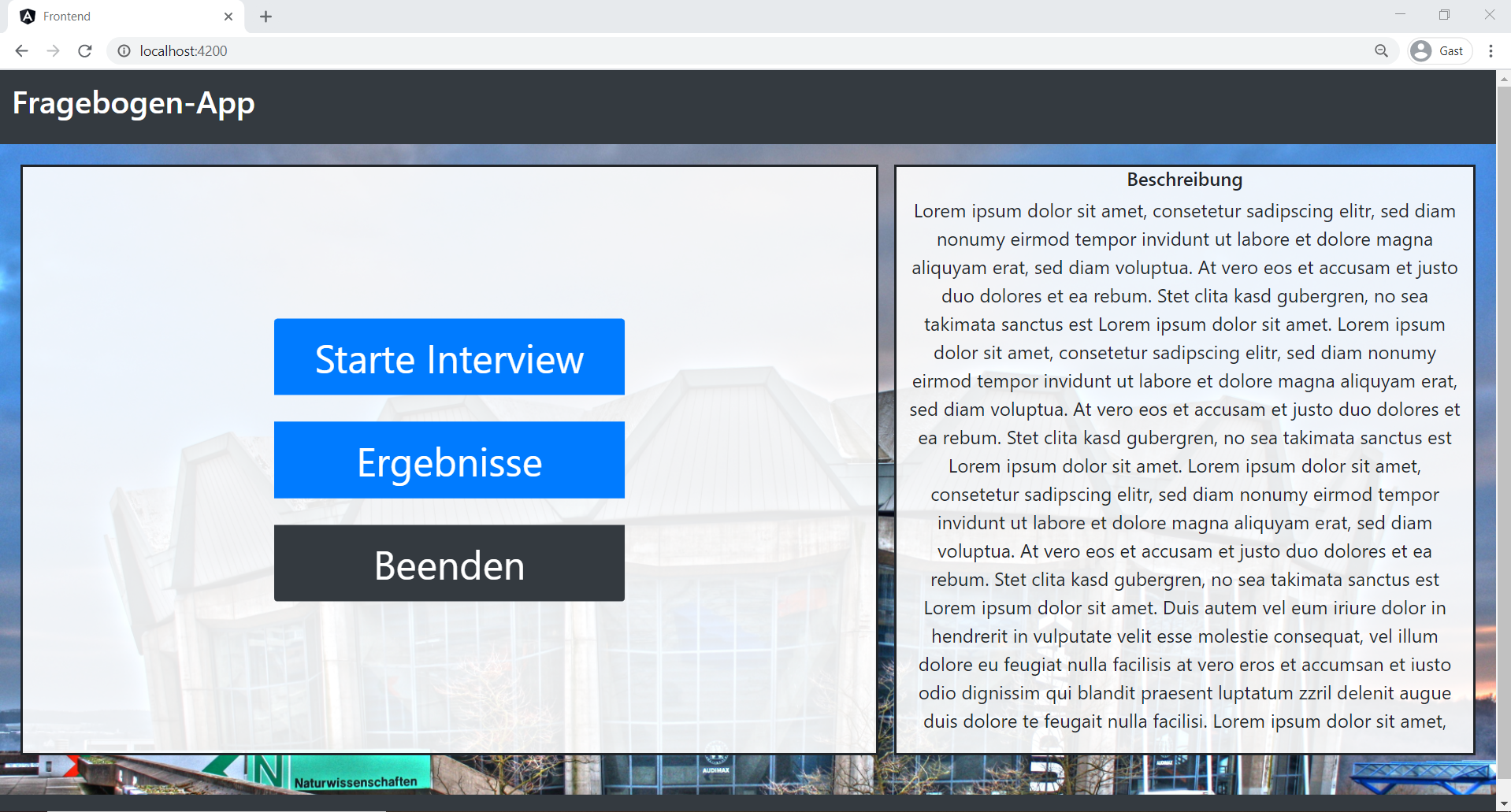


Abbildung – Menü

### **5.3.3 Fragebogen – Allgemeine Informationen**

Wird im Menü der Button beschriftet mit „Starte Interview“ angeklickt, leitet die Anwendung den Benutzer zur Nächsten Seite wobei das Routing ebenfalls angepasst wird. Hierbei werden drei Abschnitte in einem Container präsentiert, welche jeweils Informationen zum Interviewer, dem Teilnehmer sowie zur Auswahl der Heuristiken verlangen.

Für den Interviewer-Abschnitt werden jeweils der Vor- und Nachname des Interviewers sowie ein Titel für den Fragebogen in Textfeldern angegeben. Diese Informationen werden benötigt, um später die verantwortliche Person für das Interview identifizieren zu können und mit einem Titel generellen Mehrwert zur Notwendigkeit oder dem Anlass zu bringen.

Des Weiteren werden für den Teilnehmer das Alter sowie Geschlecht erfragt. Das Alter wird als Zahlenwert angegeben und das Geschlecht lässt sich über drei zur Verfügung gestellte Radiobuttons auswählen.

Zum Schluss müssen die für die Befragung notwendigen Heuristiken ausgewählt werden. Diese lassen sich in dem Fall über Checkboxen auswählen, wobei der ausgeschriebene Titel der Heuristik neben jeder Checkbox vermerkt worden ist.

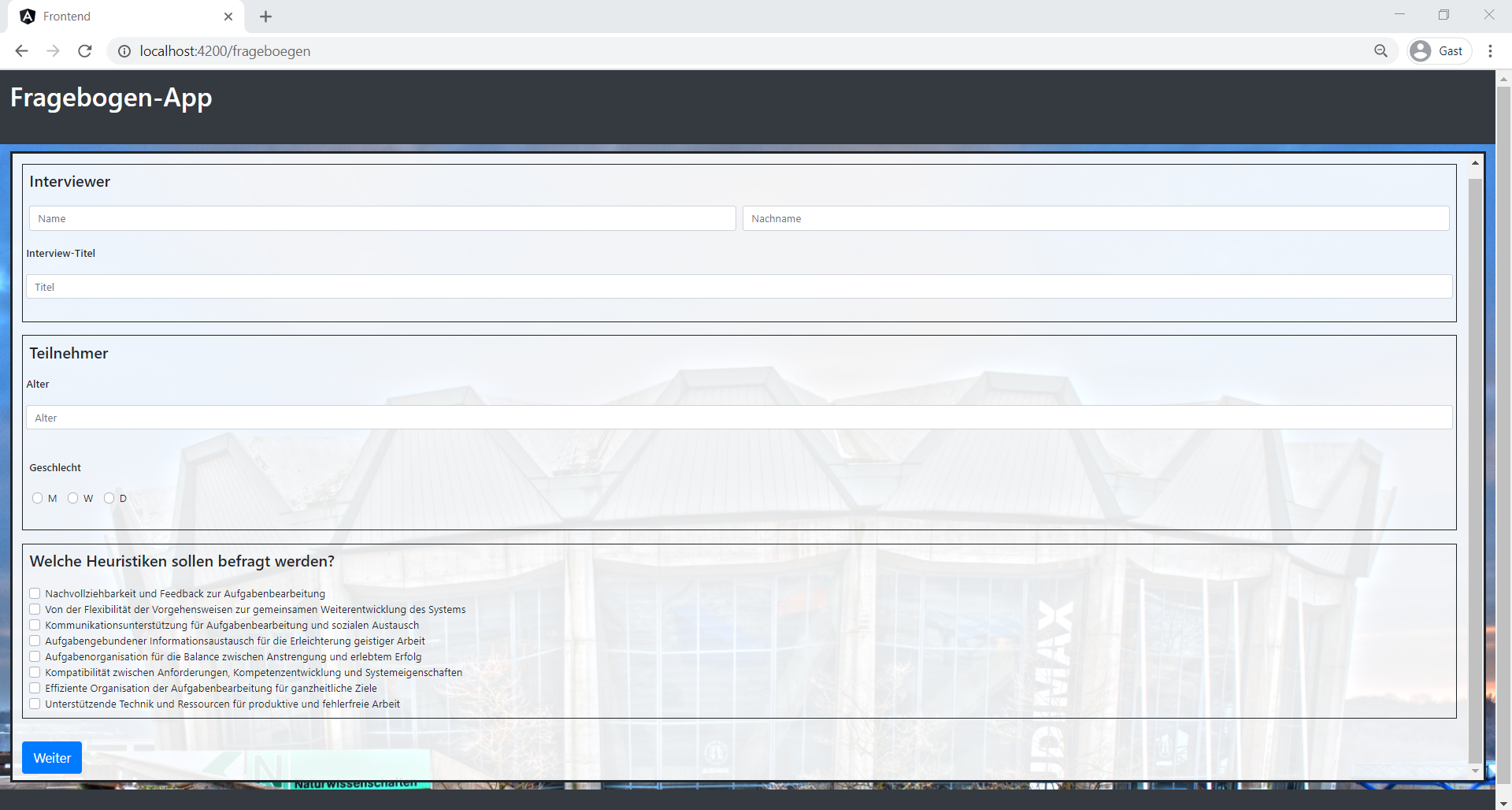


Abbildung - Angabe allgemeiner Informationen

Nachdem alle notwendigen Informationen angegeben worden sind, wird auf Button-Click eine Funktion aufgerufen, welche als Parameter alle angegebenen Informationen aufnimmt. Die passende Klasse für allgemeinen Informationen von Fragebögen wird importiert und ein entsprechendes Objekt instanziiert, sodass die einzelnen Attribute mit den angegebenen Informationen versehen werden können. Bei der Angabe des Geschlechts werden Wahrheitswerte an die Funktion übermittelt, wobei bei der Durchführung mit IF-Anweisungen überprüft wird, welcher Wert wahr ist. Entsprechend wird das zuständige Klassenattribut mit „MÄNNLICH“, „WEIBLICH“ oder „DIVERS“ versehen. Für die Heuristiken wird eine weitere Funktion aufgerufen, welche überprüft, ob die Wahrheitswerte der Checkboxen wahr sind. Ist dies der Fall wird das Array für Heuristiken mit der Heuristik-ID der ausgewählten Heuristik gefüllt, sodass zum Ende hin anhand der IDs erkennbar wird, welche Heuristiken verwendet werden. Zum Schluss wird nach Aufruf der Funktion der Fragebogen-Service aufgerufen. Dieser wurde in der Component-Klasse ebenfalls durch DI in den Konstruktor injiziert, sodass die notwendigen Methoden zur Kommunikation mit der API zur Verfügung stehen. Hierbei wird schließlich die Methode, um einen neuen Fragebogen zu erstellen, verwendet, wobei die notwendigen Informationen aus der URL sowie dem neu instanziierten Objekt entnommen werden.

### **5.3.4 Fragebogen - Durchführung**

# **6.Evaluierung: Anwendung (6)**

## **6.1 Durchführung von Methodik A & B (4)**

* Vorbereitung des Vorgehens
* Vorgehensprozess erläutern aus Sicht unterschiedlicher Personas mit Hilfe der Testpersonen

## **6.2 Durchführung von Methodik C (2)**

* Vorbereitung des Vorgehens
* Vorgehensprozess erläutern aus Sicht unterschiedlicher Personas mit Hilfe der Testpersonen

# **7.Ergebnisse (2-3)**

## **7.1 Vergleich der Resultate (2-3)**

* Kontrast zwischen den Methoden (was hat die eine Methode besser gebracht als die andere?)

# **8.Fazit (2)**

## **8.1 Ausblick (1)**

* Bezug auf Resultate der Evaluierung und grobe Konstruktion neuer zukünftiger sinnvoller Anforderungen

## **8.2 Folgerung (1)**

* Wurde das ursprüngliche Ziel wie geplant erreicht?
* Gab es Dinge die man evtl. hätte anders machen können?

# **9. Literaturverzeichnis (1-2)**

* Chen, J. (2020, 31.07). Heuristics. investopedia. <https://www.investopedia.com/terms/h/heuristics.asp>
* Cooper, A. (2004). *Inmates Are Running Asylum, The: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity*
* Herrmann, T., Nierhoff, J. (2019). *Heuristik 4.0 Heuristiken zur Evaluation digitalisierter Arbeit bei Industrie-4.0 und KI-basierten Systemen aus soziotechnischer Perspektive.* <https://www.ssoar.info/ssoar/handle/document/67682>
* ISO O 9241-210. (2010). //für UX noch richtig aufschreiben
* Nielsen, J. (1994, 24.04/2020, 11.05). 10 Usability Heuristics for User Interface Design. nngroup. <https://www.nngroup.com/articles/ten-usability-heuristics/>
* Nielsen, J. (2012, 03.01). Usability 101: Introduction to Usability*. nngroup.* [*https://www.nngroup.com/articles/usability-101-introduction-to-usability/*](https://www.nngroup.com/articles/usability-101-introduction-to-usability/)
* Nielsen, J., Molich, R. (1990). *Heuristic Evaluation of User Interfaces*
* *Nielsen, J. (1993). Usability Engineering*
* Novick, D., Hollingsed, T. (2007). *Usability inspection methods after 15 years of research and practice*
* Jean-Paul, T., Siepermann, M. (2018, 14.02). *Heuristik*. wirtschaftslexikon.gabler. https://wirtschaftslexikon.gabler.de/definition/heuristik-34474/version-257976
* Saqib Saeed et al. (2017). *Design Solutions for User-Centric Information Systems*
* Williams, A., (2003). *How to…Write and analyse a questionnaire.*
* *Cathleen Wharton et al. (1990/1992). The Cognitive Walkthrough Method: A Practitioner’s Guide*
* *// Qullen für Technikkapitel müssen noch angegeben warden*

# **10.Berufung (1) Seiten: 40- ca. 50+**