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

Wichtig: Literaturverzeichnis noch nicht angepasst, mach ich bei nächster Aktualisierung bzw. nächstem Update => Referenzen stehen aber teilweise im Text

Inhalt

[1.Einleitung 1](#_Toc59030536)

[1.1 Einführung 2](#_Toc59030537)

[1.2 Relevanz & Ziel der Arbeit 2](#_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 11](#_Toc59030554)

[4.4 MongoDB 12](#_Toc59030555)

[4.5 Architektur: MEAN-Stack 12](#_Toc59030556)

[4.6 Bootstrap 12](#_Toc59030557)

[4.Entwurf & Konzeptionierung (5-9) 12](#_Toc59030558)

[5.1 Personas (3) 12](#_Toc59030559)

[5.2 Anforderungen (1-2) 12](#_Toc59030560)

[5.3 Use-Case-Diagramm (1-2) 12](#_Toc59030561)

[5.Implementierung (13-17) 12](#_Toc59030562)

[6.1 Datenbank (2-3) 12](#_Toc59030563)

[6.2 Backend (3-4) 12](#_Toc59030564)

[6.3 Frontend (8-10) 13](#_Toc59030565)

[8.Evaluierung: Anwendung (6-?) 13](#_Toc59030566)

[8.2 Durchführung von Methodik A (3-?) 13](#_Toc59030567)

[8.3 Durchführung von Methodik B (3-?) 13](#_Toc59030568)

[9.Ergebnisse (2-3) 13](#_Toc59030569)

[9.1 Vergleich der Resultate (2-3) 13](#_Toc59030570)

[10.Fazit (2) 13](#_Toc59030571)

[10.1 Ausblick (1) 13](#_Toc59030572)

[10.2 Folgerung (1) 13](#_Toc59030573)

[11. Literaturverzeichnis (1-2) 13](#_Toc59030574)

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

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

Abbildungsverzeichnis

[Abbildung 1 – Angular allgemeiner Aufbau von Component mit Service 11](#_Toc59137458)

# **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.

# **3.Verwendete Technik (4-5)**

## **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 client 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 Benutzer 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. Eine weitere wichtige Funktionalität von Components ist das Databinding, welches mit der TypeScript-Datei und dem HTML-Teil realisiert wird. Klassenattribute Sowie Werte von HTML-Elementen können beidseitig angesprochen werden (<https://angular.io/guide/architecture-components>).

**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 dependecy injection (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.

**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 dieser 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>).

**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 1 – 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

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.

## 3.3.2 Express.js

Express.js stellt ein Node.js-Framework dar, welches

## **3.4 MongoDB**

* Was ist MondoDB?
* Was sind die Vorteile gegenüber SQL-Datenbanken? (kurz)

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

* Was ist der MEAN-Stack?
* Was ist der Vorteil?
* Diagramm von Architektur (+ evtl. zusätzlich projektbezogen)

## **3.6 Bootstrap**

# **4.Entwurf & Konzeptionierung (5-9)**

* Kurz eventuell Einführungstext zur Konzeptionierung etc.

## **5.1 Personas (3)**

* Person, die ein Experte ist (benutzt ständig)
* Person, die keine Ahnung quasi hat (nie benutzt)
* Person, die mit dieser App arbeiten würde (benutzt gelegentlich)

## **5.2 Anforderungen (1-2)**

* Was muss die App auf technischer Ebene können? (Anforderungen darstellen)
* Was muss die App für den Nutzer bringen? (Anforderungen darstellen)
* Bei Anforderungen auf Personas, Prozesse und Literatur eingehen
* Evtl. kurz auf Usibility und Utility eingehen

## **5.3 Use-Case-Diagramm (1-2)**

* Was sind die Use-Cases der App?
* Wie sollen die Abläufe aussehen? (Diagramm)

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

* Allgemein evtl.: Wie wird die Projektstruktur aufgebaut?
* Schrittweise, Abarbeitung nach Zeitplan

## **6.1 Datenbank (2-3)**

* Wie soll die Datenbank aussehen, um das Ziel zu erreichen?
* Welche Collections braucht man?
* Wie sollen die Dokumentschemata aufgebaut sein und warum?
* Evtl. eine Skizze zu Veranschaulichung in Bezug zum Fragebogen

## **6.2 Backend (3-4)**

* Wie wird mit der Datenbank kommuniziert? (Mongoose)
* Wie sieht die API aus? Welche Methoden werden benötigt?
* Wie verhilft hierbei Node und Express zum Ziel? (evtl.)
* Evtl. Skizze zur Kommunikation mit DB oder Client

## **6.3 Frontend (8-10)**

* Projektbezogenen Aufbau erklären (z.B. für jede Page ein Component, Footer und Header ein Component etc., Bootsrap, Modells, Webservices, Routing,etc.)
* Fuktionalität jeder View erläutern und auf Vorgehensweise bzw. Methodik zum Erreichen des jeweiligen Ziels einer View eingehen + Screenshots zur Veranschaulichung neben der Erklärung

# **8.Evaluierung: Anwendung (6-?)**

## **8.2 Durchführung von Methodik A (3-?)**

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

## **8.3 Durchführung von Methodik B (3-?)**

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

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

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

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

# **10.Fazit (2)**

## **10.1 Ausblick (1)**

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

## **10.2 Folgerung (1)**

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

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

* Chen, J. (2020, 31.07). Heuristics. investopedia. <https://www.investopedia.com/terms/h/heuristics.asp>
* 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>
* Nielsen, J. (1994, 24.04). 10 Usability Heuristics for User Interface Design. nngroup. <https://www.nngroup.com/articles/ten-usability-heuristics/>
* Saqib Saeed et al. (2017). *Design Solutions for User-Centric Information Systems*

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