MASTERARBEIT

ENTWICKLUNG EINES ENTSCHEIDUNGSMODELLS FÜR DIE AUSWAHL EINES SOFTWARE FRAMEWORKS

ausgeführt am

Studiengang
Informationstechnologien und Wirtschaftsinformatik

Von: Manuel Reinhart
Personenkennzeichen: 1910320014

Graz, am 09. Dez. 2020 .....

Unterschrift

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....

Unterschrift

DANKSAGUNG

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Masterarbeit beigetragen haben.
Zuerst gebührt mein Dank Herrn Dipl.-Ing. Hans-Peter Grahsl für die äußerst kompetente und hilfreiche Betreuung dieser Masterarbeit.
Des Weiteren gilt mein Dank Herrn DI Robert Goldgruber, sowie der Firma evon GmbH, die mir während meiner Forschung beistanden.
Ein besonderer Dank gilt allen Teilnehmer*innen meiner Umfrage, ohne die diese Arbeit nicht hätte entstehen können.
Abschließend möchte ich mich bei meiner Familie und meiner Freundin bedanken, die in der Zeit der Erstellung dieser Arbeit für mich da waren und mir stets mit Rückhalt und Motivation zur Seite gestanden sind.

KURZFASSUNG

Zu Beginn eines Softwareprojekts stellt sich fast immer die Frage, welche Technologien dafür eingesetzt werden. Die Entscheidung über die Auswahl der Frameworks wird dabei oft nach Präferenz oder vorhandener Erfahrung der Entwickler*innen getroffen. Gerade bei wichtigen oder großen Projekten sollte die Auswahl der Frameworks rational anhand der gegebenen Projektanforderungen getroffen werden.

Ziel dieser Masterarbeit ist es, zu bestimmen, ob die Auswahl der Frameworks durch ein systematisches Entscheidungsmodell unterstützt werden kann. Aus diesem Ziel wurde folgende Forschungsfrage gestellt: *Kann mithilfe eines systematischen Entscheidungsmodells die Auswahl eines geeigneten Single-Page-Application-Frameworks im Bereich Web-Frontendentwicklung getroffen werden?* Die Forschungsfrage wird dabei primär durch Literaturrecherche zur Entscheidungstheorie untersucht. Ferner wird anhand der Recherche ein Entscheidungsmodell entwickelt.

Um die Forschungsfrage zu beantworten, wurden Single-Page-Application- (SPA)Frameworks recherchiert, gefiltert und anhand eines Kriterienkataloges bewertet, um sukzessive ein Entscheidungsmodell zu erarbeiten. Das entstandene Modell bietet die Möglichkeit, SPA-Frameworks anhand ihrer Eignung für individuelle an das Modell übergebene Projektanforderungen zu sortieren und ungeeignete Frameworks auszuscheiden.

Die Bewertung der Frameworks erfolgte dabei mittels einer Bestandsaufnahme messbarer Fakten sowie durch den Vergleich der Frameworks anhand eines erstellten Prototyps bei nicht messbaren Kriterien.

Für die Validierung des erstellten Modells wurde unter Personen aus der Softwareentwicklungsbranche eine Umfrage durchgeführt, bei der die Teilnehmer*innen die Frameworks bewerten konnten. Anschließend wurde mit der Bewertung der Umfrageergebnisse ein neues Modell erstellt und dieses anhand fiktiver Projektanforderungen mit dem ursprünglich erstellten Modell verglichen. Dabei wurde dasselbe Ergebnis erzielt und somit das Modell validiert.

Die vorliegenden Ergebnisse zeigen, dass ein Entscheidungsmodell die Auswahl eines Frameworks ermöglicht oder als Entscheidungsgrundlage dienen kann. Voraussetzung dafür ist die objektive Bewertung der Frameworks sowie der individuellen Projektanforderungen.

ABSTRACT

At the beginning of a software project, there is almost always a need to clarify which technologies are used. The decision about the selected framework is often influenced by the preferences or existing experiences of the developer. Especially for large or important projects, the frameworks should be chosen rationally based on the given project requirements.

The goal of this master's thesis is to determine whether the framework selection process can be supported by a systematic decision-making model. With this goal in mind, the following research question is asked: *Can a systematic decision model be used to select a suitable single-page-application-framework in the area of frontend-web-development?* The research question is primarily investigated through an analysis of literature on decision theory, which is used to develop a decision model.

To answer the research question, single-page-application (SPA) frameworks are researched, filtered and assessed using a catalog of criteria in order to gradually develop a decision model. The resulting decision model offers the possibility of sorting SPA-frameworks based on their suitability for individual project requirements and eliminating unsuitable frameworks.

The frameworks are assessed by evaluating measurable facts and by comparing the frameworks based on a prototype for non-measurable criteria.

To validate the created model, a survey is conducted in which participants from the software development industry evaluate the frameworks. The evaluation in the previously created decision model is then exchanged for that of the survey results to create a new decision model for comparing the survey with the created model using fictitious project requirements. The same result is achieved, so the model is validated.

The results show that a decision model can be used to select a framework or as a basis for decision-making. The prerequisite for this is an objective evaluation of the frameworks and the individual project requirements.

INHALTSVERZEICHNIS

1 EINLEITUNG

In der Softwareentwicklung ist die Auswahl der verwendeten Technologie ein integraler Bestandteil der Entwicklung eines neuen Produkts. Fehlentscheidungen sind zu einem späteren Zeitpunkt oftmals nur schwer und mit viel Aufwand und Kosten reversibel. Zeitgleich steigt bei den Anwender*innen die Erwartung hinsichtlich ständiger Verfügbarkeit, Qualität und hoher Performance. Um diesen Anforderungen nachzukommen, ist der Einsatz von Frameworks, die eine Rahmenstruktur vorgeben, unverzichtbar geworden. Zudem gibt es im Bereich der Softwareentwicklung ein breites Feld an Anwendungsbereichen, die durch die verschiedenen Anforderungen auch die unterschiedlichsten Technologien erfordern. Aufgrund dieser Faktoren stellt die Wahl eines für das Produkt passenden Frameworks eine Herausforderung dar (Suh 2005).

Durch den schnellen Wandel und der Menge und Vielfalt möglicher Software Frameworks ist dem Entwickler*innenteam zumeist nur ein Teil dieser Frameworks bekannt. Der eingegrenzte Blickwinkel sowie die gewohnte Domäne der Entwickler*innen begrenzen möglicherweise auch bessere Alternativen oder führen zur Wahl eines suboptimalen Frameworks.

Um Entwickler*innen im Rahmen des Entscheidungsprozesses zu unterstützen, müsste ein Entscheidungsmodell zur Verfügung gestellt werden, mit dem aus einer Vielfalt an Alternativen die Auswahl möglichst objektiv eingegrenzt werden kann. Gleichzeitig könnten damit Fehlentscheidungen hinsichtlich weniger geeigneter Frameworks vermindert werden.

1.1 Aufgabenstellung

Anhand der zuvor skizzierten Ausgangssituation soll ein Entscheidungsmodell für eine objektivere Softwareframework-Auswahl entwickelt werden. Das Entscheidungsmodell soll Entwickler*innen, durch Berücksichtigung der individuellen Anforderungen des zu entwickelnden Produktes oder Services, das am besten geeignetste Framework als Ergebnis liefern.

1.2 Ziel der Arbeit

Ziel dieser Masterarbeit ist es, Entwickler*innen bei der Auswahl eines geeigneten Software Frameworks durch ein Entscheidungsmodell zu unterstützen und dadurch Zeit und gleichzeitig Kosten zu sparen. Zudem soll das Entscheidungsmodell dafür sorgen, dass die Auswahl von Frameworks, die den Entscheider*innen möglicherweise nicht bekannt oder vertraut sind, in den Entscheidungsprozess mit eingebunden werden kann.

Zusätzlich soll das Modell dazu beitragen, Fehlentscheidungen, die oftmals aufgrund nicht fundierter Meinungen zustande kommen, zu erschweren beziehungsweise zu minimieren.

1.3 Methodik

Als Methode wird für diese Masterarbeit primär auf die Literaturrecherche zurückgegriffen. Neben der Filterung der Frameworks wird zusätzlich ein Kriterienkatalog zur Kategorisierung gebildet. Zudem werden verschiedene wissenschaftliche Bewertungsmethoden, zum Beispiel eine Checkliste oder Ja-Nein-Abfragen angewendet, um möglichst sachliche, objektive und transparente Bewertungen zu gewährleisten. Für die Entwicklung des Entscheidungsmodells selbst werden gängige Methoden der Entscheidungstheorie, beispielsweise die Bildung einer Ergebnismatrix oder von Entscheidungsregeln herangezogen. Für die Validierung des Ergebnisses wird eine Umfrage mit Expert*innen aus der Softwareentwicklungsbranche durchgeführt und die Meinungen der Expert*innen mit dem Ergebnis des Entscheidungsmodells verglichen.

1.4 Ablauf

Zu Beginn soll mittels einer Literaturrecherche ein Überblick über das Gesamtthemenfeld geschaffen und anschließen anhand aktueller Trends der Softwareentwicklung hinsichtlich der Detailbreite und Detailtiefe eingegrenzt werden.

Im nächsten Schritt sollen für das zuvor festgelegte Themenfeld mittels einer Literaturrecherche Frameworks gesucht und eine Auswahl gefiltert werden. Diese Filterung soll anhand objektiver Fakten, beispielsweise anhand der Beliebtheit oder des Verwendungsgrades, durchgeführt werden, um möglichst relevante Frameworks für das Entscheidungsmodell zu identifizieren.

Nach der Sammlung der Frameworks soll ein Kriterienkatalog erstellt werden, mit dem sie nachfolgend kategorisiert werden können und in Folge für die Zusammenstellung des Modells benötigt werden. Die Kriterien sollen mittels einer Literaturrecherche ausgewählt werden.

Mithilfe des Kriterienkataloges können die Frameworks anhand der einzelnen Kriterien bewertet werden. Dabei sollen verschiedene wissenschaftliche Bewertungsmethoden herangezogen werden, um eine möglichst hohe Objektivität zu gewährleisten.

Um nicht messbare Kriterien des Kriterienkataloges zu bewerten, soll mit den zuvor ausgewählten Frameworks jeweils ein Prototyp mit gleichen Anforderungen erstellt werden. Mithilfe der erstellten Prototypen soll anhand eines direkten Vergleiches der Frameworks die Bewertung dieser Kriterien erfolgen.

Anhand der bewerteten Frameworks kann ein geeignetes Entscheidungsmodell zusammengestellt werden, das durch eine Gewichtung der Kriterien hinsichtlich spezifischer Projektanforderungen die dafür geeignetsten Frameworks als Ergebnis liefern soll.

Nach der Erstellung des Entscheidungsmodells wird eine Umfrage durchgeführt. Diese Umfrage richtet sich an Expert*innen aus der Softwareentwicklungsbranche und soll darüber Auskunft geben, welches Framework für welche Kriterien in der Praxis verwendet wurde.

Im letzten Schritt soll das Modell validiert werden. Dabei wird ein zweites Modell mit den Umfrageergebnissen aufgestellt und mit dem zuvor erstellen verglichen werden, um dadurch qualitativ feststellen zu können, wie gut das Modell performt.

Um den Ablauf der Masterarbeit übersichtlich zusammen zu fassen, wird dieser in der Abbildung 1 als Prozess dargestellt. Der Prozess startet mit der Forschungsfrage und endet mit den Ergebnissen der Forschung. Dabei nimmt jeder Prozessschritt die Ergebnisse der vorherigen Prozesse entgegen und lässt durch dessen Verarbeitung ein neues Ergebnis entstehen.

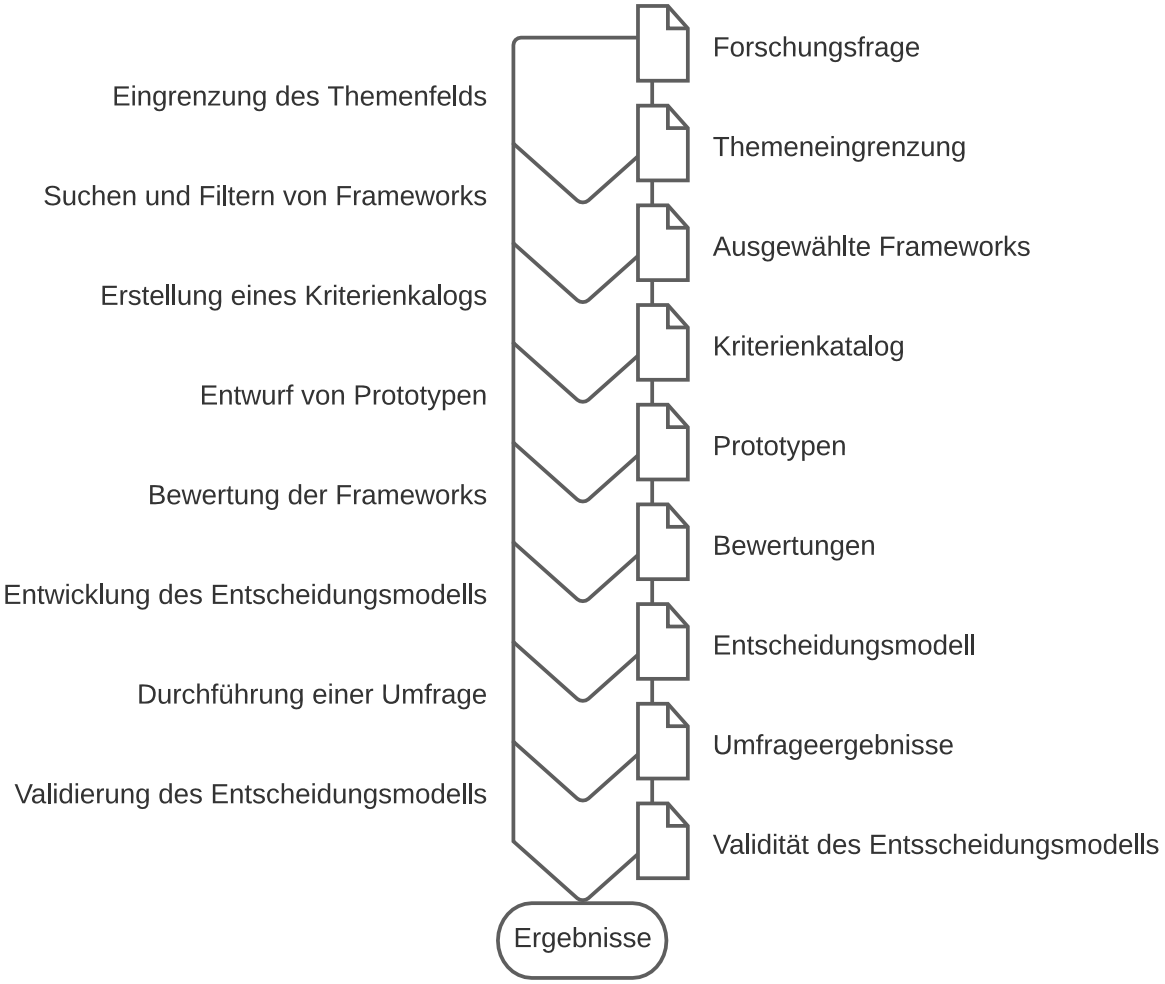


Abbildung 1: Ablauf der Masterarbeit als Prozess (Eigene Darstellung)

1.5 Hypothesen und Forschungsfrage

Die Forschungsfrage, die in dieser Masterarbeit beantwortet werden soll, wird in Kapitel 3 eingegrenzt. Sie lautet wie folgt:

Kann mithilfe eines systematischen Entscheidungsmodells die Auswahl eines geeigneten Single-Page-Application-Frameworks im Bereich Web-Frontendentwicklung getroffen werden?

Die daraus abgeleiteten Hypothesen lauten:

H1: *Mithilfe eines systematischen Entscheidungsmodells kann die Auswahl eines geeigneten Single-Page-Application-Frameworks im Bereich Web-Frontendentwicklung getroffen werden.*

H0: *Mithilfe eines systematischen Entscheidungsmodells kann die Auswahl eines geeigneten Single-Page-Application-Frameworks im Bereich Web-Frontendentwicklung nicht getroffen werden.*

2 GRUNDLAGEN

Nachfolgend werden einige Begriffsbestimmungen, sowie theoretische Grundlagen der Erarbeitung der Forschungsfrage beschrieben. Das Verständnis dieser Begriffe und Grundlagen sind für die weitern Kapitel dieser Masterarbeit essenziell.

2.1 Begriffsbestimmungen

Mit Hilfe der Definitionen der einzelnen Begriffe wird ein einheitliches Verständnis dieser Arbeit geschaffen. Es werden dabei jene Begriffe bestimmt, die eine fundamentale Rolle in dieser Arbeit spielen.

2.1.1 Software

Je nach Autor*in und Kontext variiert die Definition des Begriffes *Software*, jedoch überschneiden sie sich im Wesentlichen. Im allgemeinen Sprachgebrauch bezeichnet der Begriff ein Programm, das bestimmt, was eine darunterliegende Hardware ausführen soll. Zusätzlich schließt der Begriff auch andere Arten von Informationen und Daten wie Quellcode oder Dokumentation ein. In der Literatur wird *Software* vom dazu komplementären Begriff *Hardware* oft dadurch abgegrenzt, dass Software alle immateriellen und Hardware alle materiellen Teile eines computerbasierten Systems umfasst (Kurbel 2008; ISO/IEC 2382:2015-05).

Neben dem allgemeinen Begriff wird in weiterer Folge in Unterbegriffe gegliedert. So wird in der Wirtschaftsinformatik zwischen System- und Anwender*innensoftware unterschieden. Dabei umfasst Erstere das Betriebssystem und systemnahe Software wie Datenbanksysteme oder Firewalls. Letztere arbeitet oberhalb der Systemsoftware und wurde speziell für die Lösung eines Anwender*innenproblems, wie ein Warenwirtschaftssystem, entwickelt (Kurbel 2008).

Eine spezielle Form der Anwender*innensoftware ist die Entwicklungssoftware. Diese wird für die Herstellung anderer Software benötigt. Dabei wird im Bereich der Softwareentwicklung häufig der englischsprachige Begriff Integrated Development Environment (IDE) verwendet, der übersetzt integrierte Softwareentwicklungsumgebung bedeutet (Kurbel 2008).

2.1.2 Framework

Der englischsprachige Begriff *Framework* bedeutet übersetzt Rahmenstruktur. Er wird in der Informationstechnik (IT) dafür verwendet, ein Programmiergerüst zu beschreiben. Frameworks sind per se keine fertigen Programme, sondern stellen die Basis für die Entwicklung einer

individuellen Software dar. Dabei wird durch das Framework in der Regel die Architektur der Software vorgegeben. Ferner beinhaltet ein Framework zumeist allgemeine abstrakte Funktionen und Komponenten, die bei der Entwicklung unterstützen. Bereits im Jahr 1988 beschrieben Johnson und Foote den Begriff folgendermaßen: „A framework is a set of classes that embodies an abstract design for solutions to a family of related problems, and supports reuses at a larger granularity than classes. During the early phases of a system's history, a framework makes heavier use of inheritance and the software engineer must know how a component is implemented in order to reuse it. As a framework becomes more refined, it leads to “black box” components that can be reused without knowing their implementations.” Auf diese Definition verweisen im Jahr 2004 auch die beiden Autoren Massol und Husted in ihrem Buch „JUnit in Action” und definieren selbst den Begriff wie folgt: “A framework is a semi-complete application. A framework provides a reusable, common structure that can be shared between applications. Developers incorporate the framework into their own applications and extend it to meet their specific needs. Frameworks differ from other toolkits by providing a coherent structure, rather than a simple set of utility classes.” (Johnson und Foote 1988; Massol und Husted 2004)

Da Frameworks für verschiedene Bereiche und Anwendungszwecke ausgelegt sind, werden Frameworks in Applikations-, Domänen-, Klassen-, Komponenten-, Koordinierungs-, Web und Testframeworks untergliedert. Häufig decken Frameworks jedoch mehrere Typen ab, wodurch keine strikte Trennung möglich ist (Cwalina und Abrams 2007).

Um den Begriff *Framework* von Bedeutungen außerhalb der Softwaretechnik abzugrenzen wird in dieser Arbeit der Begriff *Softwareframework* verwendet.

2.1.3 Web-Frontendentwicklung

Um die Begrifflichkeit *Web-Frontendentwicklung* bestimmen zu können, ist es erforderlich, die Teilbegriffe *Web*, *Frontend* und *Entwicklung* zu verstehen. Der letztere Begriff entstammt dem Begriff *Softwareentwicklung* und bezieht sich auf die Herstellung einer Software oder eines einzelnen Programms (Lackes 2018).

Als *Frontend*, wörtlich übersetzt *vorderes Ende*, wird im Bereich der Software jener Teil des Programms bezeichnet, der sich näher dem Nutzenden befindet. Das Pendant zum Frontend ist das Backend, das sich näher auf der Systemebene befindet. Eine strikte Trennung ist nicht immer möglich. Der Begriff findet insbesondere bei Server-Client-Architekturen seine Bedeutung (Fischer und Hofer 2011).

2.1.4 Single-Page-Application

Eine Single-Page-Application (SPA) bezeichnet eine Webseite oder -anwendung, die im Gegensatz zu traditionellen Webseiten mit einer einzigen Seite realisiert ist. Die Inhalte der Webanwendungen werden dabei dynamisch nachgeladen. Diese Vorgehensweise erhöht die Interaktivität und vermittelt Anwender*innen das Bediengefühl einer Desktopanwendung.

Gleichzeitig wird die Serverlast durch das verstärkte clientseitige Ausführen von Skripts verringert. Mithilfe von Caching Verfahren ist es möglich, SPA-Webanwendungen offline auszuführen. Ein Nachteil des SPA-Ansatzes gegenüber traditionellen Webseiten ist die schwerer durchführbare Suchmaschinenoptimierung, da die Inhalte der Webseite beim Aufrufen nicht von Beginn an geladen werden (Siepermann 2018).

2.1.5 Entscheidung

Der Begriff *Entscheidung* bedeutet, aus mindestens zwei vorhandenen Alternativen eine Handlung zu wählen, wobei auf die Erfüllung übergeordneter Ziele geachtet wird. Generell werden Entscheidungen von sogenannten Entscheidungsträger*innen getroffen. Dabei handelt es sich im Allgemeinen um natürliche Personen, die für sich selbst oder für Organisationen entscheiden. Entscheidungen, die von computerbasierten Systemen getroffen werden, sind das Ergebnis eines logisch ablaufenden Programmes, das wiederum von einer natürlichen Person erstellt wurde (Wessler 2012).

Der Weg von der Entstehung bis zur Umsetzung der Entscheidung wird durch einen Entscheidungsprozess mit sieben Phasen abgebildet. Diese sind in der nachfolgenden Abbildung grafisch dargestellt (Haun 2016).

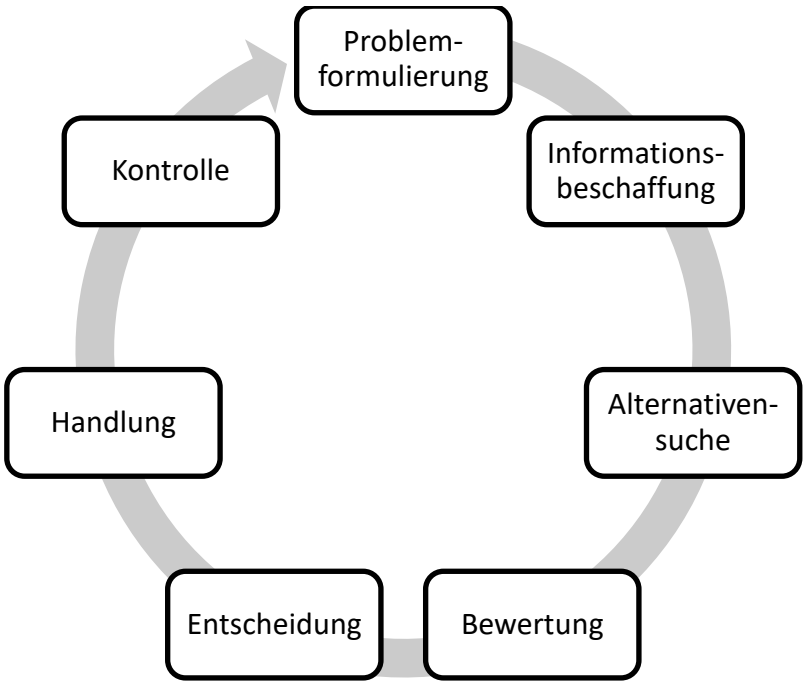


Abbildung 2: Phasen eines Entscheidungsprozesses (Haun 2016)

In der ersten Phase, der Problemformulierung, wird die Notwendigkeit einer Entscheidung erkannt und das Entscheidungsproblem bewusst oder unbewusst formuliert. In der zweiten Phase, der Informationsbeschaffung, werden alle relevanten Informationen, die zu einem besseren Verständnis des Problems und dessen Ursachen beitragen, gesammelt und zu einem Lösungsfeld aufgespannt. Die Informationssammlung hat einen starken Einfluss auf die Entscheidungsqualität. Durch die vorhandenen Informationen können in der dritten Phase, der Alternativensuche, mithilfe von Kreativitätstechniken mögliche Handlungsoptionen aufgestellt

werden. Dabei werden die Handlungsalternativen durch die Einhaltung von Rahmenbedingungen begrenzt. Nach Abschluss dieser Phase können in der nächsten Phase, der Bewertung, die einzelnen Alternativen auf Basis übergeordneter Ziele bewertet werden. Unsicherheiten von Eintreten von Bedingungen erschweren dabei die Bewertung. Mit der fünften Phase, der Entscheidung, wird durch die Entscheidungsträger*innen eine Handlungsalternative gewählt. Dabei wird die Wahl sowohl durch die rationalen Kriterien der Bewertung als auch durch irrationale weiche Faktoren beeinflusst. Nach dem Entschluss wird mit der sechsten Phase, der Handlung, die ausgewählte Alternative umgesetzt. Abschließend wird mit der letzten Phase, der Kontrolle, der Erfolg der Entscheidung hinsichtlich des geplanten Ziels und der sich ändernden Umweltbedingungen überprüft und ist Auslöser für den Beginn eines neuen Regelkreises (Haun 2016).

2.1.6 Entscheidungsmodell

Entscheidungsmodelle zählen zu den wichtigsten Entscheidungshilfen, die in der deduktiven Forschung erarbeitet werden. Bretzke definierte im Jahr 1980 den Begriff in seinem Buch „Der Problembezug von Entscheidungsmodellen“ folgendermaßen: „Als Entscheidungsmodell bezeichnen wir im Folgenden ganz allgemein das Ergebnis eines Versuches, die für wesentlich gehaltenen Elemente und Beziehungen einer als ‚Problem‘ empfundenen Handlungssituation in einer formalisierten Sprache so zu definieren, dass aus dem resultierenden Strukturkomplex die Problemlösung als logische Implikation abgeleitet werden kann“. Es werden also bestimmte Typen von Entscheidungssituationen oder -problemen mithilfe eines Modells allgemein abgebildet, sodass damit reale Entscheidungsprobleme gelöst werden können. Durch seine strukturierte Logik kann das Modell als Computerprogramm umgesetzt werden und ermöglicht dadurch eine rechnergestützte Problemlösung (Amann 2019).

Weiters können allgemeine in konkrete Entscheidungsmodelle überführt werden. Ein konkretes Modell bezieht sich auf ein spezifisches Entscheidungsproblem, bei dem die jeweiligen Modellparameter durch die konkrete Entscheidungssituation festgelegt sind (Amann 2019).

2.2 Web-Grundlagen

Mithilfe der in diesem Kapitel angeführten Grundlagen im Bereich der Web-Frontendentwicklung soll ein grundlegendes Verständnis der Funktionsweise von Webseiten vermittelt werden.

2.2.1 Aufbau einer Webseite

Die Basis jeder Webseite im World Wide Web sind die Inhalte von Hypermedia-Dokumenten, die der Browser von einem Webserver abruft. Zur Beschreibung der Hypermedia-Dokumente wird die Beschreibungssprache Hypertext-Markup-Language (HTML) verwendet, mit der die inhaltliche Struktur des Dokumenteninhaltes der Webseite ohne Formatierung beschrieben wird. Inhalte dieser Dokumente können Text, Grafiken, Bilder, Hyperlinks zu anderen

Dokumenten, Ton, Video oder andere mediale Inhalte sein. Zusätzlich enthalten HTML-Dateien Dokumentenheader und Metatags, die dem Browser zusätzliche Information wie den Dokumententitel oder die Angabe zusätzlich benötigter Dateien bereitstellen. Wird die HTML-Datei vom Browser geladen, erstellt dieser ein sogenanntes Document-Object-Model (DOM), was übersetzt *Dokumenten-Objektemodell* bedeutet. Dieses bildet eine Baumstruktur mit einzelnen Objekten (Meinel und Sack 2004; Berners-Lee und Fischetti 1999).

In der nachfolgenden Abbildung wird die Struktur einer beispielhaften Webseite anhand eines DOM dargestellt. Die dargestellten Elemente repräsentieren dabei die einzelnen Objekte im Dokumentenbaum und können jeweils zusätzliche Attributs- und Textknoten besitzen, die zum Zwecke der Übersichtlichkeit hier nicht dargestellt sind (Keith 2005).

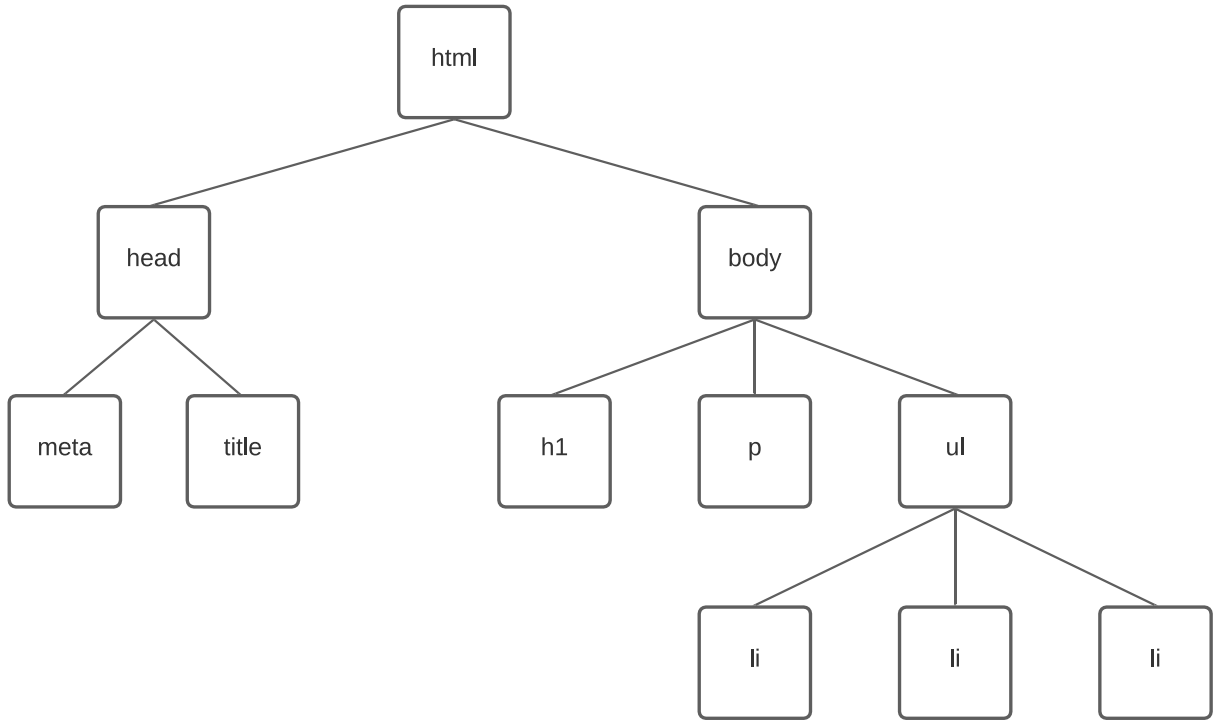


Abbildung 3: Struktur einer Webseite mithilfe des DOM (Keith 2005)

Um das in Abbildung 3 dargestellte Dokumentenmodell zu erzeugen ist nachfolgender HTML Text notwendig. Durch die Einrückung der Elemente wird deren Ebene im DOM sichtbar gemacht. Die Einrückungen dienen jedoch nur der Lesbarkeit des Dokuments und haben keinen Einfluss auf die Interpretation des Browsers (Keith 2005).

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="content-type" content="text/html charset=utf-8" />
        <title>Shopping list</title>
    </head>
    <body>
        <h1>What to buy</h1>
        <p title="a gentle reminder">Don’t forget to buy this stuff.</p>
        <ul id="purchases">
            <li>A tin of beans</li>
            <li>Cheese</li>
            <li>Milk</li>
        </ul>
    </body>
</html>
```

Listing 1: HTML Text einer beispielhaften Webseite (Keith 2005)

Wird das in Listing 1 angeführte HTML-Dokument von einem Browser geladen, wird der HTML Text interpretiert und die einzelnen Elemente des DOM werden visuell dargestellt. Abbildung 4 zeigt die visuelle Ausgabe eines Browsers. Diese kann allerdings je nach Browser variieren (Keith 2005).

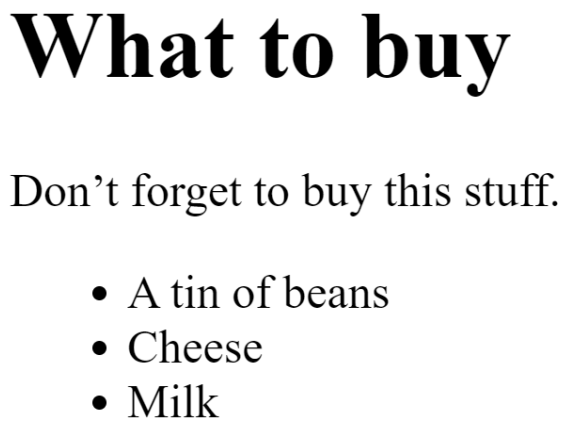


Abbildung 4: Darstellung eines HTML-Dokuments im Browser (Keith 2005)

Neben der inhaltlichen Strukturbeschreibung der Webseite gibt es auch eine Sprache zur Beschreibung der Formatierung der Inhalte. Hierzu kommen Cascading-Style-Sheets (CSS) im Web zum Einsatz. Mithilfe von CSS kann die Formatierung jedes einzelnen Elements im Dokumentenbaum definiert werden. Das CSS kann dabei in der HTML-Datei selbst oder in einer externen Datei der Webseite zur Verfügung gestellt werden (Meinel und Sack 2004).

2.2.2 Laden einer Webseite

Eine Webseite wird typischerweise mit einem Browser durch die Eingabe eines bestimmten Uniform-Resource-Locators (URL) aufgerufen. Dabei wird über dem URL der spezifische Webserver identifiziert und die Webseite mittels Hypertext-Transfer-Protocols (HTTP) zum Client übertragen. HTTP ist ein Kommunikationsstandard für die Abhandlung von Anforderungen und Antworten zwischen Client und Server. Die Aufgabe des Servers besteht darin, die Anfragen des Clients zu verarbeiten und diese bestmöglich zu beantworten. Im Falle des Aufrufens einer Webseite wäre der Webseitenaufruf die Anfrage zum Server, der das entsprechende Dokument in der Antwort mitliefert. Zwischen Client und Server können sich mehrere diverse Geräte wie Router, Proxys oder Firewalls befinden, die zur korrekten Übertragung der Anforderungen und Antworten beitragen. Weiters sind Webserver in der Lage, mehrere Verbindungen unterschiedlicher Clients zu verarbeiten. In der nachfolgenden Abbildung wird der grundlegende Ablauf eines Webseitenaufrufs skizziert und anschließend beschrieben. Dieser Ablauf wiederholt sich für alle Elemente der Webseite, wie Bilder oder sonstige Ressourcen, die vom Server geladen werden müssen (Nixon 2014; RFC 1945).

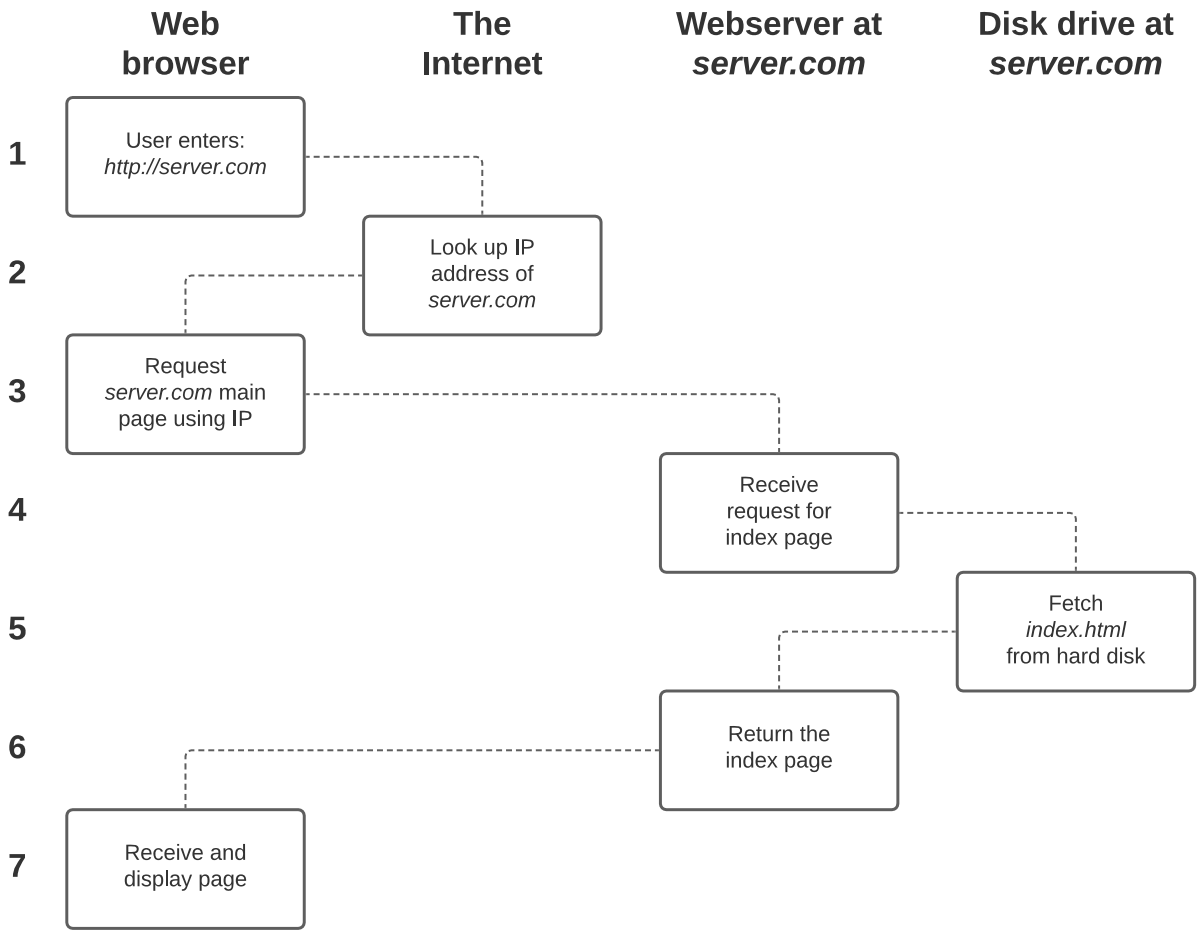


Abbildung 5: Laden einer Webseite mittels HTTP (Nixon 2014)

Wie in Abbildung 5 ersichtlich, erfolgt das grundsätzliche Laden einer Webseite in sieben Schritten, die in folgender Sequenz ablaufen (Nixon 2014):

1. Der URL wird in die Adresszeile des Browsers eingegeben.
2. Der Browser versucht, die dazugehörige IP-Adresse des URL aufzulösen.
3. Der Browser stellt eine Anfrage an den Server mittels der IP-Adresse.
4. Die Anfrage des Browsers wird durch das Internet zum Server geroutet.
5. Der Webserver verarbeitet die Anfrage und lädt die Webseite aus dem Speicher.
6. Die geladene Webseite wird als Antwort zum Browser retourniert.
7. Der Browser erhält die Antwort und stellt die enthalten Webseite dar.

Die in Schritt 2 gelistete Auflösung der IP-Adresse erfolgt durch einen Domain-Name-Service (DNS), der die Verknüpfungen zwischen Domänennamen und IP-Adressen verwaltet (Nixon 2014; RFC 1034).

2.2.3 Dynamischer Inhalt

Mit der klassischen Navigation über HTML kann zwischen einzelnen Webseiten gewechselt werden. Diese zeigen jedoch nur statischen Inhalt an, das bedeutet, dass die Webseite vom Server geladen wird und sich der Inhalt nicht verändert. Bei dynamischen Webseiten können sich die Inhalte dagegen aufgrund unterschiedlicher Kontexte oder Bedingungen laufend ändern. Bei dynamischen Webseiten wird in der Literatur auch von Webapplikationen gesprochen. Grundsätzlich gibt es folgende zwei Möglichkeiten, Inhalte dynamisch zu verändern. Diese werden nachfolgend erklärt (Doyle und Lopes 2008):

- serverseitige Webseitenprogrammierung
- clientseitige Webseitenprogrammierung

Bei der serverseitigen Webseitenprogrammierung werden Webseiten vor der Auslieferung zum Client vom Server dynamisch verändert. Dazu wird das HTML vom Webserver geparst und entsprechend der Funktionalität der Webapplikation verändert. Typische Funktionen einer serverseitigen Webseitenprogrammierung sind die Verwaltung der Benutzer*innensitzungen, das Verarbeiten von Formulareingaben, die Anwendung von Autorisierungsmechanismen und das Laden von Inhalten aus Datenbanken und externen Diensten. In der nachfolgenden Abbildung werden die grundlegenden Elemente der serverseitigen Webseitenprogrammierung grafisch skizziert (Doyle und Lopes 2008).

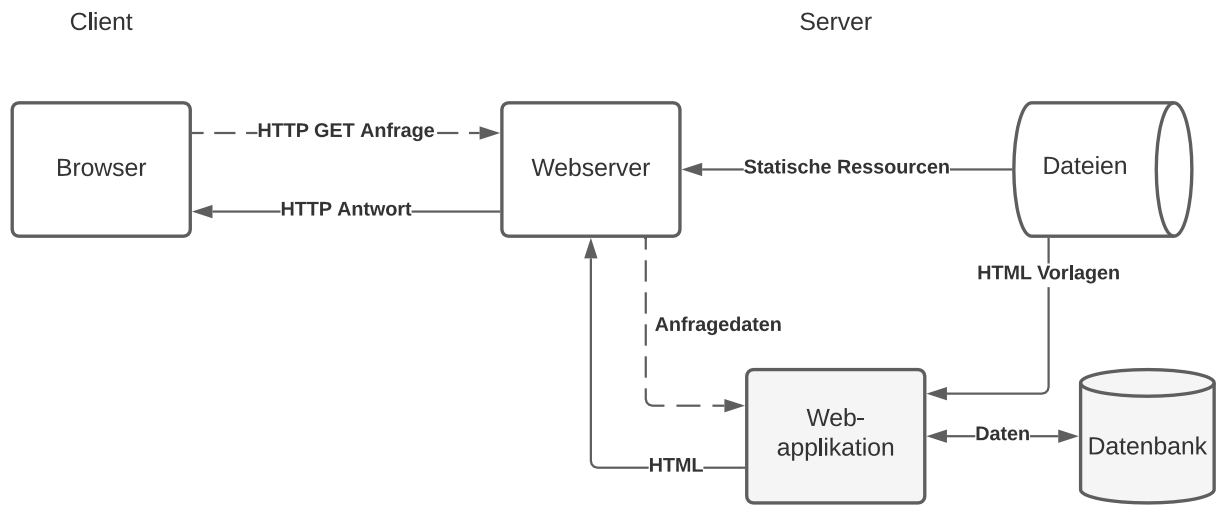


Abbildung 6: Serverseitige Webseitenprogrammierung (in Anlehnung an MDN Web Docs 2016)

Bei der clientseitigen Webseitenprogrammierung wird im Gegensatz zur serverseitigen der Inhalt direkt im Browser verändert. Der entsprechende Programmcode wird beim Laden einer Webseite vom Webserver mitgeliefert und vom Browser ausgeführt. Typischerweise wird dieses Verfahren dazu verwendet, auf Maus-, Tastatur oder Zeitereignisse zu reagieren und in weiterer Folge den Webseiteninhalt zu verändern. In der nachfolgenden Abbildung werden die grundlegenden Komponenten clientseitiger Webseitenprogrammierung grafisch dargestellt. Diese Vorgehensweise wird insbesondere beim SPA-Ansatz verwendet (Doyle und Lopes 2008).

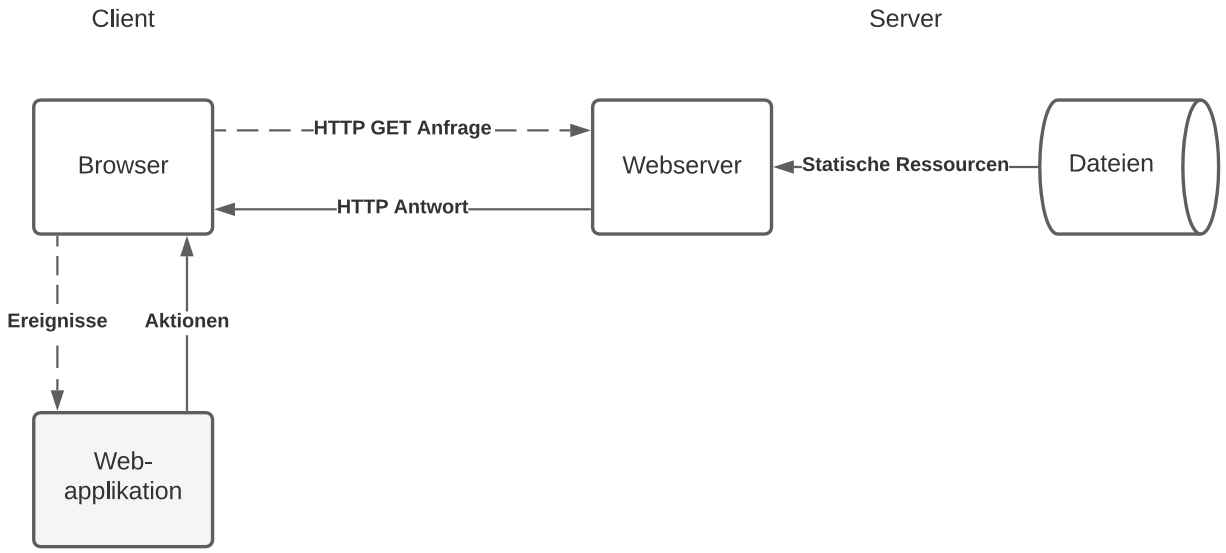


Abbildung 7: Clientseitige Webseitenprogrammierung (in Anlehnung an Tech Differences 2018)

2.3 Bewertung und Bewertungsmethoden

Grundlage für objektive Entscheidungen sind bewertete Handlungsalternativen. Nur durch den Vergleich von gewichteten Bewertungen kann eine Entscheidung möglichst frei von subjektiven Werthaltungen getroffen werden (Laux und Liermann 2005).

2.3.1 Bewertungskriterien

Objektivität, Reliabilität, Validität und Handhabbarkeit sind wichtige Kriterien bei empirischen Untersuchungen (Flick et al. 2004).

Eine Aussage einer empirischen Untersuchung ist dann objektiv, wenn sie keine Beeinflussung des Durchführenden aufweist und keinen Spielraum für die subjektive Interpretation zulässt (Ramb 2018).

Ein Maß der Zuverlässigkeit wissenschaftlicher Untersuchungen ist die Reliabilität. Sie gibt an, wie gut Messergebnisse unter gleichen Messbedingungen korrelieren. Die Reliabilität kann dabei Werte zwischen -1 und 1 annehmen. Je stärker der Korrelationskoeffizient gegen 1 geht, desto besser stimmen die Wiederholungsmessergebnisse überein (Flick et al. 2004).

Validität ist eine Eigenschaft, die die Gültigkeit einer wissenschaftlichen Untersuchung beschreibt, und ist dann gegeben, wenn mit der festgelegten Forschungsmethode tatsächlich die beabsichtigten Ergebnisse gemessen werden. Man versteht unter Validität also die Eignung der Messung bezogen auf die konkrete Zielsetzung (Flick et al. 2004).

Bei der Handhabbarkeit wird von einem Maß gesprochen, bei der bestimmte Ziele effektiv, effizient und zufriedenstellend erreicht werden (Flick et al. 2004).

2.3.2 Skalenniveau

Bei der Bewertung der einzelnen Kriterien in Kapitel 7 werden für unterschiedliche Merkmale verschiedene Skalen verwendet. Hier werden die Grundlagen für die Systematik der Skalen angeführt. Das Skalenniveau, das in der Literatur auch als *Messniveau* oder *Skalendignität* vorkommt, unterscheidet sich in den Skalen Nominalskala, Ordinalskala und Kardinalskala. Diese werden in der Abbildung 8 dargestellt und anschließend näher beschrieben. Je nach Skalenniveau können bestimme mathematische Operationen oder Transformationen durchgeführt werden. Durch das Skalenniveau können Informationen zum entsprechenden Merkmal und dazu, welche Interpretationen deren Ausprägungen zulassen, gewonnen werden. Die Information, ob ein Merkmal diskret oder stetig ist, enthält das Skalenniveau jedoch nicht (Tachtsoglou und König 2017).

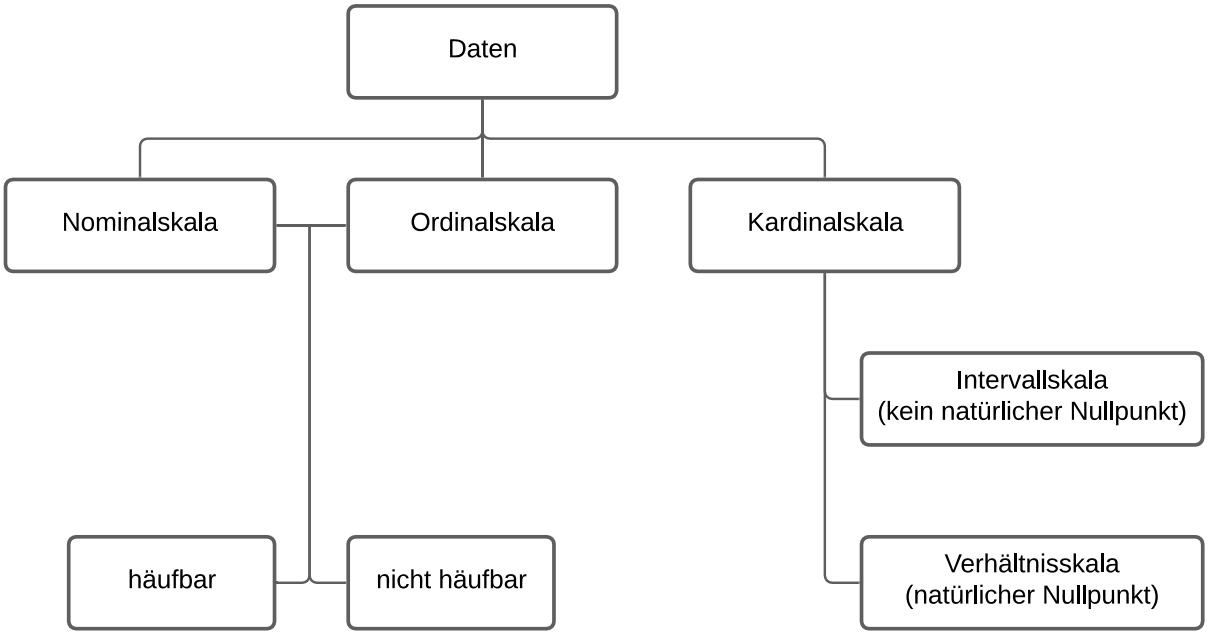


Abbildung 8: Einteilung der Daten anhand des Skalenniveaus (Tachtsoglou und König 2017)

Eine Nominalskala besitzt ein Merkmal dann, wenn dessen Ausprägungen zwar unterschieden werden können, jedoch keinen Rang aufweisen. Bei einem Vergleich zwischen verschiedenen Objekten kann also nur eine Entscheidung zwischen Gleichheit und Ungleichheit getroffen werden. Eine Sortierung der Werte nach Ausprägung ist bei nominal skalierten Werten nicht möglich, da alle Ausprägungen denselben Rang haben. Beispiele für nominal skalierte Werte sind Postleitzahl, Geschlecht, Beruf oder Farbe (Tachtsoglou und König 2017).

Bei der Ordinalskala besteht im Gegensatz zur Nominalskala eine Rangordnung zwischen zwei Merkmalswerten. Dadurch können Werte aufgrund von ordinal skalierten Merkmalen sortiert werden. Dies erfolgt dadurch, dass die Ausprägung eines Merkmals mehr, größer, stärker, weniger, kleiner oder schwächer als die Ausprägung eines anderen Merkmals ist. Die Abstände zwischen den Ausprägungen können aber nicht interpretiert werden, da beispielsweise bei Schulnoten der Abstand zwischen 1 und 2 nicht dem Abstand zwischen 4 und 5 entspricht. Beispiele für Merkmale mit einer Ordinalskala sind Schulnoten, Steuerklassen oder Sternebewertungen (Tachtsoglou und König 2017).

Die Kardinal- oder auch metrische Skala besitzt ein Merkmal dann, wenn dessen Ausprägungen eine natürliche Reihenfolge und quantifizierbare Abstände besitzen. Mit diesen Werten können mathematische Operationen durchgeführt werden. Die Kardinalskala wird weiters in Intervall- und Verhältnisskala unterteilt. Diese unterscheiden sich darin, dass Letztere im Gegensatz zur Ersteren einen natürlichen Nullpunkt aufweist. Eine Kelvin dargestellte Temperatur ist aufgrund des natürlichen Nullpunktes also verhältnis-skaliert, während eine Temperatur in Celsius aufgrund des nicht vorhanden natürlichen Nullpunkts intervall-skaliert ist. Beispiele für kardinal skalierte Werte sind Größe, Preis, Besucher oder Abstand (Tachtsoglou und König 2017).

Ein Skalenniveau kann auch dadurch unterschieden werden, ob es häufbar oder nicht häufbar sein kann. Häufbarkeit bedeutet, dass ein Merkmal mehrere Ausprägungen annehmen kann, während nicht häufbare Merkmale nur eine Ausprägung haben können. Nominal und ordinal-skalierte Daten können sowohl häufbar als auch nicht häufbar sein. Kardinal-skalierte Daten hingegen können nur eine Ausprägung haben und sind daher nie häufbar (Bourier 2018).

2.3.3 Bewertungsmethoden

In der Literatur gibt es zahlreiche Bewertungsmethoden und -verfahren. In der folgenden Aufzählung werden die in der Literatur vorkommenden gängigen Methoden angeführt:

- ABC-Analyse
- Nutzwertanalyse
- Checklisten
- Portfolio-Analyse
- SWOT-Analyse
- Ja-Nein-Abfragen

Die ABC-Analyse ist ein Verfahren zur Schwerpunktbestimmung bei Planungen und Untersuchungen, dessen Zweck es ist, einen hohen Nutzen mit möglichst geringem Aufwand zu erzielen. Dies wird durch die Trennung von Wesentlichen und Unwesentlichen ermöglicht. Dies erfolgt in drei Teilbereichen, die in absteigender Reihenfolge mit den Buchstaben A, B und C beziffert werden. Die ABC-Analyse ist einfach durchführbar und kann in der Praxis für unterschiedliche Anwendungsbereiche eingesetzt werden. Die nachfolgende Grafik zeigt die visuelle Darstellung einer beispielhaften ABC-Analyse (Cordts und Lensing 1992).

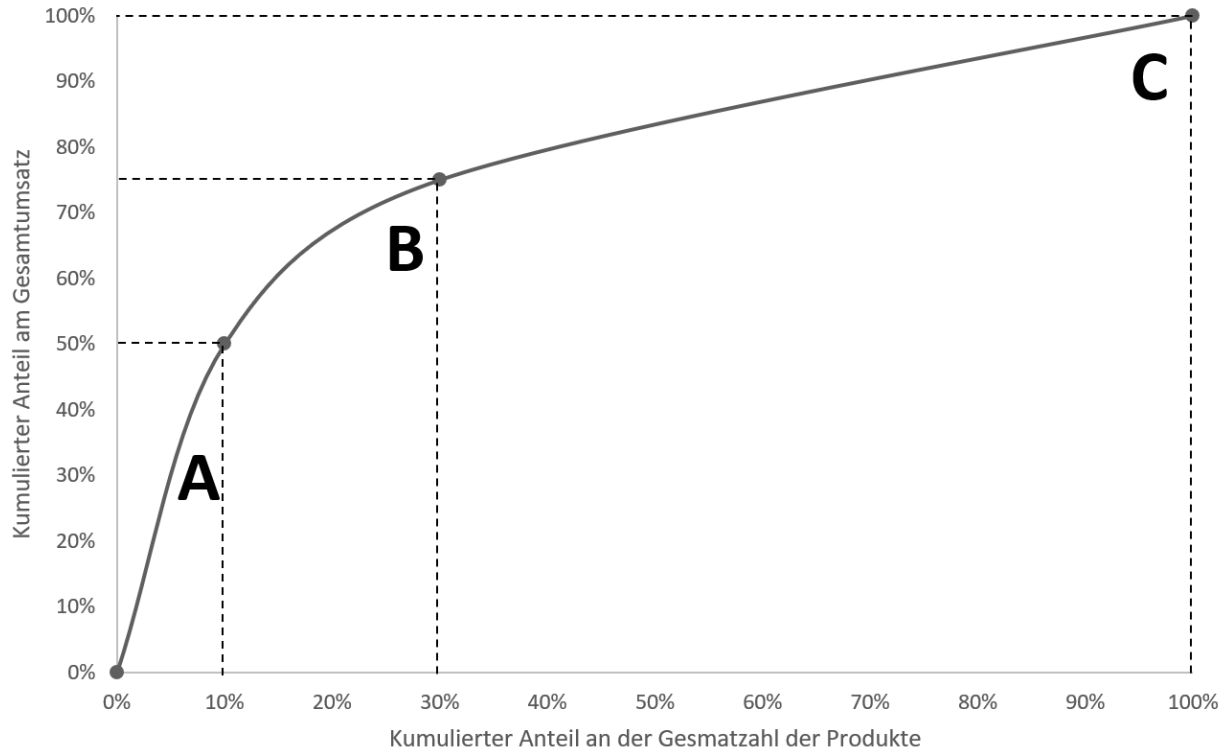


Abbildung 9: Visuelle Darstellung einer ABC-Analyse (Cordts und Lensing 1992)

Die Nutzwertanalyse ist eine Methode zur Analyse einer Menge komplexer Handlungsalternativen. Ziel der Nutzwertanalyse ist die Ordnung der Handlungsalternativen entsprechend der Präferenzen der Entscheidungsträger*innen. Die Ordnung erfolgt dabei aufgrund des Nutzwertes, der sich durch die Bewertung und Gewichtung mehrere Bewertungskriterien von einzelnen Handlungsalternativen ergibt (Zangemeister 2014).

Die Nutzwertanalyse ist in vier Phasen aufteilbar, die in Abbildung 10 grafisch dargestellt werden. Die erste Phase ist die Definition situationsrelevanter Ziele. Hierauf folgt die Beschreibung der zielrelevanten Konsequenzen der Alternativen. Anschließend wird die Bewertung der Alternativen aufgrund ihrer Konsequenzen durchgeführt. Abschließend wird die höchst bewertete Handlungsalternative als die geeignetste ausgewählt (Zangemeister 2014).

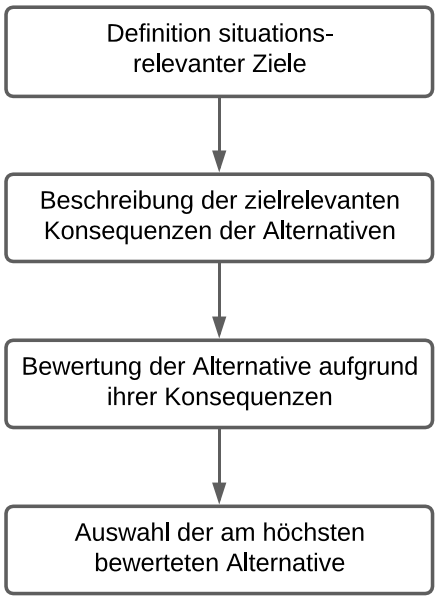


Abbildung 10: Phasen der Nutzwertanalyse (Zangemeister 2014)

In Checklisten werden Merkmale eines Gegenstandes schriftlich erfasst. Hierdurch wird verhindert, dass Teilaspekte vergessen oder versehentlich bzw. absichtlich übergangen werden. Da Checklisten keine hohen Anforderungen an die Anwender*innen stellen, werden sie in der Praxis häufig verwendet (Schneck et al. 2015).

Die Portfolio-Analyse, auch Vier-Felder-Matrix genannt, ist eine Technik zur Formulierung von Unternehmensstrategien. Grundlage ist dabei eine Umwelt- und Unternehmensanalyse. Erstere wird durch eine Chancen-Risiko-Analyse für das eigene Unternehmen durchgeführt. Bei der Unternehmensanalyse werden die Stärken und Schwächen des eigenen Unternehmens ermittelt. Für die Durchführung der Portfolio-Analyse werden die Geschäftseinheiten bestimmt und nach beliebigen Kriterien bewertet. Diese Kriterien, wie Marktanteil oder Marktwachstum, bestimmen die Grenzen im Portfolio. Abschließend werden die Geschäftseinheiten anhand der ermittelten Werte in das Portfolio eingetragen. Mithilfe der fertigen Portfolio-Analyse können Strategien für das langfristige Bestehen des Unternehmens abgeleitete werden. In der nachfolgenden Abbildung wird eine beispielhafte Portfolio-Analyse skizziert (Schawel und Billing 2009).

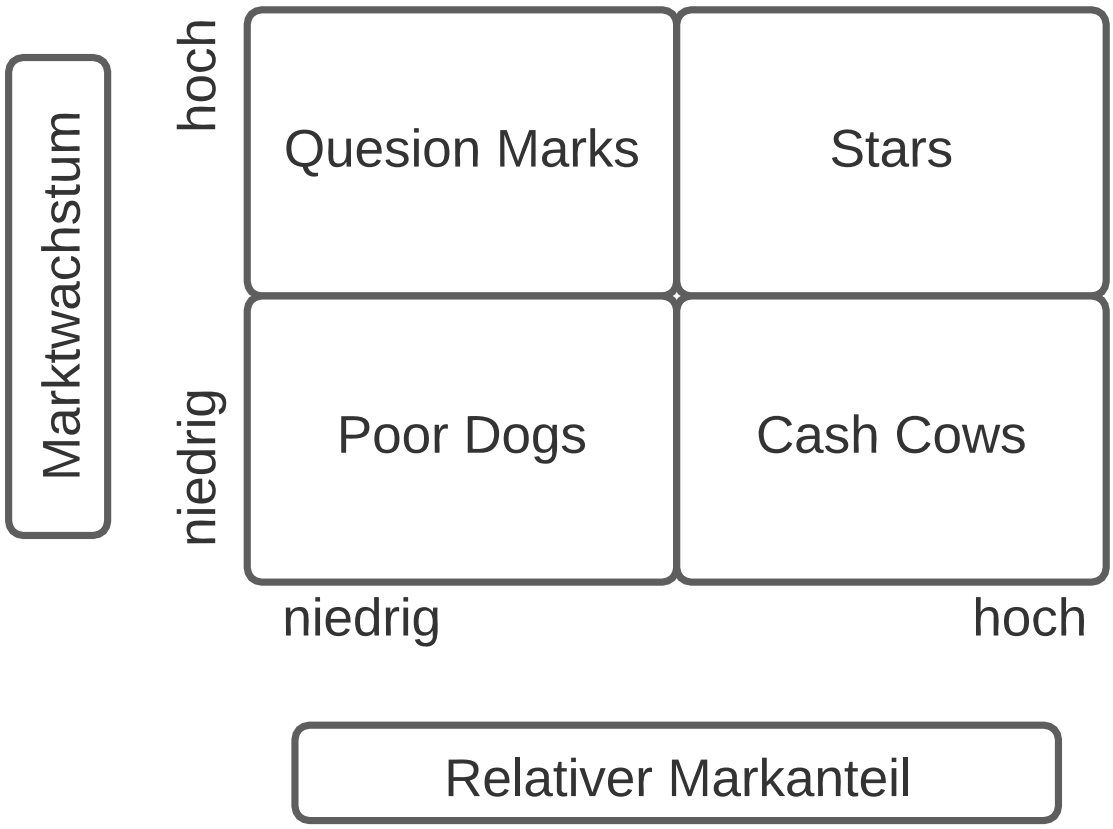


Abbildung 11: Skizzierung einer Portfolio-Analyse (Schawel und Billing 2009)

Die SWOT-Analyse ist eine Abkürzung und setzt sich aus den englischen Begriffen *Strengths*, *Weakness*, *Opportunities* und *Threats* zusammen und bedeutet übersetzt Stärken-Schwächen-Chancen-Risiko-Analyse. Sie dient dazu, die Positionierung der eignen Aktivitäten gegenüber dem Wettbewerb zu erfassen. Bei der Durchführung einer SWOT-Analyse werden im ersten Schritt die Stärken, Schwächen, Chancen und Risiken des Unternehmens gegenüber der Konkurrenz in Listenform erfasst. Anschließend werden die Stärken und Schwächen mit den Chancen und Risiken kombiniert, woraus Maßnahmen für das weitere strategische Vorgehen abgeleitet werden. Dabei werden je nach Chance oder Risiko Stärken ausgebaut oder abgesichert und Schwächen aufgeholt oder vermieden. In der nachfolgenden Abbildung wird die Kombination einer SWOT-Analyse grafisch dargestellt (Schneider 2019).

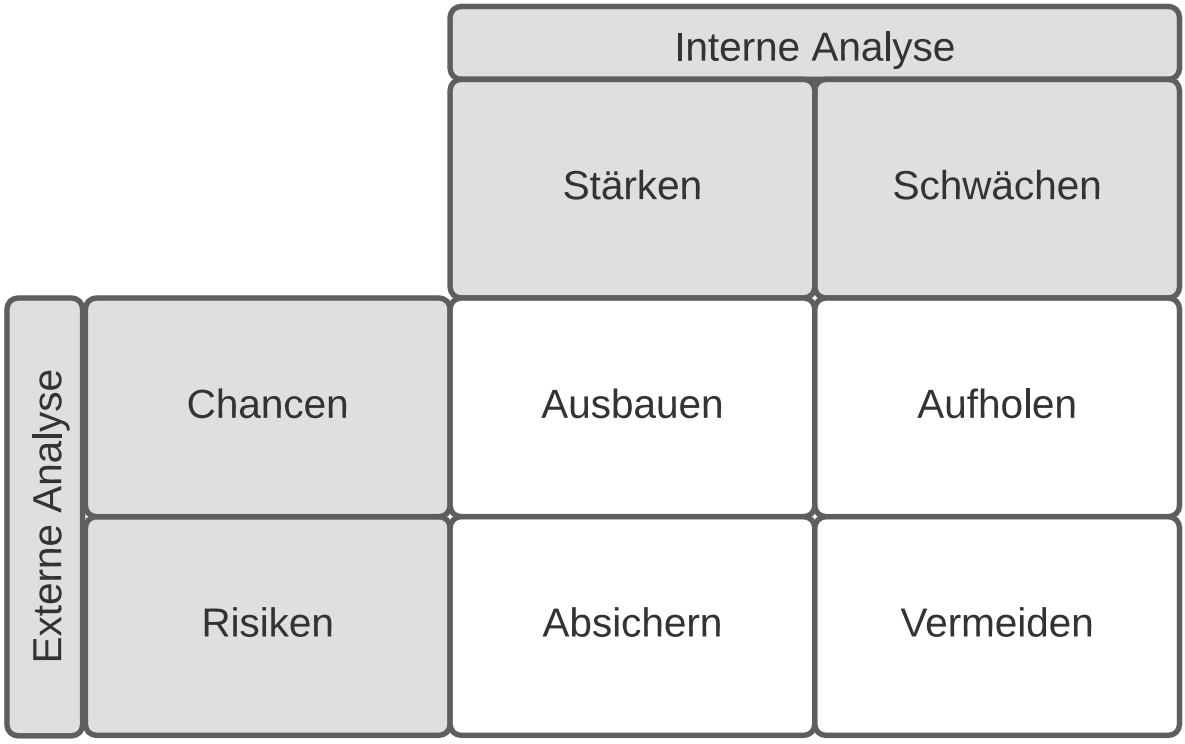


Abbildung 12: Darstellung der SWOT-Matrix (Schneider 2019)

Ja-Nein-Abfragen werden meist zur Vorfilterung verwendet, wobei einzelne Kriterien abgefragt werden. Dabei wird zwischen Muss und Soll-Kriterien unterschieden. Bei Nichterfüllung eines Muss-Kriteriums scheidet eine Handlungsalternative aus, während die Nichterfüllung von Soll-Kriterien die Präferenz der Alternative verringert. Durch die duale Bewertung sind Ja-Nein-Abfragen einfach handhabbar und führen zu eindeutigen Entscheidungen (Vahs und Burmester 2009).

2.4 Entscheidungstheorie

Nahezu ständig ist der Mensch in der Situation, allein oder in einer Gruppe, eine Entscheidung treffen zu müssen, wobei dies meist unterbewusst erfolgt. Die Folgen einer Entscheidung können dabei von existentieller Bedeutung sein und die Lebensbedingungen nachhaltig beeinflussen. Die Entscheidung selbst ist, wie in Kapitel 2.1.4 angeführt, die Auswahl einer aus mehreren Handlungsalternativen (Laux et al. 2012).

Da für verschiedene wissenschaftliche Disziplinen die Lösung und die Formulierung von Entscheidungsproblemen ein zentrales Thema darstellt, hat sich als interdisziplinärer Forschungsschwerpunkt die Entscheidungstheorie entwickelt (Laux et al. 2012).

2.4.1 Teilgebiete

Die Entscheidungstheorie wird in die folgenden drei Gebiete untereilt (Laux et al. 2012):

- Normative Entscheidungstheorie: basiert auf der Theorie der rationalen Entscheidung und auf normativen Modellen.
- Präskriptive Entscheidungstheorie: verwendet normative Modelle und versucht, durch die Herleitung von Strategien und Methoden bei der Entscheidungsfindung zu unterstützen.
- Deskriptive Entscheidungstheorie: Untersucht empirisch, wie Entscheidungen in der Realität tatsächlich getroffen werden.

2.4.2 Sicherheit und Unsicherheit

Entscheidungen können unter Sicherheit oder unter Unsicherheit getroffen werden. Ersteres bedeutet, dass der wahre Umweltzustand bekannt ist, während Letzteres bedeutet, dass die Umweltsituation nicht sicher bekannt ist. Bei einer Entscheidung unter Unsicherheit kann weiters unterteilt werden, ob die Eintrittswahrscheinlichkeit der Umweltsituation bekannt ist oder nicht. Ist sie bekannt, kann sie im Entscheidungsprozess mitberücksichtigt werden und man spricht von einer *Entscheidung unter Risiko*. Ist die Eintrittswahrscheinlichkeit der Umwelteinflüsse jedoch nicht bekannt, muss diesen jeweils dieselbe Wahrscheinlichkeit zugeordnet werden und es wird von einer *Entscheidung unter Ungewissheit* gesprochen. In der nachfolgenden Abbildung wird die Untergliederung von Entscheidungen anhand ihrer Sicherheit visuell dargestellt (Meyer 2000).

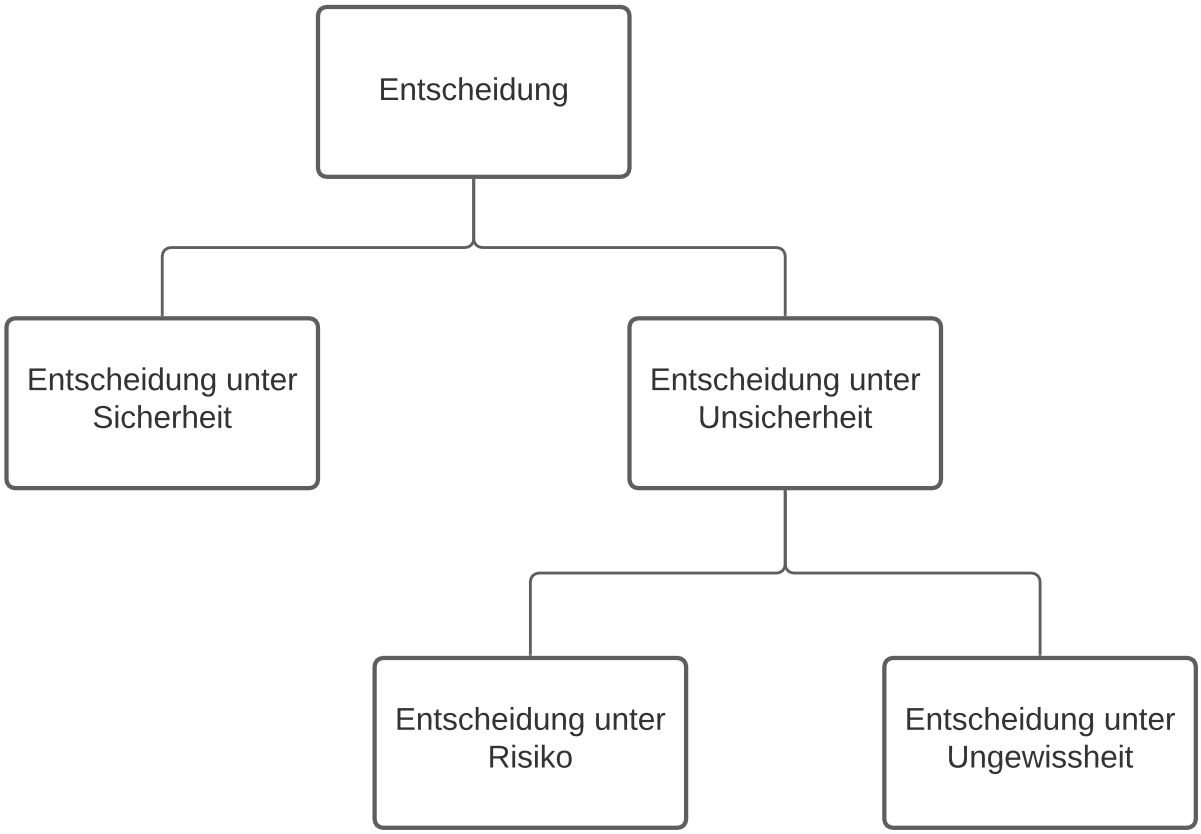


Abbildung 13: Sicherheit und Unsicherheit einer Entscheidung (Eigene Darstellung)

2.4.3 Entscheidungsmodelle

In Kapitel 2.1.6 ist die Definition des Begriffs Entscheidungsmodell und dessen Einsatzzweck erläutert. In diesem Kapitel werden vertiefende Grundlagen von Entscheidungsmodellen vermittelt. Dabei werden verschiedene Modelltypen angeführt und die wesentlichen Elemente eines solchen Modells erklärt.

Entscheidungsmodelle können in praktische und theoretische Modelle unterteilt werden. Unter Ersteren versteht man mathematische Modelle, die zur Lösung eines konkreten betrieblichen Entscheidungsproblems, das unmittelbar ansteht, entwickelt werden. Das Modell wird dabei auf eine einmalige Entscheidungssituation zugeschnitten. Es geht also bei der Formulierung eines praktischen Modells darum, aus einer vorliegenden Menge an realen Gegebenheiten unter Berücksichtigung von bestimmten Aspekten ein Formalkalkül abzuleiten, das dann über einen Lösungsprozess zu einem Ergebnis führt. Die nachfolgende Abbildung zeigt eine schematische Darstellung eines praktischen Entscheidungsmodells (Bitz 1977; Schupp 2004).

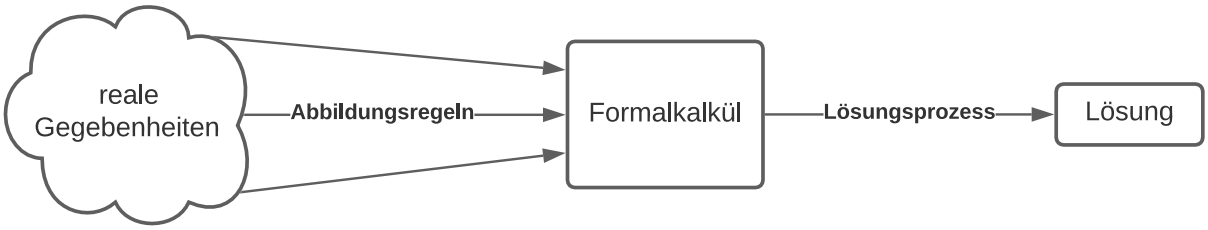


Abbildung 14: Schematische Darstellung eines praktischen Entscheidungsmodells (Bitz 1977)

Im Gegensatz zu praktischen Entscheidungsmodellen befassen sich theoretische nicht mit konkreten Entscheidungsproblemen, sondern mit hypothetischen Situationen. Das theoretische Modell hat daher keine komplexe Menge realer Gegebenheiten als Bezugsbasis, sondern ein überschaubares System mit hypothetischen Gegebenheiten. Durch diese Vereinfachung kann wesentlich einfacher ein Modell mit geringerer Komplexität für die gedachte Entscheidungssituation formuliert werden. In der nachfolgenden Abbildung wird das theoretische Entscheidungsmodell schematisch dargestellt (Bitz 1977; Schupp 2004).

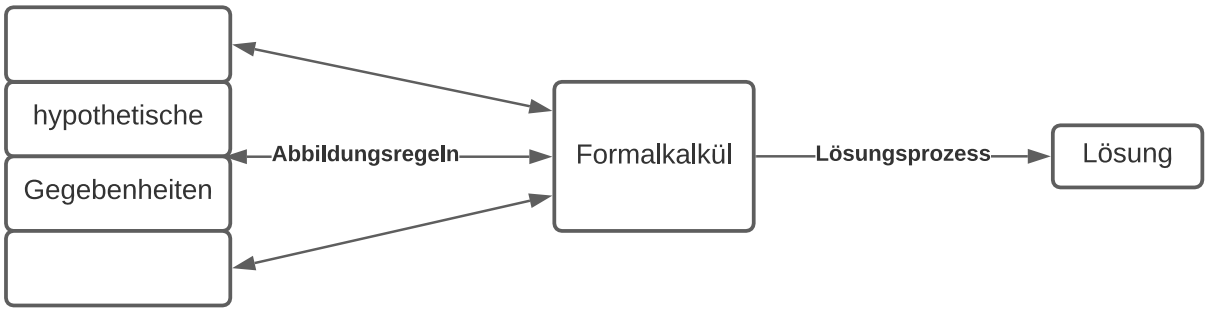


Abbildung 15: Schematische Darstellung eines theoretischen Entscheidungsmodells (Bitz 1977)

Um Entscheidungsmodelle für reale Entscheidungsprobleme zu formulieren, werden die realen Gegebenheiten vereinfacht, ehe ein theoretisches Modell formuliert wird. In der nachfolgenden Abbildung ist die Vereinfachung realer Gegebenheiten schematisch dargestellt (Bitz 1977; Schupp 2004).

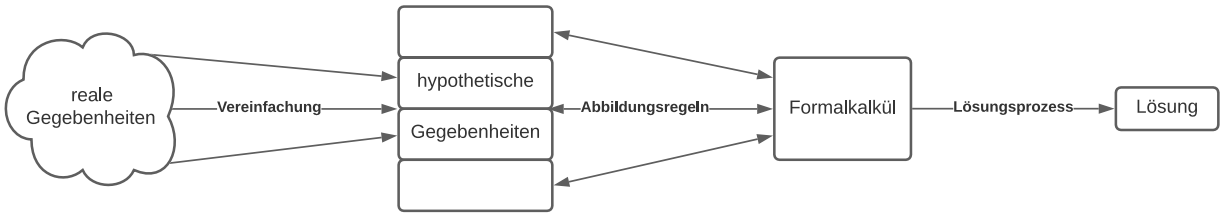


Abbildung 16: Entscheidungsmodell durch Vereinfachung realer Gegebenheiten (Bitz 1977)

Weiters können Entscheidungsmodelle in problemorientierte und lösungsorientierte Modelle unterteilt werden. Problemorientierte Modelle werden verschiedene Entscheidungssituationen aus der betrieblichen Praxis betrachtet und derartige Probleme in ein Modell abgebildet. Zusätzlich wird beim problemorientierten Ansatz versucht, die spezifisch vorliegenden Entscheidungsprobleme zu abstrahieren und daraus allgemeine Modelle zu entwickeln. Beim lösungsorientierten Ansatz hingegen werden primär bestimmte Lösungsverfahren betrachtet. Aus der Analyse von bekannten Algorithmen werden mathematische Formalstrukturen entwickelt und dadurch hypothetische Entscheidungssituationen abgeleitet. Die daraus gewonnenen Entscheidungssituationen können anschließend verwendet werden, um reale Entscheidungsprobleme zu lösen (Bitz 1977; Schupp 2004).

Jedes Entscheidungsmodell kann in die nachfolgenden wesentlichen Elementen gegliedert werden. Diese werden anschließend näher erläutert (Sztuka 2020):

- Handlungsraum
- Zustandsraum
- Ergebnisraum
- Ziele
- Präferenzen

Der Handlungsraum stellt alle möglichen Handlungsalternativen einer Entscheidung dar und sollte alle möglichen Handlungsalternativen beinhalten. Dazu zählen auch jene, die ungewöhnlich erscheinen. Die einzelnen Handlungsalternativen müssen dabei unabhängig voneinander sein und sich dadurch gegenseitig ausschließen (Sztuka 2020).

Durch den Zustandsraum wird das Umfeld für die Entscheidung beschriebene. Er ist notwendig, um die einzelnen Handlungsalternativen vergleichen zu können. Ähnlich dem Handlungsraum sollte auch der Zustandsraum vollständig sein und sich gegenseitig ausschließen, jedoch ist hier aufgrund der unendlichen Anzahl möglicher Zustände die Relevanz miteinzubeziehen (Sztuka 2020).

Werden die einzelnen Alternativen des Handlungsraums mit den einzelnen Umweltzuständen des Zustandsraumes verknüpft, resultieren mögliche Ergebnisse, die den Ergebnisraum darstellen. Hierin sind jedoch nur jene Ergebnisse zu betrachten, die die Entscheidungsträger*innen betreffen (Sztuka 2020).

Für jede Entscheidung ist die Definition eines Ziels notwendig. Dieses kann dabei von ökonomischer, sozialer oder persönlicher Natur sein. Beispiele für Ziele wären etwa maximaler Gewinn, maximale Mitarbeiter*innenzufriedenheit oder maximales Ansehen. Ziele können voneinander unabhängig, zueinander konkurrierend oder komplementär sein (Sztuka 2020).

Sind für das Entscheidungsmodell mehrere Ziele definiert, ist die Festlegung von Präferenzen der Entscheidungsträger*innen notwendig. Diese Präferenzen werden durch eine subjektive Bewertung der einzelnen Ziele festgelegt. Die Auswahl der Entscheidung erfolgt anhand jener Entscheidungsregeln, die die präferierten Ziele am besten Erfüllt (Sztuka 2020).

3 THEMENEINGRENZUNG

In diesem Kapitel wird die Abgrenzung der Forschungsfrage festgelegt. Dabei wird vom Gesamtthemenfeld mittels Eingrenzung von Detailtiefe und -breite die Eingrenzung für die Forschungsfrage festgelegt.

3.1 Gesamtthemenfeld

In nachfolgender Abbildung wird versucht, den Umfang des Gesamtthemenfeldes durch ein Baumdiagramm darzustellen. Dabei wird aber nur ein beispielhafter zufälliger Ausschnitt des realen Gesamtfeldes gezeigt. In der Realität besäße jeder Knoten des dargestellten Baums weitere Geschwister- und Subknoten.

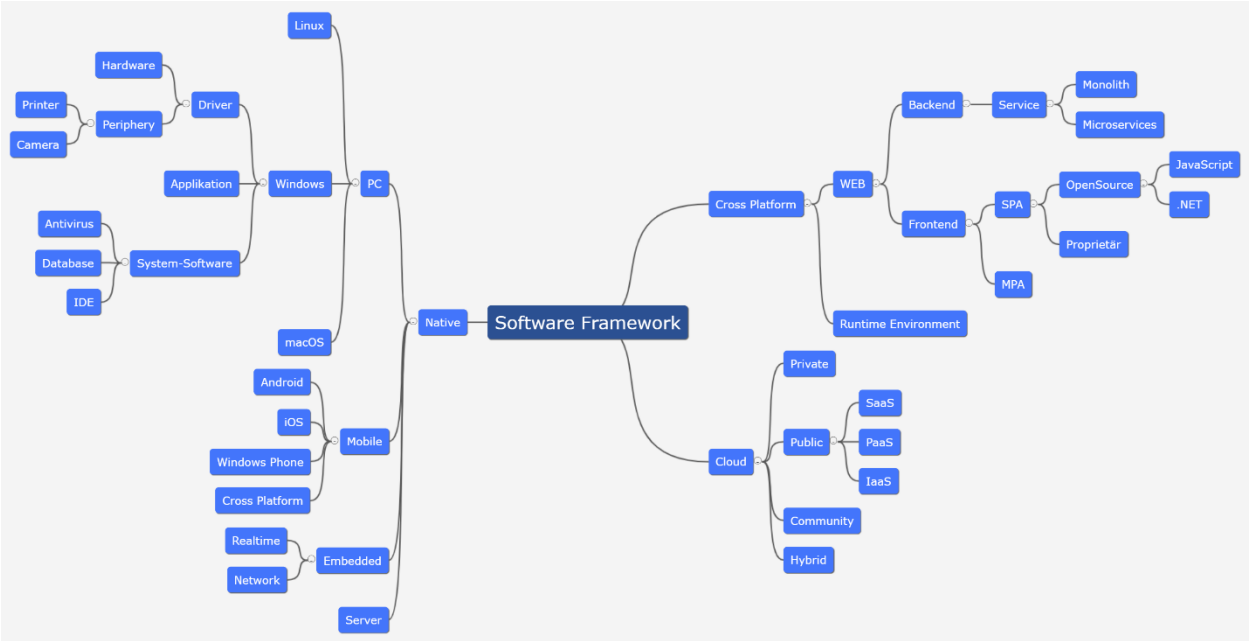


Abbildung 17: Überblick über das Gesamtthemenfeld (Eigene Darstellung)

3.2 Detailbreite vs. Detailtiefe

Um das in Abbildung 17 skizzierte Gesamtthemenfeld einzugrenzen, kann sowohl die Detailbreite als auch die Detailtiefe eingeschränkt werden. Durch die Einschränkung der Ersteren werden im Baum des Gesamtthemenfeldes Geschwisterknoten eliminiert, wodurch ein schmaler Baum mit wenig Verzweigungen entsteht. In nachfolgender Abbildung wird eine Eingrenzung der Detailbreite dargestellt.

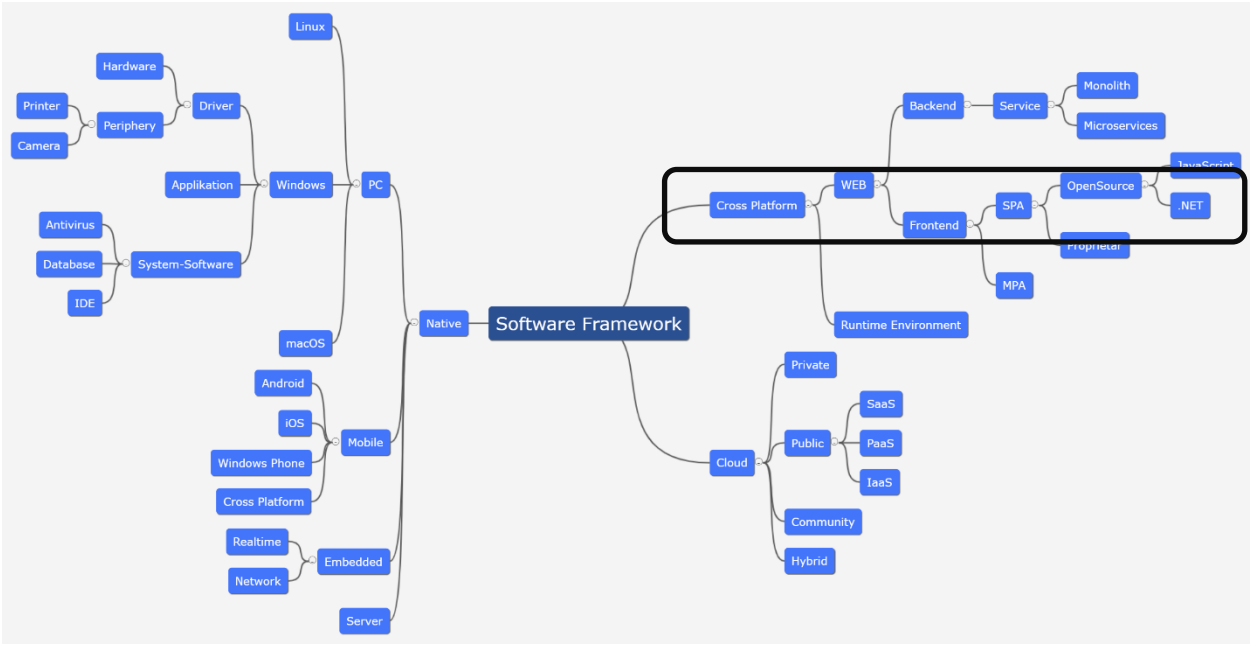


Abbildung 18: Eingrenzung der Detailbreite (Eigene Darstellung)

Bei der Eingrenzung der Detailtiefe wird die Anzahl der Subknoten beschränkt. Dabei wird die weitere Untergliederung von einzelnen Knoten nach einer bestimmten Anzahl von Subknoten unterbunden. In der nachfolgenden Abbildung wird die in Abbildung 18 eingegrenzte Detailbreite durch eine Eingrenzung der Detailtiefe erweitert. Die Schnittmenge aus eingegrenzter Detailbreite und -tiefe ergibt die Eingrenzung des Suchfeldes.

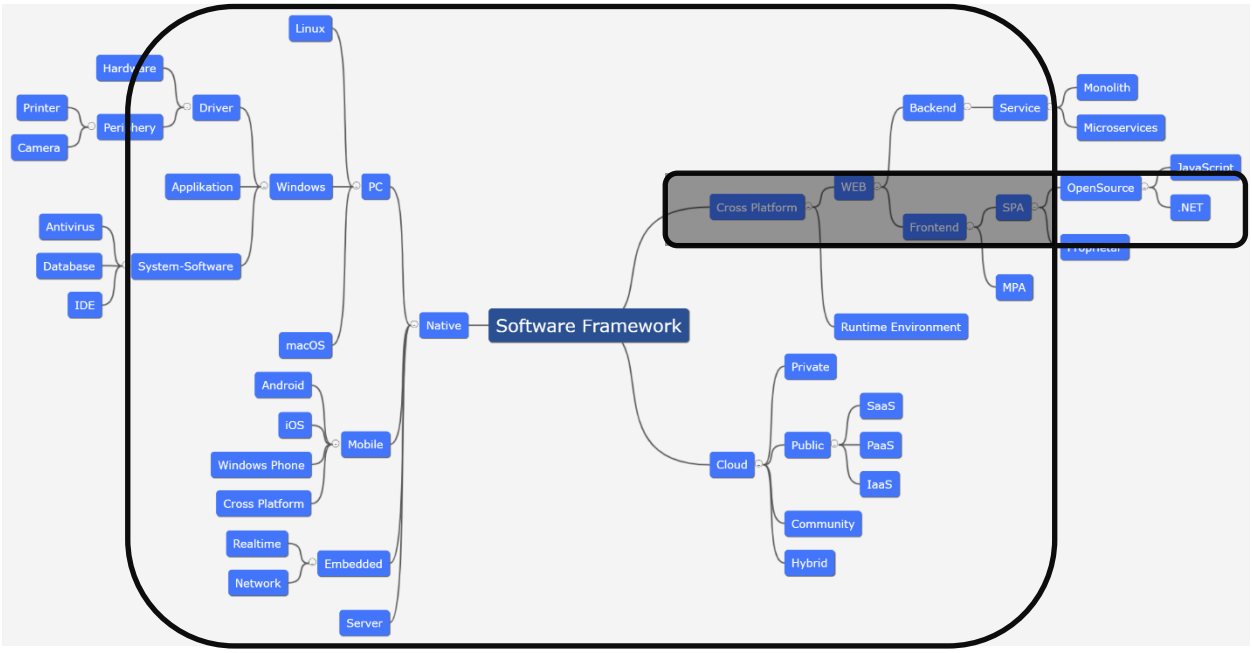


Abbildung 19: Eingrenzung der Detailtiefe (Eigene Darstellung)

3.3 Eingrenzung

Einer Umfrage zufolge, sind die meist verwendeten Technologien des Jahres 2020 JavaScript, HTML und CSS. Da alle drei Technologien für die Webentwicklung verwendet werden, lässt sich daraus schließen, dass von den meisten Entwickler*innen derzeit Webentwicklung betrieben wird. Diese Technologien werden insbesondere bei der Frontend-Entwicklung eingesetzt (Stack Overflow 2020).

Ein weiteres Merkmal, das Rückschluss auf die häufige Verwendung von Webtechnologien gibt, ist die kontinuierliche Zunahme der weltweiten Internetnutzer*innen. Laut Statistiken des Telekommunikationskonzerns Cisco nutzten bereits im Jahr 2018 mehr als die Hälfte der Weltbevölkerung das Internet. Für das Jahr 2023 werden 65 Prozent der Weltbevölkerung prognostiziert (Janson 2020).

Durch die Auswertung der zuvor angeführten statischen Merkmale erfolgt eine Eingrenzung auf den Bereich Web-Frontendentwicklung.

Aus mehreren Internetartikeln geht hervor, dass der Trend der eingesetzten Methode im Bereich der Web-Frontendentwicklung zu Single Page Applikationen (SPA) geht. Aus diesem Grund wird auch in dieser Arbeit das Thema auf SPA eingegrenzt (Aggarwal 2020; Ismail 2019; Sharma 2020; Kölpin 2017; Vesic und Minović 2015).

Die Forschungsfrage lautet somit nach Eingrenzung des Themenbereichs wie folgt:

Kann mithilfe eines systematischen Entscheidungsmodells die Auswahl eines geeigneten Single-Page-Application-Frameworks im Bereich Web-Frontendentwicklung getroffen werden?

4 FRAMEWORKS

In diesem Kapitel werden die für die Forschungsfrage relevanten SPA-Frameworks mittels Literaturrecherche gesucht und gefiltert. Anschließend werden die einzelnen Webframeworks beschrieben, die im folgenden Kapitel zur Entwicklung des Entscheidungsmodells herangezogen werden.

4.1 Filterung der Frameworks

Um den Umfang des Entscheidungsmodells kompakt zu halten werden in dieser Arbeit die Webframeworks nach Popularität gefiltert, wobei Letztere anhand bestehender Umfragen ermittelt wird. Gleichzeitig werden aufgrund der Themeneingrenzung nur jene Frameworks ausgewählt, die für die Entwicklung im Bereich Web und Frontend fokussiert sind.

Durch eine Umfrage wurden die beliebtesten Web-Frameworks des Jahres 2020 ermittelt. Die Umfrageergebnisse sind in Abbildung 20 in Diagrammform dargestellt. Die meistgenutzte Technologie ist laut Umfrage jQuery, wobei es sich aber nicht um ein Framework, sondern um eine Bibliothek handelt. Das am häufigsten verwendete Framework ist React.js gefolgt von Angular (Stack Overflow 2020).

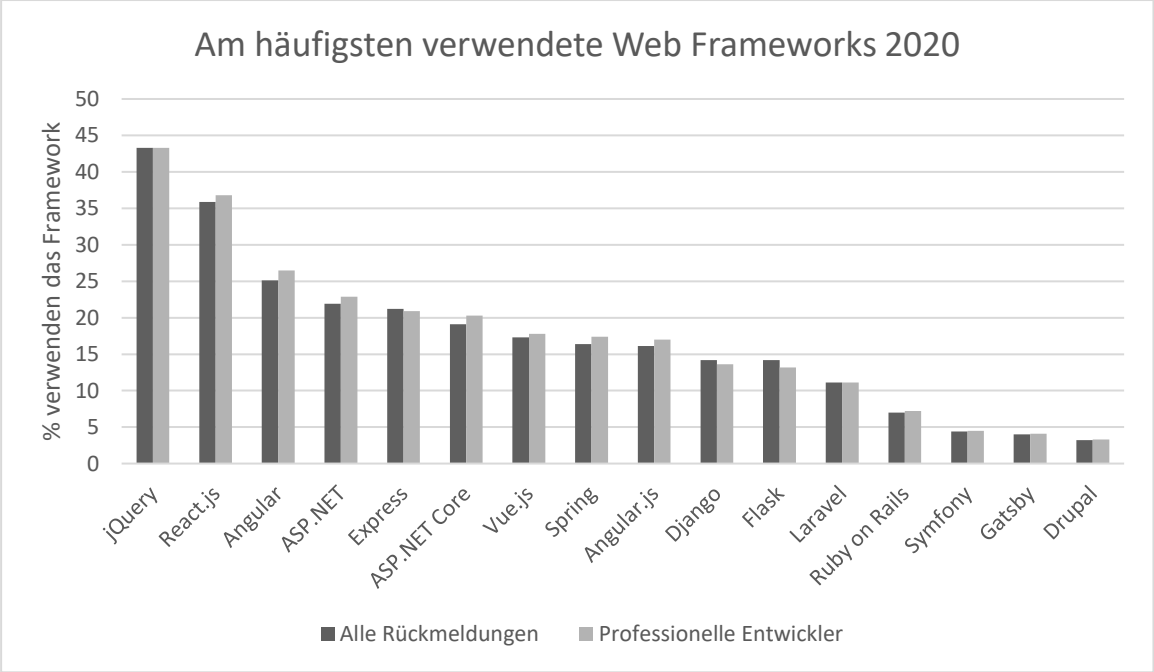


Abbildung 20: Am häufigsten verwendete Web Frameworks 2020 (Stack Overflow 2020)

Weiters wurde durch diese Umfrage die Beliebtheit von Web Frameworks erhoben. Die im nachfolgenden Diagramm dargestellten Balken geben an, wieviel Prozent der Umfrageteilnehmer*innen das Framework bereits verwenden und es auch weiterhin nutzen möchten (Stack Overflow 2020).

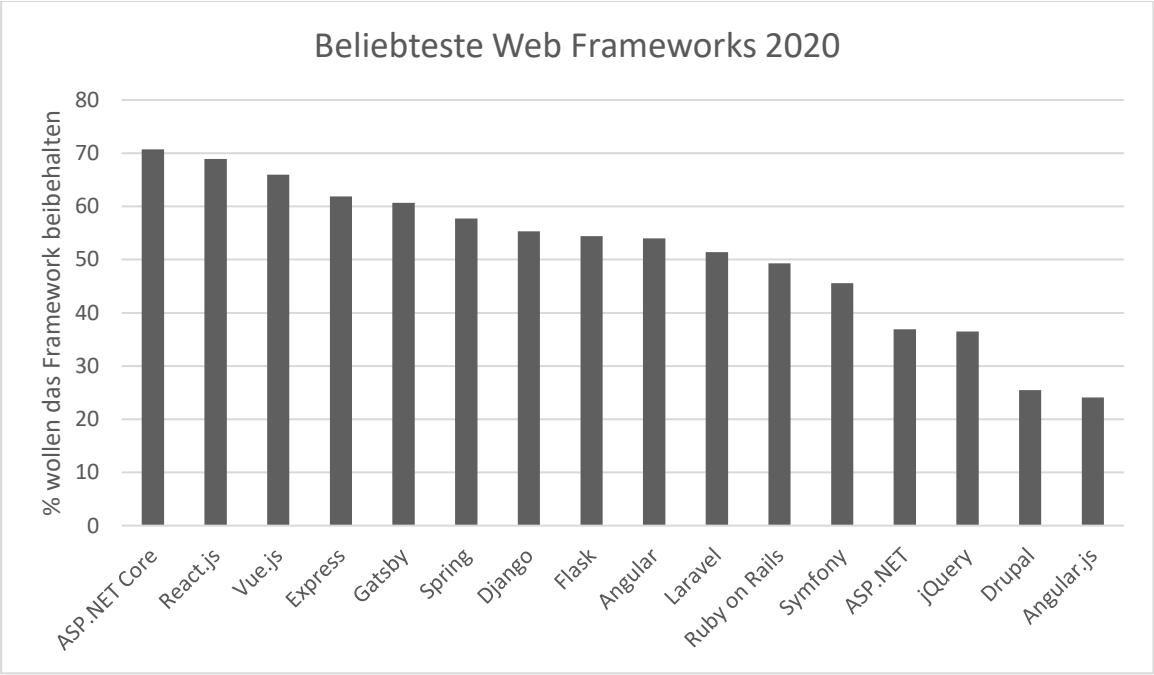


Abbildung 21: Beliebteste Web Frameworks 2020 (Stack Overflow 2020)

Im Umkehrschluss zu den beliebtesten wurden in der Umfrage auch die gefürchtetsten Web-Frameworks erhoben. Im Diagramm der Abbildung 22 wurden die Anteile in Prozent jener Umfrageteilnehmer*innen dargestellt, die ein Framework aktuell verwenden, zukünftig aber auf ein anderes Framework umsteigen wollen.

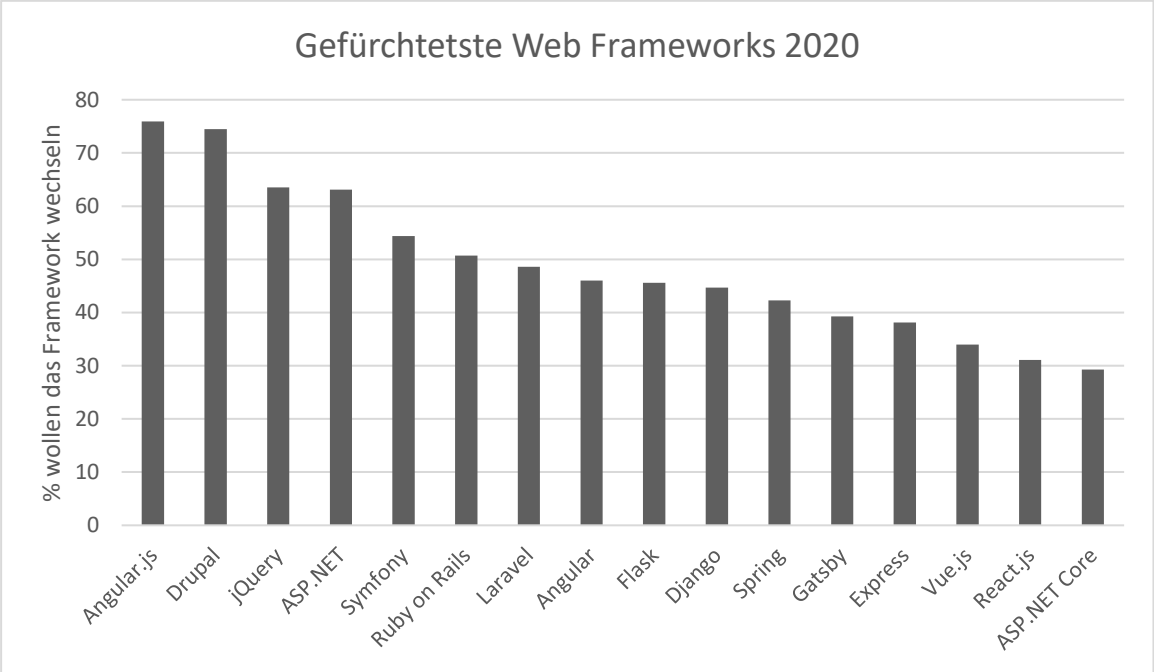


Abbildung 22: Meistgefürchtete Web-Frameworks 2020 (Stack Overflow 2020)

Zusätzlich wurden in der Umfrage jene Web-Frameworks erhoben, die von den Teilnehmer*innen aktuell noch nicht verwendet werden, an deren zukünftigen Einsatz aber Interesse besteht. Die Anteile der an den Frameworks interessierten Teilnehmer*innen werden im nachfolgenden Diagramm grafisch dargestellt.

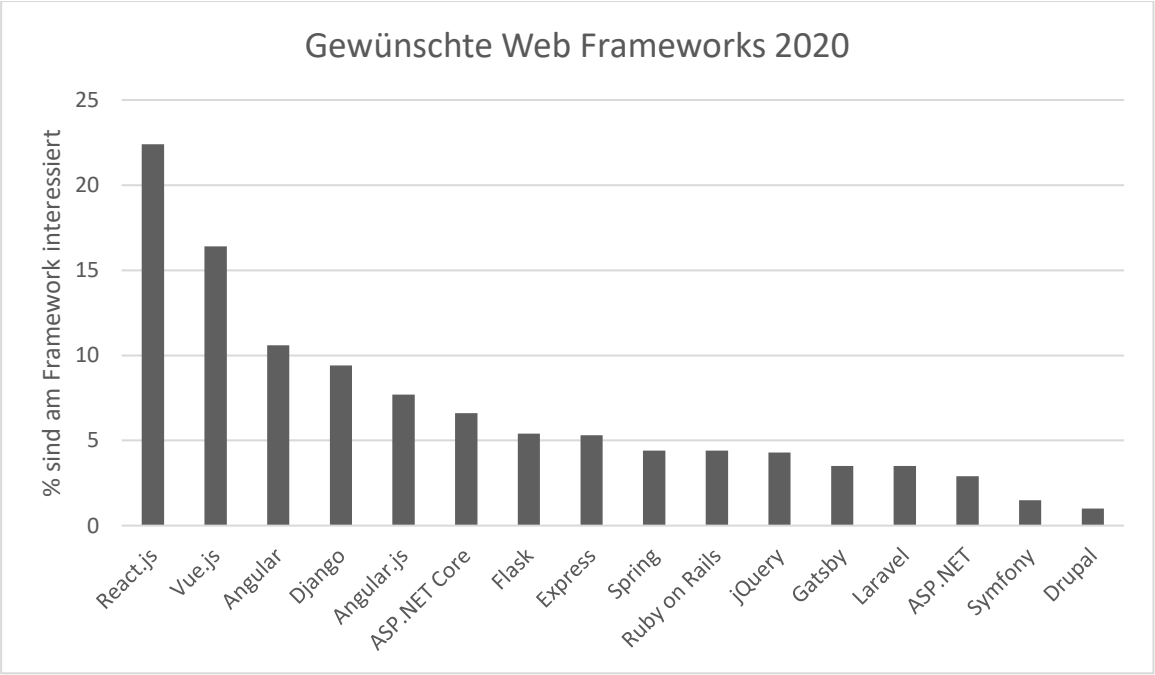


Abbildung 23: Meistgewünschte Web Frameworks 2020 (Stack Overflow 2020)

Aus den Umfrageergebnissen können zusammenfassend einige Schlüsse für das Jahr 2020 gezogen werden. Das gefürchtetste Framework ist bei den meisten Umfrageteilnehmer*innen Angular.js. Der aktuelle Nachfolger dieses Frameworks, Angular, ist jedoch sowohl unter den Top 3 der am meistverwendeten Web-Frameworks als auch unter den Top 3 der gewünschten Web Frameworks des Jahres 2020. Dies könnte bedeuten, dass viele Teilnehmer*innen möglicherweise auf den Nachfolger umsteigen möchten. Diese Interpretation könnte aber auch falsch sein, da die Verwender*innen von Angular.js auch auf ein anderes Framework umsteigen könnten.

Ähnliches lässt sich hinsichtlich der Frameworks von Microsoft feststellen. Obwohl ASP.NET noch häufiger in Verwendung ist als das neuere ASP.NET Core, sticht Letzteres als das beliebteste und am wenigsten gefürchtete Web-Framework des Jahres 2020 heraus. ASP.NET stellt hingegen das viertgefürchtetste Web-Framework dar.

Das Framework React sticht aus den Umfrageergebnissen dadurch hervor, dass es das meistgewünschte und nach jQuery am häufigsten verwendete Web-Framework des Jahres 2020 ist. Zusätzlich besetzt es den zweiten Platz der beliebtesten Web-Frameworks 2020 und steht an vorletzter Stelle der gefürchtetsten Frameworks. Zusammengefasst kann gesagt werden, dass viele Entwickler*innen das Framework aktiv nutzen und dies weiterhin tun wollen, während jene, die das Framework noch nicht verwenden, daran interessiert sind.

Vue.js ist nach ASP.NET Core und React das drittbeliebteste und das am drittwenigsten gefürchtete Framework. Dies legt nahe, dass die Entwickler*innen, die dieses Framework verwenden, gerne damit arbeiten und es daher weiterverwenden möchten. Nebenbei schafft das SPA-Framework Interesse für potenzielle neue Verwender*innen des Frameworks. Aus der Umfrage geht hervor, dass Vue.js das von den Teilnehmer*innen am zweitmeisten gewünschte Web-Framework ist.

Aufgrund dieser Umfrageergebnisse werden nachfolgende SPA-Web-Frameworks als die vier für das Entscheidungsmodell relevantesten ausgewählt. Da ASP.NET Core jedoch kein SPA-Framework ist, wird stattdessen das integrierte SPA-Framework Blazor betrachtet:

- Angular
- React
- Vue.js
- Blazor

4.2 Angular

Das Front-End-Webapplikationsframework Angular ist der Nachfolger von AngularJS und basiert auf der Programmiersprache TypeScript. Die Open-Source-Software wird von Google, unterstützt von einer Community aus Unternehmen und einzelnen Personen, entwickelt und veröffentlicht (Böhm 2017).

Als Kernfunktionen des Produkts werden auf der offiziellen Webseite die folgenden vier (sinngemäß übersetzten) aufgelistet und anschließend näher erklärt (Google 2020b):

- plattformübergreifend
- Geschwindigkeit und Performanz
- Produktivität
- vollständiger Entwicklungsprozess

Angular schafft durch die optionale Entwicklung von Progressive-Web-Apps (PWA) die Möglichkeit, Webapplikationen ohne Installation ähnlich einer nativen App performant und offline zu benutzen. Weiters ist möglich, nativ mobile oder desktopbasierte Applikationen mit den Methoden der klassischen Webentwicklung zu entwickeln und dabei native Funktionen zu nutzen (Google 2020b).

In puncto Geschwindigkeit und Performanz gibt die Framework-Community an, durch die Optimierung des geschriebenen Codes auf aktuelle JavaScript-Umgebungen die Geschwindigkeit zu verbessern. Durch das Aufteilen der Komponenten kann der Code bei Bedarf in Teilpaketen geladen werden, was die Ladezeit der einzelnen Komponenten verkürzt (Google 2020b).

Um die Produktivität bei der Entwicklung mit Angular zu gewährleisten, kommen Templates zum Einsatz, die eine schnelle Erstellung von User-Interfaces (UI) ermöglichen. Mit dem Angular-Command-Line-Interface (CLI) ist es mit wenigen Befehlen möglich, Projekte zu erstellen, zu erweitern, zu testen und auszuliefern. Beliebte Entwicklerumgebungen bieten für das Framework eine Code-Vervollständigung, sowie eine Fehlererkennung und sonstiges Feedback an (Steyer und Schwab 2017).

Durch die Verwendung von Test-Frameworks in Verbindung mit Angular wird die Stabilität des Softwareprojekts verbessert, was dazu beiträgt, den Entwicklungsprozess zu vervollständigen (Google 2020b).

4.3 React

React ist ein Framework für die Programmiersprache JavaScript und stellt ein Grundgerüst für User-Interface-Komponenten zur Verfügung. Entwickelt wurde das Framework ursprünglich von Facebook, später wurde es jedoch als Open-Source-Projekt veröffentlicht und wird heutzutage von der Community weiterentwickelt (Facebook Inc. 2020).

Ziel der Entwicklung des Frameworks war das Anzeigen von Daten in Benutzeroberflächen. Dies ist grundsätzlich auch ohne Framework problemlos umsetzbar, jedoch war die Herausforderungen von Facebook die laufende Änderung der Daten, die mit der Entwicklung von React wesentlich vereinfacht wurde (Gackenheimer 2015).

Bei der Entwicklung dieses Frameworks wurde im Gegensatz zu anderen SPA-Frameworks der Funktionsumfang stark eingegrenzt. Primär grenzt sich React dadurch ab, dass es nicht den klassischen Model-View-Controller- (MVC-)Ansatz verfolgt. Beim diesem werden die Zuständigkeiten des Frontend in den drei Komponenten aufgeteilt und verfügen über bidirektionale Schnittstellen zwischen den Komponenten. React hingegen kümmert sich nur um die Anzeige der Benutzeroberflächen, in denen von außen über unidirektionale Datenbindungen die Daten verknüpft werden können. Dadurch soll React im Vergleich zu MVC-Frameworks besser dazu in der Lage sein, skalieren zu können (Gackenheimer 2015).

Trotz des eingeschränkten Funktionsumfangs ist das Framework ohne Notwendigkeit zusätzlicher SPA-Frameworks dazu in der Lage, vollständige Single-Page-Applications zu betreiben. Dennoch kann React auch in einzelnen Komponenten eines anderen SPA-Frameworks verwendet werden (Gackenheimer 2015).

4.4 Vue.js

Das JavaScript-Framework Vue.js wurde im Jahr 2014 vom Entwickler Evan You als Open-Source-Projekt veröffentlicht. Es ist primär auf die Entwicklung von Single-Page-Webapplikationen konzipiert, kann aber auch in mehrseitigen Webanwendungen verwendet werden. Im Vergleich zu anderen Webframeworks ist Vue.js dafür bekannt, dass es aufgrund der einfach gehaltenen Struktur schnell erlernbar ist und dass außer den Basiskenntnissen für Webentwicklung die Dokumentation ausreichend ist. Dadurch, dass das Framework inkrementell und progressiv implementiert werden kann, kann das Ökosystem zwischen einer einfachen Bibliothek bis hin zum vollumfänglichen Webframework skaliert werden. Das Framework selbst kann bereits mit 20 Kilobyte in die Webapplikation integriert werden, was ein schnelles Laden des Frameworks durch den Browser ermöglicht (You 2020).

Grundsätzlich ist Vue.js für die Programmiersprache JavaScript konzipiert, jedoch wird auch die von Microsoft entwickelte Programmiersprache TypeScript unterstützt. Deren Nutzung beinhaltet neben einer Typisierung von Variablen auch die objektorientierte Programmierung (OOP) durch echte Klassen, Vererbung, Interfaces und Namensräume. Seit dem JavaScript-Standard ECMAScript 6 sind diese Konzepte teilweise auch bei der Verwendung von JavaScript vorhanden, jedoch muss hierbei die Browserkompatibilität bei älteren Browsern beachtet werden. Der Nachteil von TypeScript besteht darin, dass es im Gegensatz zu JavaScript für die Ausführung im Browser transpiliert werden muss (Steyer 2019).

4.5 Blazor

Blazor ist ein Web-Frontend-Framework von Microsoft, das im Jahr 2018 als Open-Source-Framework veröffentlicht wurde. Primär ist das Framework dafür gedacht, Webapplikationen mit HTML und der Programmiersprache C# zu entwickeln, jedoch können auch JavaScript-Funktionen verwendet werden. Aktuell bietet das Framework zwei Hosting-Modelle an. Zum einen kann Blazor direkt im Browser mithilfe der durch WebAssembly (Wasm) ermöglichten .NET-Umgebung betrieben werden. Zum anderen lässt sich die Applikation serverseitig mithilfe des ASP.NET Core-Framework betreiben. In beiden Varianten bleiben die Applikation und deren Komponenten unverändert (Aponte 2020).

Wasm ermöglicht die Ausführung der Common-Language-Runtime (CLR) von .NET in allen modernen Browsern wie Edge, Chrome, Firefox und Safari ohne die Notwendigkeit zusätzlicher Plug-Ins. Da Wasm jedoch nicht direkt auf den DOM zugreifen kann, verwendet Blazor im Hintergrund JavaScript, das von Wasm und umgekehrt aufgerufen werden kann und auf gemeinsame Speicherbereiche zugreifen kann. Dadurch ist es möglich, über C#-Code auf den DOM zuzugreifen. Wasm ist ein standardisiertes, portables Binärformat für eine virtuelle Maschine und kann grundsätzlich überall laufen. Der Fokus richtet sich aber auf Browser (Microsoft 2020).

Einen erheblichen Vorteil bietet das Framework für .NET-Entwickler*innen, die bereits bestehenden C#-Code besitzen und diesen, ohne neu kompilieren zu müssen, im Framework verwenden können (Microsoft 2020).

5 KRITERIENKATALOG

Grundlage für eine Bewertung sind Bewertungskriterien. In diesem Kapitel werden die Kriterien für die Bewertung der SPA-Frameworks gesammelt und ein Kriterienkatalog erstellt.

5.1 Auswahl der Kriterien

Durch die nachfolgenden Kriterien sollen individuelle Anforderungen an das Framework möglichst umfänglich abgebildet werden, um bei der Auswahl des Frameworks auf individuelle Projektanforderungen einzugehen.

5.1.1 Programmiersprache

Es wird nominal bewertet, mit welchen Programmiersprachen das Webframework eingesetzt werden kann. Durch die Nominalskala werden alle Programmiersprachen ohne Rangordnung mit gleichem Wert behandelt.

5.1.2 Beliebtheit

Durch hohe Beliebtheit steigt das Interesse der Community, an der aktiven Weiterentwicklung des Frameworks. Gleichzeitig werden etwaige Probleme im Framework durch den hohen Verwendungsgrad schneller entdeckt. Erhoben wird die Beliebtheit anhand der Anzahl der Personen, die auf GitHub das Projekt mit einem Stern versehen haben.

5.1.3 Community

Die Community umfasst alle Entwickler*innen, die am Open-Source-Softwareprojekt des Frameworks beteiligt sind. Durch eine große Community wird die Weiterentwicklung des Frameworks stärker vorangetrieben und die Gefahr eines Entwicklungsstopps verringert. Gleichzeitig wird der Support meist durch die Community unterstützt. Für die Bewertung wird die Community durch die Anzahl der Contributors des Projekts erhoben, also jener Entwickler*innen, die außerhalb des Kernentwicklungsteams am Projekt Änderungen vornehmen.

5.1.4 Lernkurve

Mit der Lernkurve soll bewertet werden, wie schnell der Einstieg in das Framework gelingt. Ermittelt wird dies durch den Bau eines Prototyps mit dem jeweiligen Framework. Bewertet wird dies mithilfe einer ordinal skalierten Bewertung von einem bis fünf Sternen.

5.1.5 Lernunterstützung

Mit der Lernunterstützung wird ermittelt, wie gut Unterstützung für das Erlernen des Frameworks zur Verfügung steht. Als Unterstützung wird das Angebot an Einführungsunterlagen, Tutorials, Beispiele und Schulungen evaluiert. Das Angebot bezieht sich dabei auf das offizielle Material der Herausgeber des Frameworks, sowie Unterstützung durch externe Dienstleistungen wie beispielsweise angebotene Kurse durch Framework-unabhängige Online-Lernplattformen oder anderen Dienstleistungsunternehmen. Die Bewertung erfolgt anhand einer Skala von einem bis fünf Sternen.

5.1.6 Performanz

Um für den Anwender einer Webapplikation das Benutzererlebnis einer nativen Applikation zu erreichen ist Performanz notwendig. Webframeworks erreichen schnelle Ladezeiten und performante Reaktionen durch unterschiedliche Techniken wie Caching oder Lazy-Loading. Dennoch muss bei SPA-Frameworks auch das Framework selbst geladen werden, was die Performanz reduziert. Für die Bewertung der Frameworks wird dahingehend auf vorhandene Benchmarks zurückgegriffen und dadurch ein kardinalskalierter Index ermittelt.

5.1.7 Testbarkeit

Qualität ist in der Softwareentwicklung ein großes Thema und bildet sich aus der Summe mehrerer Qualitätsmerkmale. Typische Beispiele hierfür sind Funktionalität und Zuverlässigkeit. Beide Merkmale lassen sich am effizientesten mithilfe von Tests sicherstellen. Mit dem Kriterium der Testbarkeit wird mithilfe einer Sternebewertung von ein bis fünf Sternen bewertet, ob und wie gut der Quellcode, der auf Basis des jeweiligen Frameworks geschrieben wurde, testbar ist.

5.1.8 Dokumentation

Die Bewertung der Dokumentation erfolgt ordinalskaliert mittels einer Sternebewertung von einem bis fünf Sternen an. Die Ausprägung stellt dabei den Status von nicht vorhandener Dokumentation mit einem Stern bis hin zur optimalen Dokumentation mit fünf Sternen dar.

5.1.9 Installationsaufwand

Der Installationsaufwand soll bewerten, wie viel Zeit die Installation des Frameworks in Anspruch nimmt. Der Aufwand beinhaltet dabei Zeit und notwendige Tools. Dieser Aufwand wird anhand der Umsetzung der Prototypen ermittelt. Die Bewertung des Installationsaufwandes wird dabei mittels einer ordinalskalierten Sternebewertung von ein bis fünf Sternen durchgeführt.

5.1.10 Entwicklungsgeschwindigkeit

Die Entwicklungsgeschwindigkeit gibt an, wie schnell die Entwicklung einer SPA vorangeht. Dies wird durch die Unterstützung durch Vorlagen oder durch die automatische Generierung von Code beeinflusst. Ermittelt wird die Entwicklungsgeschwindigkeit der Frameworks anhand der Umsetzung der Prototypen und wird mittels einer Sternebewertung von einem bis fünf Sternen bewertet.

5.1.11 Funktionsumfang

Der Funktionsumfang gibt an, welche Funktionen das Framework zur Verfügung stellt. Bewertet wird dies durch eine Liste an Funktionen, die mit einer Ja-Nein-Abfrage beantwortet werden. Der Liste des Funktionsumfanges kann bei der Entscheidung als Grundlage für Soll und Muss-Kriterien verwendet werden und wird nachfolgend angeführt. Aufgrund der in der Softwareentwicklung einheitlich gebräuchlichen Wörter werden die Begriffe teilweise in englischer Sprache aufgelistet.

- Routing ohne Serveranfragen: Mithilfe des Frameworks kann ohne einen Aufruf zum Webserver zwischen einzelnen Seiten navigiert werden.
- Komponenten/Templates: Mit dem Framework können wiederverwendbar und abgekapselte Komponenten erstellt und mehrmalig verwendet werden.
- Datenbindungen unidirektional: Mit dem Framework können Variablen einer Komponente mit Attributen einzelner Elemente verknüpft werden.
- Datenbindungen bidirektional: Mit dem Framework können Variablen einer Komponente so verknüpft werden, dass der Datenfluss in beide Richtungen möglich ist.
- Ereignisbindungen: Ereignisse können mit Komponenten verknüpft werden, so dass auf bestimmte Ereignisse reagiert werden kann.
- Pipes: Mithilfe des Frameworks können sogenannte Pipes erstellt werden, die verknüpfte Daten automatisch verändern. Ein Beispiel dafür ist das Formatieren eines verknüpften Datumswert.
- Direktiven: Das Framework ermöglicht die Verwendung von benutzerdefinierten HTML-Tags oder Attributen.

- Dependency Injection: Das Framework ermöglich die Registrierung von Elementen oder Services und kann diese bei Bedarf automatisch in jeder Komponente zur Verfügung stellen.
- Internationalisierung: Mithilfe des Frameworks kann eine Webapplikation in Form von Übersetzungen internationialisiert werden.
- Formulare: Mithilfe des Frameworks können programmatisch Formulare für Benutzereingaben erzeugt und verwaltet werden.
- Validierung: Durch das Framework wird die Validierung von Benutzereingaben vereinfacht.
- Animationen: Durch das Framework werden gängige Animationen, wie das Ein- oder Ausblenden von Komponenten vereinfacht.
- HTTP-Client: Die Kommunikation mit externen Diensten über das HTTP-Protokoll für beispielsweise einer Datenabfrage wird durch das Framework vereinfacht.
- Logging: Das Framework bietet eine integrierte Logging-Funktionalität an, um bei der Entwicklung bessere Analysen zu erhalten.
- Lifecycle: Das Framework ermöglicht die Reaktion auf Ereignisse der Lebenszyklen der Webapplikation.
- Authentifizierung: Mithilfe des Frameworks wird die Authentifizierung für den Zugriff auf die Webapplikation vereinfacht.

5.1.12 Erweiterbarkeit

Die Erweiterbarkeit gibt an, ob und wie gut das Framework mit Funktionen durch Plug-Ins erweitert werden kann. Diese wird im Zuge der Entwicklung der Prototypen ermittelt und mittels einer Sternebewertung von ein bis fünf Sternen bewertet.

5.1.13 Browserkompatibilität

Die Browserkompatibilität gibt an, für welche Browser das Framework ab welchem Jahr kompatibel ist. Bei der Bewertung wird jenes Jahr angegeben, ab dem das Framework bei den Browsern Google Chrome, Mozilla Firefox, Apple Safari und Microsoft Edge funktioniert. Für die Zeit vor dem Aufkommen des Browsers Microsoft Edge wird dessen Vorgänger, Internet Explorer, zur Bewertung verwendet. Die Auswahl der Browser erfolgt aufgrund der darunterliegenden Statistik der meistgenutzten Browser weltweit (StatCounter Global Stats 2020).

5.1.14 Support

Beim Kriterium des Supports handelt es sich um die Unterstützung bei Fragen oder Problemen bezüglich des Frameworks. Hier wird ermittelt, über welche Kanäle Support erhältlich ist und mit welchen Kosten dies verbunden ist.

5.1.15 Personalbedarf

Durch das Kriterium des Personalbedarfs wird bewertet, wie hoch der Bedarf an Mitarbeiter*innen für das Framework seitens der Unternehmen ist. Dies wird durch die Erhebung der Stellenangebote für das jeweilige Framework auf der Jobplattform Devjobs.at ermittelt.

5.2 Wahl der Bewertungsmethodik

Einige Kriterien des zusammengestellten Kriterienkataloges sind messbare Fakten, die bei der Bewertung objektiv erfassbar sind. Der Kriterienkatalog beinhaltet zusätzlich Kriterien, die nicht direkt messbar sind und daher nicht vollkommen objektiv bewertet werden können. Um die Bewertung dieser Kriterien dennoch möglichst objektiv durchzuführen, erfolgt dies anhand prototypischer Projekte mit allen zu bewertenden Frameworks. Die Prototypen haben dabei den gleichen Funktionsumfang als Ziel und die Bewertung erfolgt anhand eines direkten Vergleichs zwischen den Frameworks.

Die Performanz eines Frameworks kann anhand eines Prototyps nicht ohne zusätzlichem Testaufbau für Geschwindigkeitstests bewertet werden. Weiters können durch eine Literaturrecherche keine Fakten zum direkten Vergleich der Frameworks erhoben werden. In Folge dessen wird für die Bewertung der Performanz der Frameworks auf vorhandene Studien zurückgegriffen und anhand dieser die Bewertung mittels Vergleich der Ergebnisse der Studien bewertet.

5.3 Überblickstabelle

In der Nachfolgenden Tabelle wird zusammenfassend ein Überblick über die zu bewertenden Kriterien, das dazugehörige Skalenniveau sowie die zu verwendende Bewertungsmethode aufgezeigt.

Kriterium	**Skalenniveau**	**Bewertungsmethode**
Programmiersprache	Nominal	Literaturrecherche
Beliebtheit	Kardinal	Erhebung
Community	Kardinal	Erhebung
Lernkurve	Ordinal	Prototyp
Lernunterstützung	Ordinal	Prototyp
Performanz	Kardinal	Benchmark
Testbarkeit	Ordinal	Prototyp
Dokumentation	Ordinal	Prototyp
Installationsaufwand	Ordinal	Prototyp
Entwicklungsgeschwindigkeit	Ordinal	Prototyp
Funktionsumfang	Nominal	Literaturrecherche
Erweiterbarkeit	Ordinal	Prototyp
Browserkompatibilität	Kardinal	Literaturrecherche
Support	Ordinal	Literaturrecherche
Personalbedarf	Kardinal	Erhebung

Tabelle 1: Bewertungskriterien mit Skalenniveau und Bewertungsmethode (Eigene Darstellung)

6 PROTOTYPEN

Wie in Kapitel 5 bereits erwähnt, wird für die Bewertung der nicht messbaren Kriterien ein Prototyp je Framework herangezogen, um anhand dieser die Frameworks möglichst objektiv zu vergleichen. Nachfolgend werden das Ziel, die Anforderungen und die Ergebnisse dieser Prototypen skizziert.

6.1 Ziel der Prototypen

Das Ziel der Entwicklung der Prototypen mit den einzelnen Frameworks ist die Bewertung der nicht messbaren Kriterien des in Kapitel 5 zusammengestellten Fragenkatalogs. Durch die Entwicklung eines Prototyps für jedes der ausgewählten Frameworks mit den gleichen Anforderungen können die Kriterien wie Entwicklungsgeschwindigkeit anhand eines direkten Vergleichs bewertet werden. Durch den direkten Vergleich wird die Objektivität gegenüber einer Bewertung ohne Prototypen erhöht. Durch die Erstellung der Prototypen werden dabei sowohl die Kriterien, die direkt, als auch indirekt mit der Entwicklung zusammenhängen, bewertet. Direkt mit der Entwicklung zusammenhängende Kriterien sind beispielsweise die Entwicklungsgeschwindigkeit oder die Testbarkeit des Frameworks, während Beispiele für indirekt zusammenhängende Kriterien die Dokumentation oder die Lernunterstützung sind.

6.2 Anforderungen der Prototypen

Die Anforderungen für die Entwicklung der Prototypen werden so definiert, dass alle durch den Prototyp zu bewertenden Kriterien des festgelegten Kriterienkatalogs bewertet werden können. Dies betrifft alle nicht messbaren Kriterien, die nachfolgend aufgeführt werden:

- Lernkurve
- Lernunterstützung
- Testbarkeit
- Dokumentation
- Installationsaufwand
- Entwicklungsgeschwindigkeit
- Erweiterbarkeit

Um die zuvor angeführten Kriterien zu bewerten, werden für die Erstellung der Prototypen folgende Aufgaben und Anforderungen gestellt:

- Erlernen der grundlegenden Funktionen des Frameworks anhand der angebotenen Lernunterstützung sowie der vorhandenen Dokumentation
- Entwicklung einer Webapplikation mit festgelegtem Funktionsumfang
- Anwendung von Erweiterungen des Frameworks in der Webapplikation
- Erstellung von Tests für den Quellcode der Webapplikation

Um den Umfang der Entwicklung der Prototypen einzuschränken, werden folgende Limitierungen festgelegt:

- Für das Erlernen der Frameworks wird nur auf kostenfreie Unterstützung zurückgegriffen
- Der für das Erlernen der Frameworks sowie für die Entwicklung der Webapplikationen notwendiger Funktionsumfang wird auf folgende Funktionen eingeschränkt:
    - Navigation ohne Serveranfragen
    - Auftrennen der Applikation in einzelne Komponenten
    - Mehrfachverwendung von Templates
    - unidirektionale Datenbindung auf HTML-Elemente
    - Datenaustausch zwischen einzelnen Komponenten

6.3 Entwurf der Prototypen

Für die Umsetzung der definierten Anforderungen wird ein Entwurf einer Webapplikation erstellt, der den Funktionsumfang erfüllt. Um die Funktion der Navigation ohne Serveranfragen zu implementieren, soll der Prototyp zwei Seiten beinhalten, zwischen denen über eine Navigationsleiste hin und her gewechselt werden kann.

Die Funktion der Mehrfachverwendung von Templates soll anhand eines Template für einen Zähler implementiert werden. Der Zähler besteht dabei aus einem Namen, einen Zählerstand sowie zwei Buttons zur Verringerung und Erhöhung des Zählerstandes um den Wert 1.

Das Template des Zählers soll dabei als eine eigene Komponente außerhalb der restlichen Applikation gespeichert werden. Dadurch wird die Funktion des Auftrennens der Applikation in einzelne Komponenten umgesetzt.

Jede Zählerkomponente besitzt einen eigenen Namen und einen eigenen Zählerstand. Die beiden Werte des Zählers werden in der Komponente als Variable gehalten. Diese Variablen sollen auf HTML-Elemente gebunden werden, sodass bei einer Änderung der Werte die Elemente automatisch aktualisiert werden. Dadurch wird die Funktion der Datenbindung auf HTML-Elementen implementiert.

Die Funktion des Datenaustausch zwischen einzelnen Komponenten wird in mehreren Variationen umgesetzt. Diesbezüglich sollen auf der Management Seite die Zähler mit Namen angelegt und wieder gelöscht werden können. Die angelegten Zähler sollen auf der Dashboard Seite darauffolgend automatisch aktualisiert werden, sodass für jeden angelegten Zähler eine Zählerkomponente dargestellt wird. Dies stellt einen Datenaustausch zwischen Komponenten zweier Seiten dar. Zusätzlich wird der Name des angelegten Zählers in die Komponente übergeben und darin angezeigt. Weiters soll in auf der Dashboard Seite die Anzahl sowie die Summe aller Zählerstände angezeigt werden. Der Datenaustausch mit der Zählerkomponente findet somit bidirektional statt.

Der Entwurf des Prototyps wird im ersten Schritt mithilfe eines sogenannten Mockups umgesetzt, das als Vorlage für die Entwicklung der Prototypen dienen soll. Dieser wird mit der Software Adobe XD erstellt. In der Abbildung 24 wird das Mockup der Dashboard Seite für den Prototyp dargestellt.

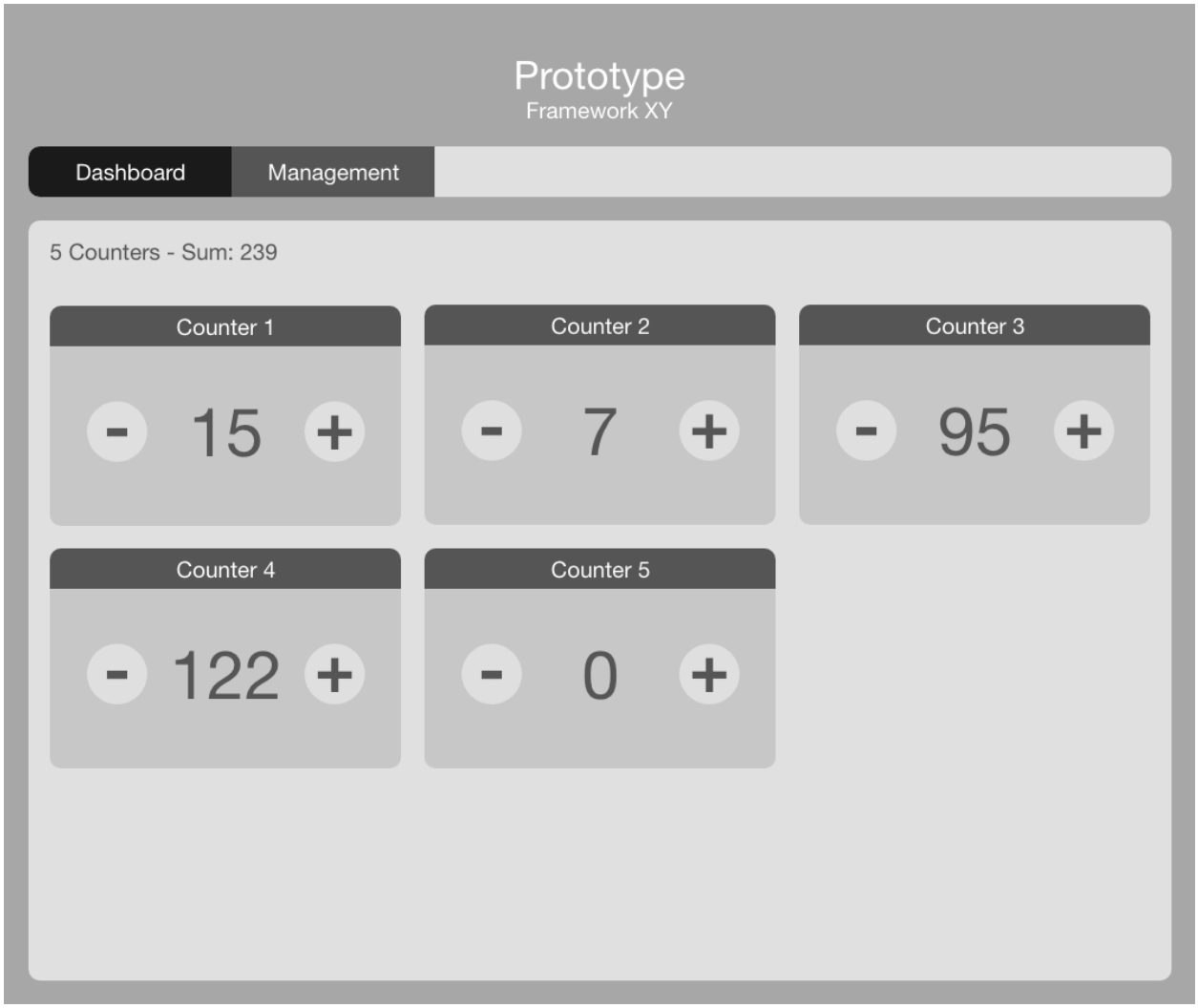


Abbildung 24: Mockup der Dashboard Seite für den Prototyp (Eigene Darstellung)

Neben dem Mockup für die Dashboard Seite wird auch ein Mockup für die Management Seite erstellt. In der nachfolgenden Abbildung wird die Management Seite anhand des Mockups dargestellt.

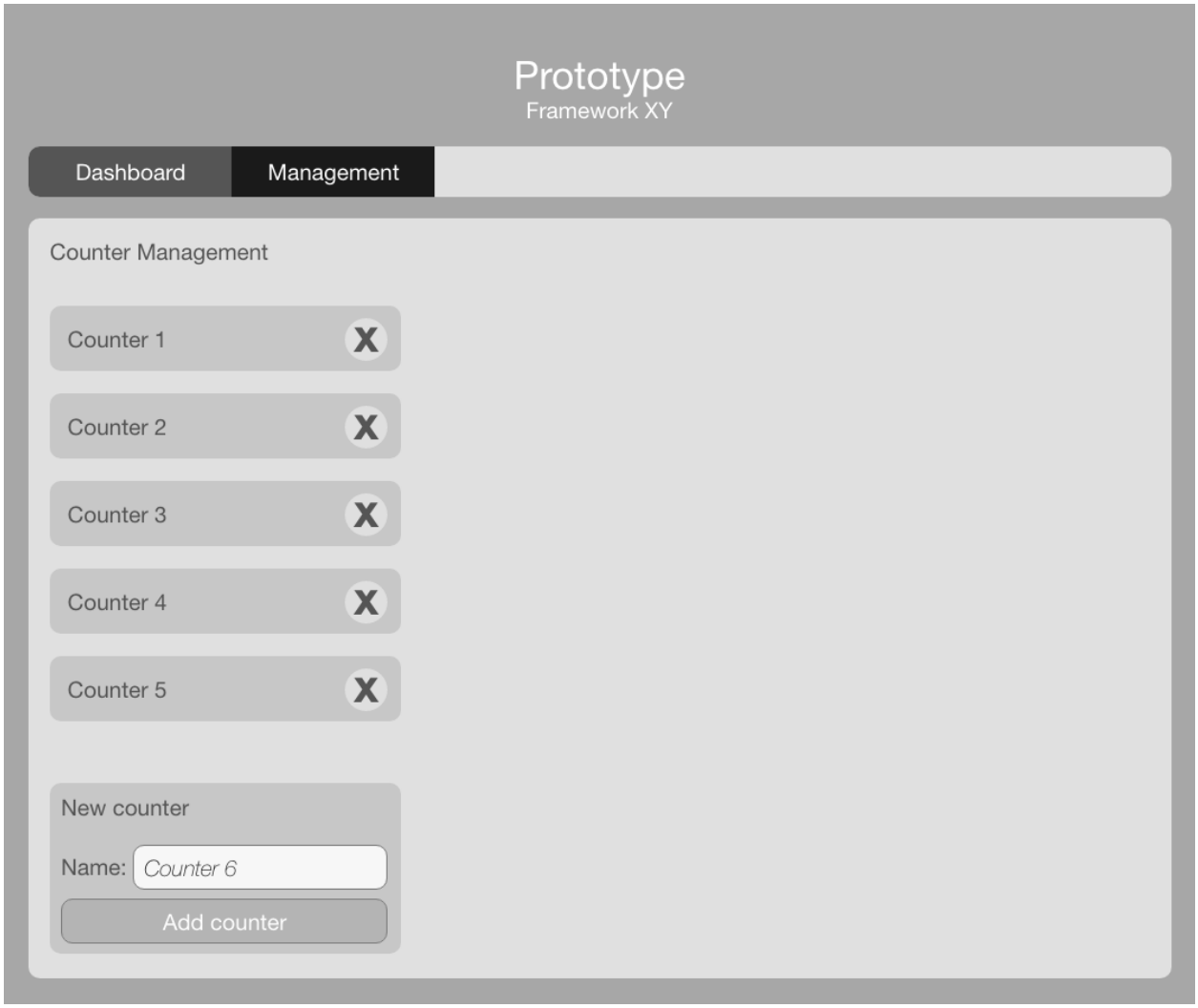


Abbildung 25: Mockup der Management Seite für den Prototyp (Eigene Darstellung)

6.4 Ergebnisse der Prototypen

In der Umsetzungsphase wurde für jedes der ausgewählten Frameworks ein Prototyp anhand des Entwurfs erstellt und sind auf dem Datenträger dieser Arbeit abgelegt. Der Aufbau der Komponenten der Webapplikation ist bei allen Frameworks annähernd gleich, jedoch unterscheidet sich die Syntax sowie die Programmiersprache. Stellvertretend für alle Frameworks wird die Umsetzung des Prototyps für das Frameworks Angular erklärt. Dabei wird in den einzelnen Schritten auch auf Gemeinsamkeiten und Unterschiede zwischen den Frameworks eingegangen.

Im ersten Schritt wird die Entwicklungsumgebung für die Verwendung des Angular Frameworks eingerichtet. Dazu wird die IDE Visual Studio Code von Microsoft installiert. Weiters wird als Backend-Technologie für das Betreiben eines lokalen Webservers die serverseitige JavaScript Laufzeitumgebung node.js installiert. Durch die Installation von node.js wird zugleich für die Paketverwaltung der Paketmanager node package manager (npm) mitinstalliert. Dieser ermöglicht es, das Framework Angular und dessen Abhängigkeiten über die Kommandozeile des Betriebssystems herunterzuladen und zu installieren. Für alle Frameworks wird die gleiche IDE verwendet. Bei allen JavaScript Frameworks wird ebenfalls die Laufzeitumgebung node.js eingesetzt. Beim Framework Blazor wird anstatt node.js die .NET Core Plattform von Microsoft verwendet.

Nach der Einrichtung der Entwicklungsumgebung wird das Projekt für die Webapplikation angelegt. Dies erfolgt durch die Verwendung des CLI, mit dem das Projekt angelegt wird. Bei den anderen Frameworks wurde für die Erstellung der Webapplikation ebenfalls das im Framework integrierte oder ein externes CLI verwendet. In der nachfolgenden Abbildung wird das Anlegen des Projekts mit dem Angular CLI dargestellt.

```
C:\Users\manuel.reinhart\iCloudDrive\FH\Masterarbeit\08_Source\01_Demo_Angular>ng new
? What name would you like to use for the project? Prototype-Angular
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE Prototype-Angular/angular.json (3867 bytes)
CREATE Prototype-Angular/package.json (1324 bytes)
CREATE Prototype-Angular/README.md (1033 bytes)
CREATE Prototype-Angular/tsconfig.json (408 bytes)
CREATE Prototype-Angular/tslint.json (2837 bytes)
CREATE Prototype-Angular/.editorconfig (246 bytes)
```

Abbildung 26: Anlegen eines Projekts mit dem Angular CLI (Eigene Darstellung)

Ebenfalls wurden die Seiten aus dem in Kapitel 6.3 erstellten Entwurf in einzelne Komponenten unterteilt. Diese sind wie folgt:

- App: Hauptkomponente mit Titel und Untertitel
- Navbar: Komponente für die Navigationsleiste
- Counter-Overview: Komponente für die Dashboard Seite
- Counter-Management: Komponente für die Management Seite
- Counter: Komponente für den Zähler

Für das Anlegen der Komponenten wurde wie bereits vorhin für die Erstellung der Applikation auch das Angular CLI verwendet. Ebenso wurden bei den anderen Frameworks CLIs zur Erstellung der Komponenten herangezogen. In der nachfolgenden Abbildung wird das Hinzufügen der Komponenten mithilfe des CLI dargestellt.

```
C:\Users\manuel.reinhart\iCloudDrive\FH\Masterarbeit\08_Source\01_Demo_Angular\Prototype-Angular>ng generate component
? What name would you like to use for the component? navbar
CREATE src/app/navbar/navbar.component.html (25 bytes)
CREATE src/app/navbar/navbar.component.spec.ts (628 bytes)
CREATE src/app/navbar/navbar.component.ts (269 bytes)
CREATE src/app/navbar/navbar.component.css (0 bytes)
UPDATE src/app/app.module.ts (475 bytes)

C:\Users\manuel.reinhart\iCloudDrive\FH\Masterarbeit\08_Source\01_Demo_Angular\Prototype-Angular>ng g c
? What name would you like to use for the component? counter
CREATE src/app/counter/counter.component.html (26 bytes)
CREATE src/app/counter/counter.component.spec.ts (635 bytes)
CREATE src/app/counter/counter.component.ts (273 bytes)
CREATE src/app/counter/counter.component.css (0 bytes)
UPDATE src/app/app.module.ts (561 bytes)

C:\Users\manuel.reinhart\iCloudDrive\FH\Masterarbeit\08_Source\01_Demo_Angular\Prototype-Angular>ng g c counter-overview
CREATE src/app/counter-overview/counter-overview.component.html (35 bytes)
CREATE src/app/counter-overview/counter-overview.component.spec.ts (692 bytes)
CREATE src/app/counter-overview/counter-overview.component.ts (308 bytes)
CREATE src/app/counter-overview/counter-overview.component.css (0 bytes)
UPDATE src/app/app.module.ts (681 bytes)

C:\Users\manuel.reinhart\iCloudDrive\FH\Masterarbeit\08_Source\01_Demo_Angular\Prototype-Angular>ng g c counter-management
CREATE src/app/counter-management/counter-management.component.html (37 bytes)
CREATE src/app/counter-management/counter-management.component.spec.ts (706 bytes)
CREATE src/app/counter-management/counter-management.component.ts (316 bytes)
CREATE src/app/counter-management/counter-management.component.css (0 bytes)
UPDATE src/app/app.module.ts (809 bytes)
```

Abbildung 27: Erstellen der Prototyp-Komponenten mit dem Angular CLI (Eigene Darstellung)

Die erstellten Komponenten werden darauffolgend im geöffneten Projekt der IDE angezeigt und bestehen jeweils aus einer HTML-Datei für das Markup, einer CSS-Datei für das Styling, einer TypeScript-Datei für Skripte sowie einer zweiten TypeScript-Datei für die Durchführung von Tests. In der nachfolgenden Abbildung wird die Dokumentenstruktur in der IDE angezeigt.

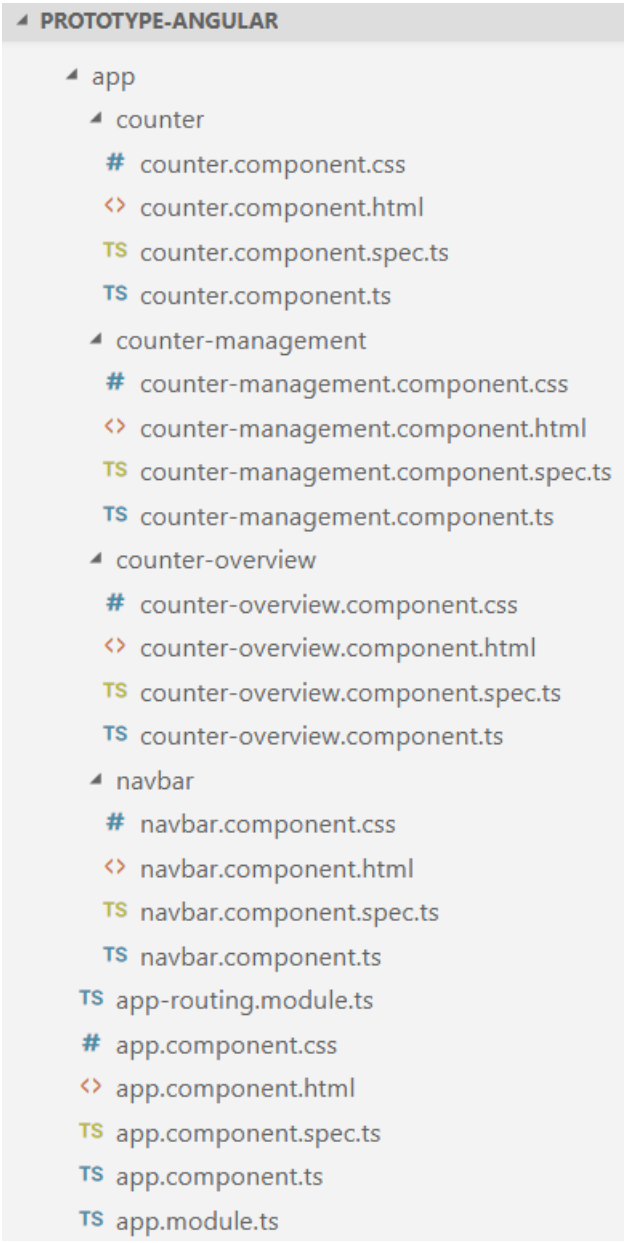


Abbildung 28: Projektstruktur der Angular Komponenten in der IDE (Eigene Darstellung)

Für die Navigation zwischen den einzelnen Seiten wurde das bei der Projekterstellung miterstellte Routing-Modul verwendet. In diesem Modul werden die Routen festgelegt, sodass je nach Pfad eine bestimmte Komponente angezeigt wird. Die Navigationsleiste wurde dabei fix in die Hauptkomponente integriert, da diese unabhängig vom Pfad immer sichtbar sein soll. Im Listing 2 sind die definierten Routen für die zwei Seiten des Prototyps definiert. Dies beinhaltet auch die Definition der Komponente für die Startseite, also bei der Navigation mit leerem Pfad.

```
const routes: Routes = [
  { path: "counters", component: CounterOverviewComponent },
  { path: "management", component: CounterManagementComponent },
  { path: '', redirectTo: '/counters', pathMatch: 'prefix' }
];
```

Listing 2: Definition der Routen für die Navigation (Eigene Darstellung)

Im Listing 3 wird der HTML-Code der Hauptkomponente dargestellt. Dabei wird ersichtlich, dass der Titel, der Untertitel, sowie die Komponente der Navigationsleiste unabhängig vom Routing fix dargestellt wird. Die Direktive router-outlet ist ein Platzhalter, der vom Routing-Modul je nach Route dynamisch ausgetauscht wird.

```
<div>
  <h1>Prototype</h1>
  <h3>Framework Angular</h3>
</div>
<app-navbar></app-navbar>
<router-outlet></router-outlet>
```

Listing 3: Einbindung der Routing-Komponente in der Hauptkomponente (Eigene Darstellung)

Für die Verwaltung der Zähler wurde ein globales Service angelegt, das die Instanzen der Zähler verwaltet. Dieses Service wird über die Dependency Injection durch das Angular Framework in jeder Komponente zur Verfügung gestellt. Im Listing 4 wird die Verwendung eines globalen Service bei der Auflistung der Zähler angeführt. Weiters sind im Listing einige Datenbindungen ersichtlich. Die Anzahl der Zähler, sowie eine automatisch berechnete Summe der Zählerstände wird mit dem Überschriftelement verbunden und zur Laufzeit automatisch vom Framework aktualisiert. Bei der Auflistung der Zähler wird durch die Verwendung von *ngFor für jeden Zähler die Zählerkomponente im DOM angefügt. Zusätzlich wird jeder Zählerkomponente mittels HTML-Attribut die Instanz des einzelnen Zählers übergeben.

```
<div class="area">
  <h3>{{counterService.Counters.length}} Counters - Summe: {{ Sum }}</h3>

  <div class="counters">
    <app-counter *ngFor="let counter of counterService.Counters" [counter]="counter"></app-counter>
  </div>

</div>
```

Listing 4: Verwaltung der Zähler durch ein globales Service (Eigene Darstellung)

Um die Webapplikation zu starten, kann mittels CLI das Projekt kompiliert und gestartet werden. Dadurch wird automatisch ein Webserver gestartet und die Webapplikation im Browser geöffnet. Wird im Quellcode eine Änderung durchgeführt, so wird diese automatsch kompiliert und in der laufenden Webapplikation aktualisiert. In der Abbildung 29 sowie in der Abbildung 30 ist die Webapplikation des Prototyps für das Framework Angular im Browser ersichtlich.

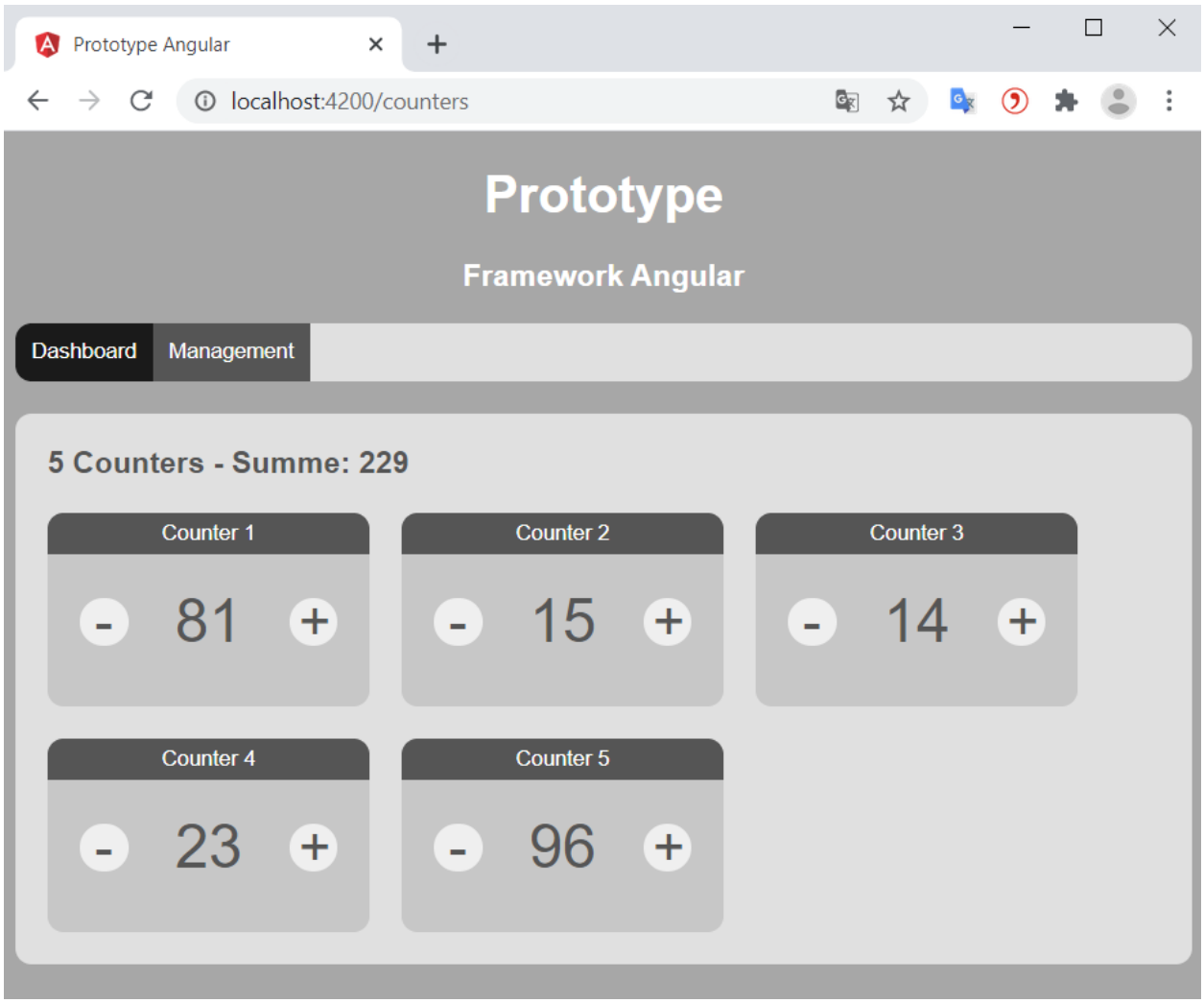


Abbildung 29: Dashboard Seite des erstellten Prototyps (Eigene Darstellung)

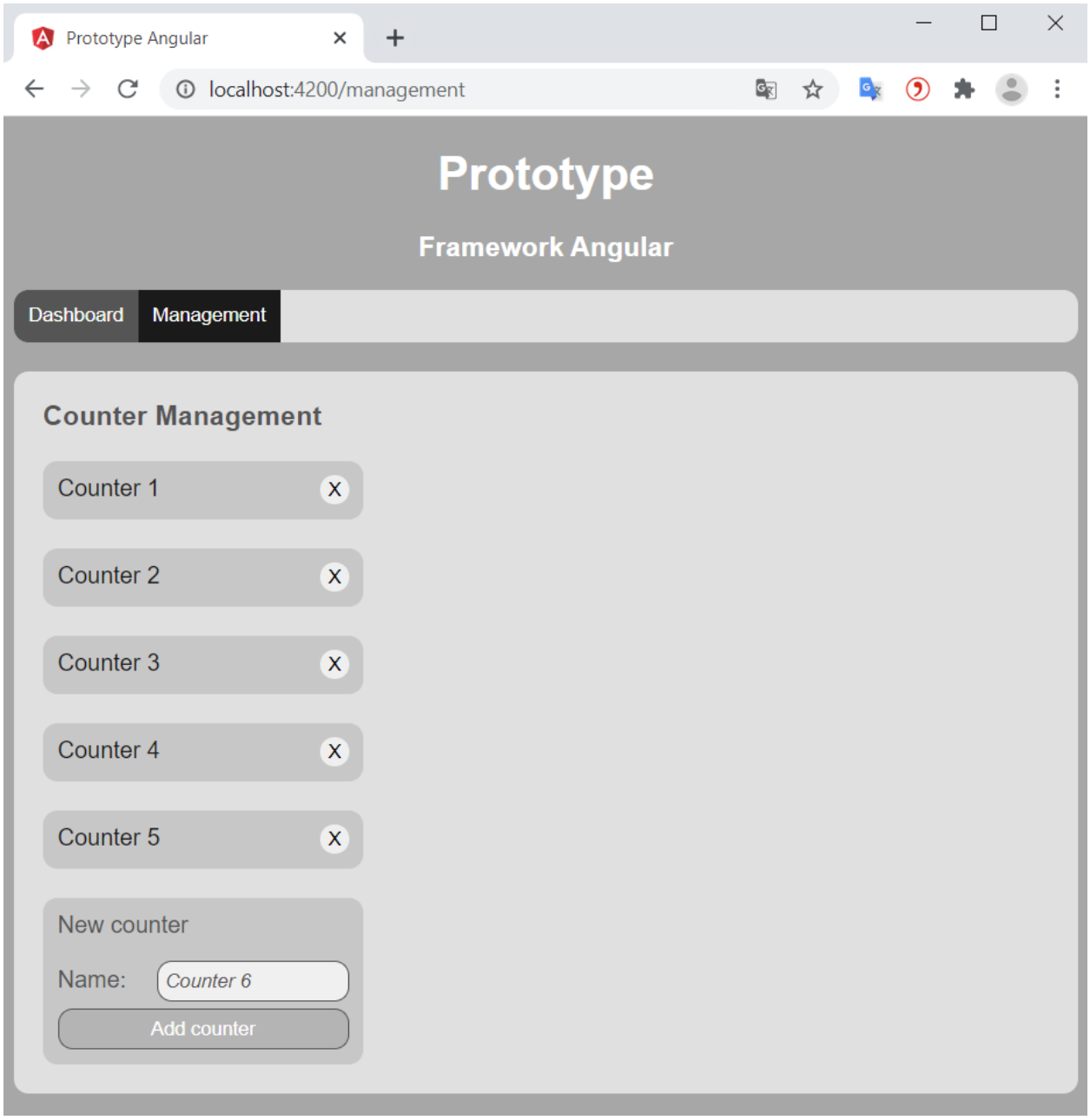


Abbildung 30: Management Seite des erstellten Prototyps (Eigene Darstellung)

Abschließend wurden automatisierte Tests für den Quellcode der Webapplikation erstellt, um die Testbarkeit des Frameworks festzustellen.

Die Prototypen für die Frameworks React, Vue.js und Blazor wurden nach der gleichen Herangehensweise entwickelt. Dabei waren die einzelnen Aufgaben mit unterschiedlicher Komplexität und unterschiedlichem Zeitaufwand behaftet. Die einzelnen Unterschiede werden für die Bewertung der Frameworks herangezogen.

7 BEWERTUNG

In diesem Kapitel werden die in Kapitel 4 ausgewählten Frameworks anhand des im Kapitel 5 zusammengestellten Kriterienkataloges bewertet.

7.1 Angular

Nachfolgend wird das Framework Angular nach den Kriterien des Kriterienkatalogs bewertet.

7.1.1 Programmiersprache

In der offiziellen Dokumentation von Angular ist die Programmiersprache TypeScript angeführt. Da TypeScript zu JavaScript transpiliert wird und im Browser letztendlich JavaScript ausgeführt wird, ist es prinzipiell möglich mit JavaScript zu entwickeln. Allerdings werden im Framework zahlreiche TypeScript-spezifische Funktionalitäten wie Decorators eingesetzt, die mit JavaScript äußerst mühsam bei der Entwicklung wären. Deshalb ist die einzig sinnvoll einsetzbare Programmiersprache für das Framework TypeScript (Google 2020a).

7.1.2 Beliebtheit

Das Open-Source-Projekt wird über die Versionsverwaltung von GitHub verwaltet. Auf dieser Plattform haben registrierte Entwickler*innen die Möglichkeit, das Framework mit einem Stern zu versehen, um ein positives Feedback zu vergeben. Das Framework erhielt dabei auf dieser Plattform rund 68.200 Sterne von den angemeldeten Nutzer*innen (GitHub 2020a).

7.1.3 Community

Ebenso sind auf der Plattform GitHub jene Entwickler*innen ersichtlich, die außerhalb des Kern-Entwicklungsteams bereits aktiv am Framework mitgestaltet haben. Beim Angular Framework sind dies bereits 1.279 Entwickler*innen (GitHub 2020a).

7.1.4 Lernkurve

Das Framework ist durch seine Vielzahl and Funktionen äußerst umfangreich, wodurch die Komplexität des Frameworks erhöht wird. Bei der Verwendung von Angular sind zahlreiche spezifische Eigenheiten des Frameworks notwendig, dazu gehört beispielsweise der Aufbau von Modulen mit der richtigen Deklarierung von Komponenten oder die Syntax für verschiedene

Arten von Datenbindungen. Um die Konzepte von Angular, beispielsweise die Dependency Injection, zu verstehen, ist ein gewisses Maß an Knowhow zur allgemeinen Softwareentwicklung notwendig. Aufgrund dieser Erkenntnisse wird Angular mit 3 von 5 Sternen für die Einfachheit des Erlernens des Frameworks bewertet.

7.1.5 Lernunterstützung

Als Unterstützung für das Erlernen des Frameworks wird auf dessen Webseite ein Tutorial namens „Tour of Hero“ angeboten. Dieses Tutorial gibt eine umfassende Einführung über alle Funktionen. Zusätzlich werden viele Bücher und private Kurse zu Angular entgeltlich angeboten. Durch die Community werden außerdem kostenlose Treffen veranstaltet, bei denen Funktionen des Frameworks vorgestellt werden und ein Erfahrungsaustausch unter Expert*innen stattfinden kann. Das Framework wird daher mit 5 von 5 Sternen hinsichtlich des Kriteriums der Lernunterstützung bewertet.

7.1.6 Performanz

Um die Performanz zu bewerten, wird auf einen Benchmark Vergleich der Frameworks zurückgegriffen. In diesem Benchmark wurde die Performanz der Frameworks hinsichtlich Manipulierung des DOMs, initiale Ladezeiten sowie Speicherverbrauch anhand verschiedenster Metriken gemessen. Daraus wurde für jede der drei Kategorien der geometrische Mittelwert gebildet. Für die Bewertung der Frameworks wird aus den geometrischen Mittelwerten der arithmetische Mittelwert gebildet und verwendet (Krause 2020).

Der arithmetische Mittelwert der Performanz für Angular wird aus den Werten 1,35, 2,32 und 1,19 aus dem Benchmark gebildet und beträgt somit 1,62.

Da im Benchmark ein höherer Wert eine schlechtere Performanz bedeutet, wird der Mittelwert invertiert und beträgt somit 0,62.

7.1.7 Testbarkeit

Im Angular Framework wird das Testen schon zu Beginn der Entwicklung einer Webapplikation im Entwicklungsprozess verankert. So wird beim Erstellen von Komponenten über das CLI automatisch eine Datei mit einer Vorlage für das Testen der angelegten Komponente generiert. Für das Erstellen und das Ausführen der Tests sind alle Komponenten im Framework integriert, sodass keine externen Test-Frameworks zusätzlich installiert werden müssen. Anhand des Prototyps war das Testen einfach umsetzbar, weshalb das Framework dafür mit 5 von 5 Sternen bewertet wird.

7.1.8 Dokumentation

Die Dokumentation des Frameworks wird auf dessen offizieller Webseite veröffentlicht. Diese ist umfangreich und beschreibt alle Funktionen. Lediglich bei den Konzepten wären grafische

Darstellung hilfreicher und teilweise wären konkrete Beispiele von Nutzen. Die Bewertung der Dokumentation beläuft sich dadurch auf 4 von 5 Sternen.

7.1.9 Installationsaufwand

Für die Installation des Frameworks ist die Installation von node.js erforderlich. Anschließend können durch node.js über den darin enthaltenen Paketmanager die benötigten Pakete für Angular heruntergeladen werden. Dadurch kann auch das Angular-CLI installiert werden. Hiermit können nach der Installation unkompliziert neue Projekte erstellt und verwaltet werden. Das Framework wird daher mit 4 von 5 Sternen bewertet.

7.1.10 Entwicklungsgeschwindigkeit

Durch den großen im Framework enthaltenen Funktionsumfang werden zahlreiche häufig erforderliche Funktionalitäten einer Webapplikation wesentlich vereinfacht. Mithilfe des CLI können neue Komponenten und Projekterweiterungen unkompliziert hinzugefügt werden. Dies erhöht die Entwicklungsgeschwindigkeit gegenüber einer Umsetzung der Funktionen ohne Framework enorm. Voraussetzung für die Reduzierung der Entwicklungszeit ist das Verstehen des Frameworks. Die Entwicklungsgeschwindigkeit wird daher anhand der Umsetzung des Prototyps mit 5 von 5 Sternen bewertet.

7.1.11 Funktionsumfang

Das Framework bietet 14 von 16 Funktionen an, die in der nachfolgenden Tabelle aufgelistet werden. Die unterstützten Funktionen wurden dabei von der offiziellen Webseite des Frameworks entnommen (Google 2020a).

Funktion	**Inkludiert**
Routing ohne Serveranfragen	Ja
Komponenten / Templates	Ja
Datenbindungen unidirektional	Ja
Datenbindungen bidirektional	Ja
Ereignisbindungen	Ja
Pipes	Ja
Direktiven	Ja
Dependency Injection	Ja
Internationalisierung	Ja
Formulare	Ja
Validierung	Ja
Animationen	Ja
HTTP-Client	Ja
Logging	Nein
Lifecycle	Ja
Authentifizierung	Nein

Tabelle 2: Funktionsumfang des Framework Angular (Eigene Darstellung)

7.1.12 Erweiterbarkeit

Für Angular werden von der Community viele Open Source Libraries angeboten. Als Beispiel für Erweiterungen könnten Libraries für das Einbinden von Karten oder Diagrammen sowie Libraries für Funktionen wie Datei-Uploads oder diverse Authentifizierungen verwendet werden. Das Framework wird bezüglich der Erweiterbarkeit mit 5 von 5 Sternen bewertet.

7.1.13 Browserkompatibilität

Die aktuelle Version von Angular kann ab dem Internet Explorer 9 mit gewissen Einschränkungen, wie beispielsweise schlechtere Performanz und geringeren Funktionsumfang, und der Verwendung zusätzlicher Ressourcen betrieben werden. Da Angular immer für die aktuellen Webtechnologien entwickelt wird, müssen bei älteren Browsern sogenannte Polyfills verwendet werden. Polyfills sind Skripte, die die Technologien neuerer Browser emulieren. Unter diesen Umständen ist Angular mit Browsern ab dem Jahr 2011 kompatibel (Google 2020a).

7.1.14 Support

Unterstützung bei Problemen mit dem Framework wird von Angular durch die Community angeboten. Entwickler*innen haben die Möglichkeit, eventuelle Fehler im Framework oder Probleme bei der Umsetzung spezifischer Anforderungen, auf der Plattform GitHub kostenlos einzutragen. Eingetragene Probleme werden von der Community oder dem Kern-Entwicklungsteams selbst priorisiert und verarbeitet. Entwickler*innen haben jedoch diesbezüglich kein Anspruch auf eine Lösung des Problems. Zusätzlich werden durch die Community des Frameworks auch auf diversen Foren im Softwareentwicklungsbereich Fragen zum Framework beantwortet. Ist dies nicht ausreichend, stehen den Entwickler*innen noch kostenpflichte Consulting-Unternehmen für eine Unterstützung zur Verfügung.

7.1.15 Personalbedarf

Aktuell werden in Österreich auf der Jobplattform devjobs.at 242 Jobs für das Framework Angular angeboten. Insgesamt sind 611 Entwicklerteams auf der Plattform registriert, die dieses Framework verwenden. Das Jahresdurchschnittsgehalt der angebotenen Jobs für das Framework beträgt 46.349 Euro und liegt in der Spanne zwischen 29.000 und 70.000 Euro.

7.2 React

Nachfolgend wird das Framework React hinsichtlich der Kriterien des Kriterienkatalogs bewertet.

7.2.1 Programmiersprache

React ist ursprünglich für die Programmiersprache JavaScript konzipiert. Das bedeutet, dass mit JavaScript das Framework vollumfänglich in der zu entwickelnden Webapplikation eingesetzt werden kann. Zusätzlich zu dieser Programmiersprache kann auch mit TypeScript gearbeitet werden, was einige Vorteile mit sich bringt - etwa die Typisierung von Variablen und dadurch eine frühzeitige Fehlererkennung. TypeScript hat gegenüber JavaScript den Nachteil, dass bei der Entwicklung der geschriebene TypeScript-Quellcode erst zu JavaScript transpiliert werden muss. Dennoch sind beide Varianten, je nach Präferenz, möglich, weshalb sowohl TypeScript als auch JavaScript mögliche Programmiersprachen für React TypeScript darstellen (Gackenheimer 2015).

7.2.2 Beliebtheit

Wie bei Angular wird auch das Open-Source-Projekt React über die Versionsverwaltung von GitHub verwaltet. Das Framework erhielt dabei auf dieser Plattform rund 160.000 Sterne von den angemeldeten Nutzer*innen (GitHub 2020c).

7.2.3 Community

Auf der Plattform GitHub ist ersichtlich, dass bereits 1.522 Entwickler*innen außerhalb des Kern-Entwicklungsteams am Framework aktiv mitgestaltet haben (GitHub 2020c).

7.2.4 Lernkurve

Für die Anwendung des Frameworks sind fast ausschließlich JavaScript-Kenntnisse und nur wenig framework-spezifisches Wissen notwendig. Da React hinsichtlich des Konzepts vollkommen anders als typische SPA-Frameworks aufgebaut ist, fällt der Einstieg wesentlich schwerer, wenn Frameworks mit MVC-Ansätze gewohnt sind. Das Framework wird daher mit 4 von 5 Sternen bewertet.

7.2.5 Lernunterstützung

Auf der offiziellen Webseite des Frameworks wird ein Tutorial angeboten, das die Prinzipien des Frameworks gut veranschaulicht. Im Tutorial wird React anhand der Entwicklung des Zweipersonen-Strategiespiels „Tic-Tac-Toe“ erklärt. Dies ist in der Praxis beispielsweise für die Entwicklung einer Geschäftsanwendung schwer umzuwandeln, da bei der Entwicklung des

gezeigten Spiels wesentliche Komponenten einer vollständigen Webapplikation, wie Routing oder Validierung, fehlen. Das Framework wird daher mit 3 von 5 Sternen bewertet.

7.2.6 Performanz

Wie bei der Bewertung des Frameworks Angular, wird auch bei der Bewertung der Performanz von React auf denselben Benchmark zurückgegriffen. Der arithmetische Mittelwert der Werte im Benchmark wird aus den Werten 1,46, 1,14 und 1,13 gebildet und beträgt 1,24. Der für die Bewertung invertierte Wert beträgt 0,80 (Krause 2020).

7.2.7 Testbarkeit

In der offiziellen Dokumentation von React wird das Thema Testen sehr gut beschrieben. Die notwendigen Tools für das Erstellen und Ausführen von Tests sind jedoch nicht im Framework integriert und müssen manuell nachinstalliert werden. Zudem müssen im Gegensatz zu Angular die Dateien selbst erstellt werden. Dennoch konnten anhand des Prototyps die Tests sehr gut umgesetzt werden, wodurch das Framework mit 4 von 5 Sternen bewertet wird.

7.2.8 Dokumentation

Die Dokumentation wird auf der offiziellen Webseite des Frameworks veröffentlicht. Ähnlich wie bei Angular sind die Funktionen gut beschrieben und mit Beispielen versehen. Für grundlegende Konzepte wären allerdings grafische Darstellungen hilfreich. Die Dokumentation des Frameworks wird mit 4 von 5 Sternen bewertet.

7.2.9 Installationsaufwand

Für die Verwendung des Frameworks in einer bereits vorhandenen Webapplikation sind prinzipiell keine spezifischen Tools notwendig und der Download und die Einbettung der JavaScript Datei ist ausreichend. Um jedoch eine vollständige SPA von Grund auf mit React zu entwickeln bietet sich node.js an. Hiermit können Projekte mithilfe eines CLI erstellt werden. Das Framework wird daher mit 4 von 5 Sternen bewertet.

7.2.10 Entwicklungsgeschwindigkeit

Da React im Vergleich zu anderen SPA-Frameworks einen eingeschränkten Funktionsumfang aufweist, müssen viele Funktionalitäten einer Webapplikation, wie bidirektionale Datenbindungen mithilfe von zusätzlichen Frameworks oder Libraries wie react-chopper umgesetzt werden. Alternativ können die fehlenden Funktionalitäten, wie bidirektionale Datenbindungen auch ohne Unterstützung eines Frameworks oder einer Library mit allgemeinen Funktionalitäten von JavaScript umgesetzt werden, was aber einen höheren

Entwicklungsaufwand bedeutet. Dadurch wird die Entwicklungsgeschwindigkeit anhand des Prototyps mit 3 von 5 Sternen bewertet.

7.2.11 Funktionsumfang

Das Framework bietet 10 von 16 Funktionen an, die in der nachfolgenden Tabelle aufgelistet werden. Die unterstützten Funktionen wurden dabei von der offiziellen Webseite des Frameworks entnommen (Facebook Inc. 2020).

Funktion	**Inkludiert**
Routing ohne Serveranfragen	Ja
Komponenten / Templates	Ja
Datenbindungen unidirektional	Ja
Datenbindungen bidirektional	Nein
Ereignisbindungen	Ja
Pipes	Nein
Direktiven	Nein
Dependency Injection	Nein
Internationalisierung	Ja
Formulare	Ja
Validierung	Ja
Animationen	Ja
HTTP-Client	Nein
Logging	Nein
Lifecycle	Ja
Authentifizierung	Ja

Tabelle 3: Funktionsumfang des Framework React (Eigene Darstellung)

7.2.12 Erweiterbarkeit

Um den Funktionsumfang von React zu erweitern gibt es viele Libraries, die mit dem Framework kombiniert werden können. Auf der offiziellen Webseite werden jedoch nur wenige Erweiterungsmöglichkeiten angeboten. Das Framework wird daher hinsichtlich des Kriteriums der Erweiterbarkeit mit 4 von 5 Sternen bewertet.

7.2.13 Browserkompatibilität

Ähnlich wie bei Angular wird das Framework für aktuelle modere Browser entwickelt. Durch die Anwendung von Polyfills funktioniert React jedoch ab Browsern aus dem Jahr 2011. Somit kann React ab dem Internet Explorer 9 verwendet werden (Facebook Inc. 2020).

7.2.14 Support

Wie beim Framework Angular wird auch bei React die kostenfreie Unterstützung durch die Community angeboten. Durch den Framework Hersteller selbst wird kein direkter Support angeboten. Diverse Consulting-Unternehmen bieten bei Bedarf kostenpflichtige Unterstützung für das Framework an.

7.2.15 Personalbedarf

Aktuell werden in Österreich auf der Jobplattform devjobs.at 155 Jobs für das Framework React angeboten. Insgesamt sind 451 Entwicklerteams auf der Plattform registriert, die dieses Framework verwenden. Das Jahresdurchschnittsgehalt der angebotenen Jobs für das Framework beträgt 49.232 Euro und liegt in der Spanne zwischen 28.000 und 84.000 Euro.

7.3 Vue.js

Nachfolgend wird das Framework Vue.js nach den Kriterien des Kriterienkatalogs bewertet.

7.3.1 Programmiersprache

Vue.js ist, wie der Name bereits verrät, ein JavaScript-Framework. Das bedeutet, dass mit der Programmiersprache JavaScript das Framework vollumfänglich in der zu entwickelnden Webapplikation eingesetzt werden kann. Wie bei React kann zusätzlich kann auch mit TypeScript gearbeitet werden, was dieselben Vor- und Nachteile hinsichtlich einer Typisierung und frühzeitiger Fehlererkennung, sowie die Notwendigkeit des Transpilieren mit sich bringt. Die Auswahl der Programmiersprache obliegt somit den Entwickler*innen, wodurch für die Bewertung JavaScript und TypeScript aufgenommen werden. (Steyer 2019).

7.3.2 Beliebtheit

Wie bei Angular und React wird auch das Open-Source-Projekt Vue.js über die Versionsverwaltung von GitHub verwaltet. Das Framework erhielt auf dieser Plattform rund 176.000 Sterne von den angemeldeten Nutzer*innen (GitHub 2020d).

7.3.3 Community

Auf der Plattform GitHub ist ersichtlich, dass bereits 382 Entwickler*innen außerhalb des Kern-Entwicklungsteams am Framework aktiv mitgestaltet haben (GitHub 2020d).

7.3.4 Lernkurve

Das Framework ist trotz seines hohen Funktionsumfang intuitiv gestaltet. Vue.js kann mithilfe der Einführung auf der offiziellen Webseite leicht erlernt werden. Anhand des Prototyps war das Framework ohne Vorkenntnisse anwendbar. Deshalb wird das Framework mit 5 von 5 Sternen bewertet.

7.3.5 Lernunterstützung

Auf der offiziellen Webseite des Frameworks wird versucht, dieses anhand von Videos zu erklären. Dies wird gut umgesetzt, weshalb die Konzepte und Prinzipien einfach zu verstehen sind. Zusätzlich wäre aber ein schriftlich dokumentiertes Tutorial, ähnlich der „Tour of Hero“ von Angular wünschenswert. Die Lernunterstützung des Frameworks wird daher mit 4 von 5 Sternen bewertet.

7.3.6 Performanz

Wie bereits bei der Bewertung der Frameworks Angular und React, wird auch bei Vue.js der gleiche Benchmark für die Bewertung verwendet. Der arithmetische Mittelwert der Benchmark Werte wird dabei aus den Werten 1,23, 1 und 1 gebildet und beträgt 1,07. Für die Bewertung der Performanz wird dieser Wert wiederum invertiert und beträgt 0,93.

7.3.7 Testbarkeit

Mit Vue.js können Unit-Tests, Komponenten-Tests sowie End-to-End Tests durchgeführt werden. Ähnlich wie beim Framework Angular kann das Testen bereits bei der Erstellung des Projekts aktiviert werden, sodass Test-Dateien automatisch generiert werden. In der Dokumentation der offiziellen Webseite ist das Testen von Webapplikationen gut beschrieben und mit Beispielen ergänzt. Die Testbarkeit des Frameworks wird somit mit 5 von 5 Sternen bewertet.

7.3.8 Dokumentation

Die Dokumentation ist umfangreich, aber dennoch übersichtlich und gut strukturiert. Die Konzepte sind im Vergleich zu anderen Frameworks sehr gut grafisch dargestellt. Aus diesem Grund wird die Dokumentation des Frameworks mit 5 von 5 Sternen bewertet.

7.3.9 Installationsaufwand

Mit der offiziellen CLI können Webapplikationen mit Vue.js schnell und unkompliziert erstellt werden. Zusätzlich bietet das Framework eine grafische Benutzeroberfläche anhand einer Webseite an, mit der Projekte erstellt und verwaltet werden können. Der Installationsaufwand des Frameworks wird daher mit 5 von 5 Sternen bewertet.

7.3.10 Entwicklungsgeschwindigkeit

Durch das gute CLI können Komponenten sehr schnell erstellt werden und durch die vorhandenen Features einiges an Entwicklungszeit ersparen. Zusätzlich werden Fehler, die beispielsweise im Template vorhanden sind, visuell sehr gut im Browser angezeigt, sodass diese nicht in der Konsole des Browsers gesucht werden muss. Anhand des Prototyps sind keine Punkte aufgekommen, die hinsichtlich der Entwicklungsgeschwindigkeit verbessert werden sollten. Daher wird das Framework hinsichtlich der Entwicklungsgeschwindigkeit mit 5 von 5 Sternen bewertet.

7.3.11 Funktionsumfang

Das Framework bietet 12 von 16 Funktionen an, die in der nachfolgenden Tabelle aufgelistet werden. Die unterstützten Funktionen wurden dabei von der offiziellen Webseite des Frameworks entnommen (You 2020).

Funktion	**Inkludiert**
Routing ohne Serveranfragen	Ja
Komponenten / Templates	Ja
Datenbindungen unidirektional	Ja
Datenbindungen bidirektional	Ja
Ereignisbindungen	Ja
Pipes	Ja
Direktiven	Ja
Dependency Injection	Ja
Internationalisierung	Ja
Formulare	Nein
Validierung	Ja
Animationen	Ja
HTTP-Client	Nein
Logging	Nein
Lifecycle	Ja
Authentifizierung	Nein

Tabelle 4: Funktionsumfang des Framework Vue.js (Eigene Darstellung)

7.3.12 Erweiterbarkeit

Es existiert eine Vielzahl an Plug-Ins, die über das CLI oder über die integrierte Management-Oberfläche verwaltet werden. Die UI bietet zudem das Browsen von Plug-Ins an. Die Erweiterbarkeit wird daher mit 5 von 5 Sternen bewertet.

7.3.13 Browserkompatibilität

Auch bei Vue.js wird bei der Entwicklung der Fokus auf aktuelle Webtechnologien gelegt. Die Verwendung des Frameworks ist mit bestimmten Einschränkungen und der Verwendung von Polyfills ab Browsern aus dem Jahr 2011 und somit ab dem Internet Explorer 9 möglich (Steyer 2019).

7.3.14 Support

Das Angebot für den Support des Frameworks ist ähnlich dem der Frameworks Angular und React. Dieser ist durch die Community über die Plattform GitHub oder diversen Foren kostenfrei zugänglich. Vom Framework Hersteller selbst wird keine direkte Unterstützung außerhalb der Community angeboten. Diverse Consulting-Unternehmen bieten jedoch kostenpflichtige Unterstützung für das Framework an.

7.3.15 Personalbedarf

Aktuell werden in Österreich auf der Jobplattform devjobs.at 85 Jobs für das Framework Vue.js angeboten. Insgesamt sind 253 Entwicklerteams auf der Plattform registriert, die dieses Framework verwenden. Das Jahresdurchschnittsgehalt der angebotenen Jobs für das Framework beträgt 47.005 Euro und liegt in der Spanne zwischen 27.000 und 70.000 Euro.

7.4 Blazor

Nachfolgend wird das Framework Blazor anhand Kriterien des Kriterienkatalogs bewertet.

7.4.1 Programmiersprache

Blazor ist mit der Programmiersprache C# zu entwickeln. Zusätzlich zum C#-Code kann auch JavaScript-Code aufgerufen werden. Weiters ist es möglich, statt JavaScript auch TypeScript-Code zu verwenden. Hierbei wird dieser beim Kompilieren des C#-Codes zu JavaScript transpiliert. Obwohl JavaScript-Code im Framework verwendet werden kann, ist für die Grundfunktionen des Frameworks C# notwendig, weshalb für das Framework die Programmiersprache C# beherrscht werden muss (Microsoft 2020).

7.4.2 Beliebtheit

Wie bei den vorherigen Frameworks wird auch das Open-Source-Projekt Blazor über die Versionsverwaltung von GitHub verwaltet. Das Framework erhielt dabei auf dieser Plattform rund 9.300 Sterne von den angemeldeten Nutzer*innen (GitHub 2020b).

7.4.3 Community

Auf der Plattform GitHub ist ersichtlich, dass bereits 74 Entwickler*innen außerhalb des Kern-Entwicklungsteams am Framework aktiv mitgestaltet haben (GitHub 2020b).

7.4.4 Lernkurve

Obwohl das Framework einen hohen Funktionsumfang aufweist, können viele Funktionen intuitiv und im Vergleich zu anderen Frameworks mit wenig notwendiger Programmierung umgesetzt werden. Ein Beispiel hierfür ist das Routing, das nur durch die Angabe von „@page“ und der dazugehörigen gewünschten Route umgesetzt wird. Um das Framework anwenden zu können, sind jedoch C#-Kenntnisse notwendig, die jedoch im Bereich der Webentwicklung eher unüblich sind. Bei der Entwicklung des Prototyps waren diese Kenntnisse bereits vorhanden, was den Einstieg in das Framework erleichterte. Aufgrund der Vorkenntnis wird die Bewertung daher abgewertet und mit 3 von 5 Sternen bewertet.

7.4.5 Lernunterstützung

Auf der offiziellen Webseite des Frameworks wird von Microsoft ein gutes Einführungsbeispiel angeboten. Hierin wird eine Webapplikation mit einer Aufgabenliste erstellt. Das Beispiel sollte aber noch erweitert werden, sodass alle Funktionen des Frameworks darin integriert sind. Das Framework wird hinsichtlich des Kriteriums der Lernunterstützung mit 4 von 5 Sternen bewertet.

7.4.6 Performanz

Für den Vergleich der Frameworks Angular, React und Vue.js wurde ein Benchmark herangezogen. Auch für die Bewertung von Blazor wird dieser Benchmark gleich wie bei den anderen Frameworks verwendet. Bei Blazor bildet sich der arithmetische Mittelwert aus den Werten 2,83, 11,93 und 1,24 und beträgt somit 5,33. Für die Bewertung wird dieser Wert wiederrum invertiert und beträgt 0,19 (Krause 2020).

7.4.7 Testbarkeit

Es ist möglich automatisierte Unit-Tests und End-to-End Tests mit dem Framework zu erstellen. Allerdings gibt es für das Testen von Komponenten kein offizielles Testframework von Microsoft selbst. Auch die Dokumentation von Tests mit Blazor ist ausbaufähig. Die Testbarkeit für das Framework wird somit mit 3 von 5 Sternen bewertet.

7.4.8 Dokumentation

Die Dokumentation ist in die ASP.NET-Dokumentation integriert. Teilweise sind Dokumentationen zum Blazor Framework in mehreren Unterkategorien des ASP.NET-Frameworks verstreut, was die Übersichtlichkeit stark reduziert. Für einige Teilgebiete des Frameworks werden leider keine Beispiele angeführt. Auf Grafiken für Grundkonzepte, die für die Verständlichkeit hilfreich wären, wurde verzichtet. Die Dokumentation des Frameworks wird daher mit 3 von 5 Sternen bewertet.

7.4.9 Installationsaufwand

Mit dem dotnet-CLI können Projekte unkompliziert mit Blazor erstellt werden. Bestimmte Tools sind dafür nicht notwendig. Lediglich das .NET-Software-Development-Kit (SDK) muss installiert werden. Der Installationsaufwand wird daher mit 4 von 5 Sternen bewertet.

7.4.10 Entwicklungsgeschwindigkeit

Durch den hohen Funktionsumfang können Funktionen einfach integriert werden. Die Syntax ist intuitiv und, wie beim Kriterium der Lernkurve bereits erwähnt, äußerst code-sparend umgesetzt. Die Entwicklungsgeschwindigkeit des Frameworks wird mit 5 von 5 Sternen bewertet.

7.4.11 Funktionsumfang

Das Framework bietet 14 von 16 Funktionen an, die in der nachfolgenden Tabelle aufgelistet werden. Die unterstützten Funktionen wurden dabei von der offiziellen Webseite des Frameworks entnommen (Microsoft 2020).

Funktion	Inkludiert
Routing ohne Serveranfragen	Ja
Komponenten / Templates	Ja
Datenbindungen unidirektional	Ja
Datenbindungen bidirektional	Ja
Ereignisbindungen	Ja
Pipes	Ja
Direktiven	Ja
Dependency Injection	Ja
Internationalisierung	Ja
Formulare	Nein
Validierung	Ja
Animationen	Nein
HTTP-Client	Ja
Logging	Ja
Lifecycle	Ja
Authentifizierung	Ja

Tabelle 5: Funktionsumfang des Framework Blazor (Eigene Darstellung)

7.4.12 Erweiterbarkeit

Es besteht die Möglichkeit, mithilfe sogenannter Extensions für das Framework anhand von NuGet-Packages die Webapplikation bzw. den Funktionsumfang des Frameworks zu erweitern. Derzeit sind aber, vor allem im Vergleich zu den anderen Frameworks, nur wenige Open-Source-Extensions zu finden. Die Erweiterbarkeit des Frameworks wird daher mit 3 von 5 Sternen bewertet.

7.4.13 Browserkompatibilität

Ab der Unterstützung von WebAssembly ist der SPA-Betrieb möglich. WebAssembly ist in jeden der meistgenutzten Browser (Google Chrome, Mozilla Firefox, Apple Safari und Microsoft Edge) ab 2017 integriert. Für den Vorgänger von Microsoft Edge, den Internet Explorer, wird das Framework nicht mehr unterstützt (Microsoft 2020).

7.4.14 Support

Ähnlich der vorherigen Frameworks wird auch beim Framework Blazor die kostenfreie Unterstützung durch die Community angeboten. Die Community des Frameworks ist im Vergleich zu den anderen Frameworks jedoch noch eher klein. Anders als bei den anderen Frameworks besteht bei Blazor die Möglichkeit, direkten Hersteller-Support von Microsoft zu bekommen. Dieser ist jedoch kostenpflichtig und wird zumeist über Partnerprogramme geregelt. Zusätzlich wird auch durch externe Consulting-Unternehmen kostenpflichtige Unterstützung angeboten. Die Anzahl dieser Consulting-Unternehmen ist aber im Vergleich zu den anderen Frameworks geringer.

7.4.15 Personalbedarf

Aktuell werden in Österreich auf der Jobplattform devjobs.at 2 Jobs für das Framework Blazor angeboten. Insgesamt sind 3 Entwicklerteams auf der Plattform registriert, die dieses Framework verwenden. Das Jahresdurchschnittsgehalt der angebotenen Jobs für das Framework beträgt 35.000 Euro.

7.5 Überblickstabelle

Um die einzelnen Bewertungen aller Kriterien der Frameworks übersichtlich zusammen zu fassen, werden diese in der Tabelle 6 gegenübergestellt. Dabei wurde aus Gründen der Übersichtlichkeit das Kriterium Funktionsumfang zusammengefasst und nur die Anzahl der verfügbaren Funktionen dargestellt.

Kriterium	**Angular**	**React**	**Vue.js**	**Blazor**
Programmiersprache	TS	TS, JS	TS, JS	C#
Beliebtheit	68.200	160.000	176.000	9.300
Community	1.279	1.522	382	74
Lernkurve	3	4	5	3
Lernunterstützung	5	3	4	4
Performanz	0,62	0,8	0,93	0,19
Testbarkeit	5	4	5	3
Dokumentation	4	4	5	3
Installationsaufwand	4	4	5	4
Entwicklungsgeschwindigkeit	5	3	5	5
Funktionsumfang	14/16	10/16	12/16	14/16
Erweiterbarkeit	5	4	5	3
Browserkompatibilität	2011	2011	2011	2017
Support	Community, Drittanbieter	Community, Drittanbieter	Community, Drittanbieter	Community, Drittanbieter, Hersteller
Personalbedarf	242	155	85	2

Tabelle 6: Überblick der Bewertungen je Framework (Eigene Darstellung)

8 ENTSCHEIDUNGSMODELL

Das zentrale Artefakt der Masterarbeit ist die Entwicklung eines systematischen Entscheidungsmodells für die Auswahl eines Software Frameworks. Die Entwicklung des Modells soll zur Beantwortung der Forschungsfrage, ob ein systematisches Entscheidungsmodell bei der Auswahl eines SPA-Frameworks unterstützen kann, beitragen. In diesem Kapitel werden Ziel und Aufbau eines solchen Entscheidungsmodells sowie das vollständige Modell anhand eines fiktiven Beispielprojekts erläutert.

8.1 Ziel des Entscheidungsmodells

Das Entscheidungsmodell soll die Sortierung der Frameworks anhand der individuellen Projektwichtigkeit ermöglichen. Zusätzlich soll das Modell die Frameworks anhand von Muss-Kriterien filtern können. Als Grundlage des Entscheidungsmodell dienen die in Kapitel 4 ausgewählten Frameworks, die in Kapitel 5 ausgewählten Kriterien und die in Kapitel 7 durchgeführte Bewertung der einzelnen Frameworks. Mit den Ergebnissen dieser 3 Kapitel soll ein Entscheidungsmodell entwickelt werden, das zur Sortierung und Filterung anhand unterschiedlicher und individuellen Projektanforderungen einheitlich angewendet werden kann. In der Abbildung 31 wird das Ziel des Entscheidungsmodell, die Umwandlung der Projektanforderungen zu einer Entscheidungsgrundlage, grafisch dargestellt.

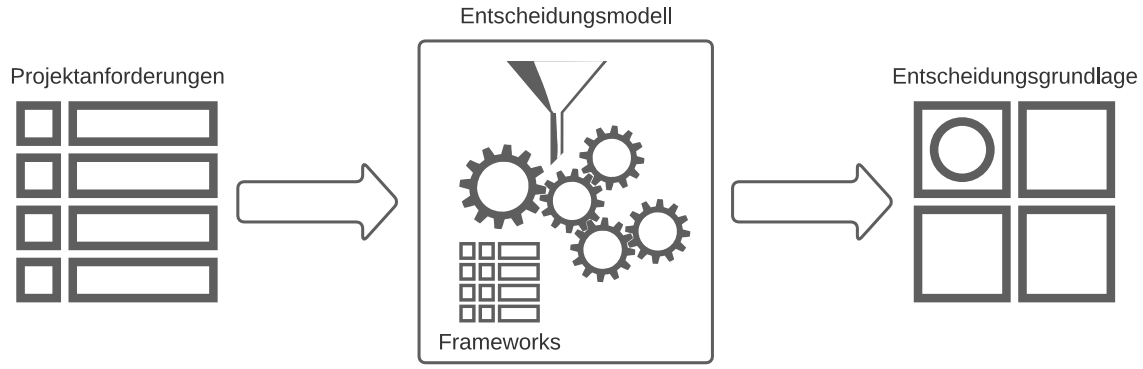


Abbildung 31: Ziel des Entscheidungsmodells (Eigene Darstellung)

8.2 Aufbau des Modells

Für die Anwendung des Modells müssen die Projektanforderungen des spezifischen Projekts an das Entscheidungsmodell übergeben werden. Anhand dieser Projektanforderungen erfolgt die Filterung und Sortierung der Frameworks anhand der Eignung des Projekts. Bei der Anwendung müssen im ersten Schritt Muss-Kriterien festgelegt werden. Anschließend wird die Projektwichtigkeit aller Kriterien anhand einer Skala von 0 bis 10 festgelegt. Dabei entspricht die Angabe einer Wichtigkeit von 0 der vollständigen Irrelevanz für das Projekt, während ein Wert von 10 die höchste Projektrelevanz darstellt. Nach der Festlegung der Muss-Kriterien und der Wichtigkeit der Kriterien für das Projekt können die Frameworks gefiltert und nach Eignung sortiert werden. Die Filterung erfolgt dabei durch Ausscheiden aller Frameworks, die die festgelegten Muss-Kriterien nicht erfüllen. Die verbleibenden Frameworks werden anschließend mithilfe einer Nutzwertanalyse sortiert. Der Nutzwert des Frameworks wird aus der Summe der Nutzwerte der einzelnen Kriterien gebildet, die sich für jedes Framework aus der Multiplikation der festgelegten Projektwichtigkeit mit der Bewertung des jeweiligen Kriteriums errechnen. Dabei ist es wichtig, zu beachten, dass die bewerteten Faktoren normalisiert werden, damit die Gewichtung der einzelnen Kriterien ausgeglichen ist. Die Auswertung kann entweder manuell durchgeführt oder programmatisch mithilfe eines Programms oder einer Kalkulationstabelle automatisiert werden. Die Frameworks sind nach der Auswertung anhand des Nutzwertes hinsichtlich der Eignung für das Projekt absteigend sortiert. Das Framework mit dem höchsten Nutzwert eignet sich also am besten für die festgelegten Anforderungen. Bei der Auswahl von vielen Muss-Kriterien kann es vorkommen, dass sich kein Framework für das Projekt eignet, da keines der im Modell enthaltenen Frameworks alle Kriterien unterstützt. Nachfolgend werden die Schritte zur Aufstellung und der Anwendung des Modells nochmals grob zusammengefasst. Die Aufstellung des Modells ist dabei für ausgewählte Frameworks nur einmalig notwendig und kann anschließend für individuelle Projekte beliebig oft angewendet werden.

Aufstellung des Modells:

1. Auswahl und Filterung relevanter Frameworks
2. Festlegen von Bewertungskriterien
3. Bewertung ausgewählter Frameworks
4. Normierung der Bewertungsfaktoren

Anwendung des Modells:

1. Festlegen der Muss-Kriterien des Projektes
2. Festlegen der Projektwichtigkeiten des Projektes
3. Filterung der Frameworks anhand der Muss-Kriterien
4. Ermitteln der einzelnen Nutzwerte durch Multiplikation der Wichtigkeit mit der Bewertung

5. Ermitteln der gesamten Nutzwerte durch Addition der einzelnen Nutzwerte
6. Sortierung der Frameworks anhand der ermittelten Nutzwerte
7. Ergebnis für die Auswahl oder als Entscheidungsgrundlage verwenden

8.3 Normalisierung der Bewertung

Um alle Kriterien des Modells mit gleicher Gewichtung zu betrachten, müssen die Bewertungen normalisiert werden. Ohne diesen Schritt würden bei Kriterien mit unterschiedlichem Minimum und Maximum bei gleicher Projektwichtigkeit unterschiedliche Nutzwerte resultieren. Bei einer Wichtigkeit von 10 betrüge also beispielsweise der Nutzwert des Performanz-Kriteriums bei einem Maximalwert von 5 maximal 50, während bei gleicher Projektwichtigkeit beim Kriterium Beliebtheit mit einem Maximalwert von 175.000 der Nutzwert bis zu 1.750.000 betragen kann. Das Kriterium der Beliebtheit würde also ohne Normierung der Faktoren im Entscheidungsmodell dominieren. Um die Werte der unterschiedlichen Kriterien auf eine Skala von 0 bis 10 zu normalisieren wird folgende Formel angewendet:

$$X_n = 10 \times \frac{X}{\max(X)}$$

Durch die Division der Bewertung X durch das Maximum aller Bewertungen des Kriteriums ergibt sich ein Wert zwischen 0 und 1, der durch die Multiplikation mit 10 auf eine Skala von 0 bis 10 transformiert wird. Das Framework mit der höchsten Bewertung des jeweiligen Kriteriums erhält somit stets die normalisierte Bewertung X_n von 10.

In der nachfolgenden Tabelle wird die Normalisierung derjenigen Frameworkbewertungen dargestellt, die für das Entscheidungsmodell verwendet werden.

Bewertete Kriterien	**Bewertung**				**Normalisierte Bewertung**			
Kriterium	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Beliebtheit	68.200	160.000	176.000	9.300	3,9	9,1	10	0,5
Community	1.279	1.522	382	74	8,4	10	2,5	0,5
Lernkurve	3	4	5	3	6	8	10	6
Lernunterstützung	5	3	4	4	10	6	8	8
Performanz	0,62	0,8	0,93	0,19	6,7	8,6	10	2
Testbarkeit	5	4	5	3	10	8	10	6
Dokumentation	4	4	5	3	8	8	10	6
Installationsaufwand	4	4	5	4	8	8	10	8
Entwicklungsgeschwindigkeit	5	3	5	5	10	6	10	10
Erweiterbarkeit	5	4	5	3	10	8	10	6
Personalbedarf	242	155	85	2	10	6,4	3,5	0,1

Tabelle 7: Normalisierung der bewerteten Kriterien (Eigene Darstellung)

8.4 Fertiges Entscheidungsmodell

In der nachfolgenden Tabelle ist die Vorlage für die Anwendung des Entscheidungsmodells ersichtlich. Um das Entscheidungsmodell anzuwenden, muss im ersten Schritt die Spalte „Projektanforderungen“ ausgefüllt werden. Dabei muss in jedes Feld die Projektwichtigkeit zwischen 0 und 10 eingetragen werden. Handelt es sich um ein Muss-Kriterium, so kann dies gekennzeichnet werden und eine Bewertung dieses Kriteriums ist nicht notwendig. Nach dem Ausfüllen dieser Spalte können Frameworks, die die Muss-Kriterien nicht erfüllen, gestrichen werden. Anschließend können die normalisierten Bewertungen mit der Projektanforderung multipliziert und so die Nutzwerte der einzelnen Frameworks ermittelt werden. Die errechneten Nutzwerte werden in der letzten Zeile je Framework auf einen gesamten Nutzwert summiert, mit dem die Frameworks abschließend sortiert werden.

	Projektanforderung		Normalisierte Bewertung				Nutzwert			
			Angular	React	Vue.js	Blazor	Angular	React	Vue.js	Blazor
Programmiersprache										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
TypeScript			JA	JA	JA	NEIN				
JavaScript			NEIN	JA	JA	NEIN				
C#			NEIN	NEIN	NEIN	JA				
Funktionsumfang										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Routing ohne Serveranfragen			JA	JA	JA	JA				
Komponenten / Templates			JA	JA	JA	JA				
Datenbindungen unidirektional			JA	JA	JA	JA				
Datenbindungen bidirektional			JA	NEIN	JA	JA				
Ereignisbindungen			JA	JA	JA	JA				
Pipes			JA	NEIN	JA	JA				
Direktiven			JA	NEIN	JA	JA				
Dependency Injection			JA	NEIN	JA	JA				
Internationalisierung			JA	JA	JA	JA				
Formulare			JA	JA	NEIN	NEIN				
Validierung			JA	JA	JA	JA				
Animationen			JA	JA	JA	NEIN				
HTTP-Client			JA	NEIN	NEIN	JA				
Logging			NEIN	NEIN	NEIN	JA				
Lifecycle			JA	JA	JA	JA				
Authentifizierung			NEIN	JA	NEIN	JA				
Browserkompatibilität										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Browser älter als 2017			JA	JA	JA	NEIN				
Support										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Support durch den Herststeller			NEIN	NEIN	NEIN	JA				
Bewertete Kriterien										
Kriterium	**Bewertung 0 - 10**		**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Beliebtheit			3,9	9,1	10	0,5				
Community			8,4	10	2,5	0,5				
Lernkurve			6	8	10	6				
Lernunterstützung			10	6	8	8				
Performanz			6,7	8,6	10	2				
Testbarkeit			10	8	10	6				
Dokumentation			8	8	10	6				
Installationsaufwand			8	8	10	8				
Entwicklungsgeschwindigkeit			10	6	10	10				
Erweiterbarkeit			10	8	10	6				
Personalbedarf			10	6,4	3,5	0,1				
						SUMME:				

Tabelle 8: Tabelle zur Anwendung des Entscheidungsmodells (Eigene Darstellung)

8.5 Anwendungsbeispiel

Um ein besseres Verständnis für die Anwendung des entwickelten Entscheidungsmodells zu schaffen wird dieses anhand eines exemplarischen Projekts angewandt und ein Framework ausgewählt.

Als exemplarisches Projekt wird die Entwicklung eines Webshops herangezogen. Die Entwickler*innen des Projekts sind auf TypeScript spezialisiert und möchten nicht auf eine andere Programmiersprache wechseln. Seitens der Auftraggeber*innen besteht die Anforderung, den Webshop auf modernen Browsern der letzten 5 Jahre, also ab 2015, aufrufen zu können. Da in den Webshop viele Funktionen integriert werden, sollen diese möglichst gut durch das Framework unterstützt werden. In der nachfolgenden Tabelle ist eine Bewertung der Kriterien für das exemplarische Projekt angeführt.

	Projektanforderung	
Programmiersprache		
Kriterium	**Bewertung 0 - 10**	**Muss**
TypeScript	-	JA
JavaScript	0	NEIN
C#	0	NEIN
Funktionsumfang		
Kriterium	**Bewertung 0 - 10**	**Muss**
Routing ohne Serveranfragen	7	NEIN
Komponenten / Templates	10	NEIN
Datenbindungen unidirektional	10	NEIN
Datenbindungen bidirektional	7	NEIN
Ereignisbindungen	8	NEIN
Pipes	4	NEIN
Direktiven	2	NEIN
Dependency Injection	5	NEIN
Internationalisierung	10	NEIN
Formulare	9	NEIN
Validierung	9	NEIN
Animationen	3	NEIN
HTTP-Client	6	NEIN
Logging	5	NEIN
Lifecycle	2	NEIN
Authentifizierung	7	NEIN
Browserkompatibilität		
Kriterium	**Bewertung 0 - 10**	**Muss**
Browser älter als 2017	-	JA
Support		
Kriterium	**Bewertung 0 - 10**	**Muss**
Support durch den Herststeller	3	NEIN
Bewertete Kriterien		
Kriterium	**Bewertung 0 - 10**	
Beliebtheit	6	
Community	8	
Lernkurve	9	
Lernunterstützung	9	
Performanz	7	
Testbarkeit	6	
Dokumentation	9	
Installationsaufwand	0	
Entwicklungsgeschwindigkeit	8	
Erweiterbarkeit	3	
Personalbedarf	1	

Tabelle 9: Festlegen der Projektanforderungen eines exemplarischen Projekts (Eigene Darstellung)

In den Projektanforderungen sind zwei Muss-Kriterien festgelegt. Diese werden, wie in der nachfolgenden Tabelle abstrahiert dargestellt, mit den bewerteten Frameworks verglichen, um dadurch Frameworks, die die Kriterien nicht erfüllen, auszuscheiden. Im aktuellen Beispiel wird durch die gegebenen Muss-Kriterien das Framework Blazor ausgeschieden, weshalb es für die weiteren Schritte vernachlässigt werden kann.

	Projektanforderung		Normalisierte Bewertung			
			Angular	**React**	**Vue.js**	**Blazor**
Programmiersprache						
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**
TypeScript	-	JA	JA	JA	JA	NEIN
Browserkompatibilität						
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**
Browser älter als 2017	-	JA	JA	JA	JA	NEIN

Tabelle 10: Ausscheiden der Frameworks durch Abgleich der Muss-Kriterien (Eigene Darstellung)

Nach der Filterung der Frameworks durch die Muss-Kriterien erfolgt die Ermittlung der Nutzwerte für die restlichen Frameworks. In der nachfolgenden Tabelle ist die vollständig ausgefüllte Tabelle für die Ermittlung des für das exemplarische Projekt bestgeeigneten Frameworks ersichtlich. Aus der Tabelle geht hervor, dass Angular den höchsten Nutzwert für die gegebenen Projektanforderungen aufweist und sich laut Entscheidungsmodell am besten für das fiktive Projekt eignen würde.

	Projektanforderung		Normalisierte Bewertung				Nutzwert			
			Angular	React	Vue.js	Blazor	Angular	React	Vue.js	Blazor
Programmiersprache										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
TypeScript	-	JA	JA	JA	JA	NEIN	10	10	10	
JavaScript	0	NEIN	NEIN	JA	JA	NEIN	0	10	10	
C#	0	NEIN	NEIN	NEIN	NEIN	JA	0	0	0	
Funktionsumfang										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Routing ohne Serveranfragen	7	NEIN	JA	JA	JA	JA	70	70	70	
Komponenten / Templates	10	NEIN	JA	JA	JA	JA	100	100	100	
Datenbindungen unidirektional	10	NEIN	JA	JA	JA	JA	100	100	100	
Datenbindungen bidirektional	7	NEIN	JA	NEIN	JA	JA	70	0	70	
Ereignisbindungen	8	NEIN	JA	JA	JA	JA	80	80	80	
Pipes	4	NEIN	JA	NEIN	JA	JA	40	0	40	
Direktiven	2	NEIN	JA	NEIN	JA	JA	20	0	20	
Dependency Injection	5	NEIN	JA	NEIN	JA	JA	50	0	50	
Internationalisierung	10	NEIN	JA	JA	JA	JA	100	100	100	
Formulare	9	NEIN	JA	JA	NEIN	NEIN	90	90	0	
Validierung	9	NEIN	JA	JA	JA	JA	90	90	90	
Animationen	3	NEIN	JA	JA	JA	NEIN	30	30	30	
HTTP-Client	6	NEIN	JA	NEIN	NEIN	JA	60	0	0	
Logging	5	NEIN	NEIN	NEIN	NEIN	JA	0	0	0	
Lifecycle	2	NEIN	JA	JA	JA	JA	20	20	20	
Authentifizierung	7	NEIN	NEIN	JA	NEIN	JA	0	70	0	
Browserkompatibilität										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Browser älter als 2017	-	JA	JA	JA	JA	NEIN	10	10	10	
Support										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Support durch den Herststeller	3	NEIN	NEIN	NEIN	NEIN	JA	0	0	30	
Bewertete Kriterien										
Kriterium	**Bewertung 0 - 10**		**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Beliebtheit	6		3,9	9,1	10	0,5	23,3	54,5	60	
Community	8		8,4	10	2,5	0,5	67,2	80	20,1	
Lernkurve	9		6	8	10	6	54	72	90	
Lernunterstützung	9		10	6	8	8	90	54	72	
Performanz	7		6,7	8,6	10	2	46,7	60,2	70	
Testbarkeit	6		10	8	10	6	60	48	60	
Dokumentation	9		8	8	10	6	72	72	90	
Installationsaufwand	0		8	8	10	8	0	0	0	
Entwicklungsgeschwindigkeit	8		10	6	10	10	80	48	80	
Erweiterbarkeit	3		10	8	10	6	30	24	30	
Personalbedarf	1		10	6,4	3,5	0,1	10	6,4	3,5	
						SUMME:	1473,1	1299,2	1405,6	

Tabelle 11: Ergebnis des Entscheidungsmodells für das exemplarische Projekt (Eigene Darstellung)

9 VALIDIERUNG ANHAND DER UMFRAGEERGEBNISSE

Um die Validität des in Kapitel 8 entwickelten Entscheidungsmodells zu prüfen, wird eine Umfrage in einer bestimmten Zielgruppe durchgeführt, deren Ergebnisse mit dem zuvor erstellten Modell verglichen werden, um infolgedessen eine Aussage hinsichtlich der Validität des Entscheidungsmodells treffen zu können.

9.1 Ziel der Umfrage

Ziel der Umfrage ist die Überprüfung der Tauglichkeit des Entscheidungsmodells. Diese wird durch den Vergleich des Modells mit Erfahrungen von Entwickler*innen ermittelt. Zusätzlich soll der Standpunkt zur Komplexität der Auswahl eines SPA-Frameworks sowie der Bedarf eines Entscheidungsmodells eruiert werden. Die Ergebnisse der Umfrage dienen bei der Beantwortung der Forschungsfrage als zusätzliche Stütze.

9.2 Vorbereitung der Umfrage

Vor der Aussendung der Umfrage sind als Vorbereitung die Auswahl des Umfrage-Tools, die Auswahl und Formulierung der Fragen, die Auswahl der Antwortmöglichkeiten, die Auswahl der Zielgruppe und der Verbreitungskanäle sowie die Testung der Umfrage notwendig.

9.2.1 Auswahl der Fragen

Die Auswahl der Fragen erfolgt durch das Ableiten des Ziels der Umfrage. Um einen Vergleich zwischen dem entwickelten Entscheidungsmodell und der Umfrage ziehen zu können werden in der Umfrage dieselben Frameworks wie im Entscheidungsmodell verwendet. Folgende Fragen sind in der Umfrage enthalten und werden teilweise je nach Beantwortung dynamisch angezeigt:

1. Welche berufliche Tätigkeit trifft bei Ihnen am ehesten zu?
2. Wie lange sind Sie bereits im Bereich der Softwareentwicklung tätig?
3. Haben Sie bereits Erfahrungen mit SPA-Frameworks gesammelt?
4. Waren Sie bereits an der Auswahl eines Frameworks beteiligt?
5. Wie komplex würden Sie die Auswahl eines optimalen SPA-Frameworks für ein Projekt beurteilen?
6. Würden Sie ein Entscheidungsmodell für die Auswahl eines Frameworks verwenden?

7. Bitte begründen Sie kurz, weshalb Sie kein Entscheidungsmodell verwenden würden.
8. Welche Kriterien sind Ihnen bei der Auswahl eines Frameworks besonders wichtig?
9. In den folgenden Fragen haben Sie die Möglichkeit, ein SPA-Framework, welches Sie bereits für ein Projekt eingesetzt haben, zu bewerten. Falls Sie keines der angeführten Frameworks kennen, wählen Sie "Keines" aus. Welches Framework möchten Sie bewerten? Falls Sie mehrere Frameworks bewerten möchten, führen Sie die Umfrage bitte im Anschluss erneut durch.
10. Was wurde mit dem Framework umgesetzt? (z. B. Webshop, Dashboard für Maschinenübersicht etc.)
11. War das Framework im Nachhinein gesehen die richtige Entscheidung für das Projekt?
12. Bitte begründen Sie kurz Ihre Auswahl.
13. Was waren Gründe, warum Sie sich für das Framework entschieden haben?
14. Wie hoch würden Sie den Installationsaufwand für die Einrichtung des Frameworks beurteilen?
15. Wie einfach würden Sie den Einstieg in das Framework bewerten?
16. Wie gut wurden Unterstützungen wie Tutorials, Beispiele und Schulungen vom Framework angeboten?
17. Wie würden Sie die Zeitersparnis bei der Projektumsetzung durch das Framework im Vergleich zur Entwicklung ohne Framework bewerten?
18. Wie gut würden Sie die Dokumentation des Frameworks bewerten?
19. Wie performant würden Sie das Framework hinsichtlich Ladezeiten bewerten?
20. Wie würden Sie die Testbarkeit des Frameworks bewerten?
21. Wie würden Sie den Support des Frameworks bewerten?
22. Wie würden Sie den Funktionsumfang des Frameworks bewerten?
23. Haben Sie noch zusätzliche Anmerkungen?

9.2.2 Auswahl der Antwortmöglichkeiten

Als Antwortmöglichkeiten wurden entweder Single-Choice, Multiple-Choice, Skalen von 1 bis 5 oder Freitext angeboten. Aufgrund der Kombination verschiedener Frageformen handelt es sich bei der Umfrage weder um eine rein quantitative noch eine rein qualitative Erhebung. In der Tabelle 14 im Anhang werden die verwendeten Antwortmöglichkeiten dargestellt. Die Nummer entspricht dabei den in Kapitel 9.2.1 aufgelisteten Fragen.

Weiters wurden bei Single-Choice- und Multiple-Choice-Fragen Antwortmöglichkeiten vorgegeben, die teilweise beim Ausfüllen erweitert werden konnten. In der Tabelle 15 im Anhang sind diese Antwortmöglichkeiten aufgelistet.

9.2.3 Auswahl der Zielgruppe

Ziel der Umfrage ist die Sammlung von Erfahrungen für SPA-Frameworks. Diesbezüglich ist es für die Beantwortung der framework-spezifischen Fragen erforderlich, zumindest eines der Frameworks bereits verwendet zu haben. Die Umfrage richtet sich daher primär an Personen aus dem Web-Entwicklungsumfeld, die sich direkt oder indirekt mit der Anwendung oder der Auswahl eines SPA-Frameworks auseinandersetzen. Dies kann im beruflichen, privaten oder ausbildungstechnischen Umfeld der Fall sein.

Um möglichst viele Umfrageergebnisse innerhalb der Zielgruppe zu bekommen wird die Umfrage in mehreren Web-Entwicklungs-Firmen, über Studentenverteiler sowie unter Bekannten aus dem Web-Umfeld verteilt.

9.3 Auswertung der Ergebnisse

Bei der Umfrage wurden 73 Umfrageergebnisse erzielt. Der größte Anteil mit 64% der Teilnehmer*innen gab dabei, wie in Abbildung 32 grafisch dargestellt an, beruflich als Softwareentwickler*in tätig zu sein.

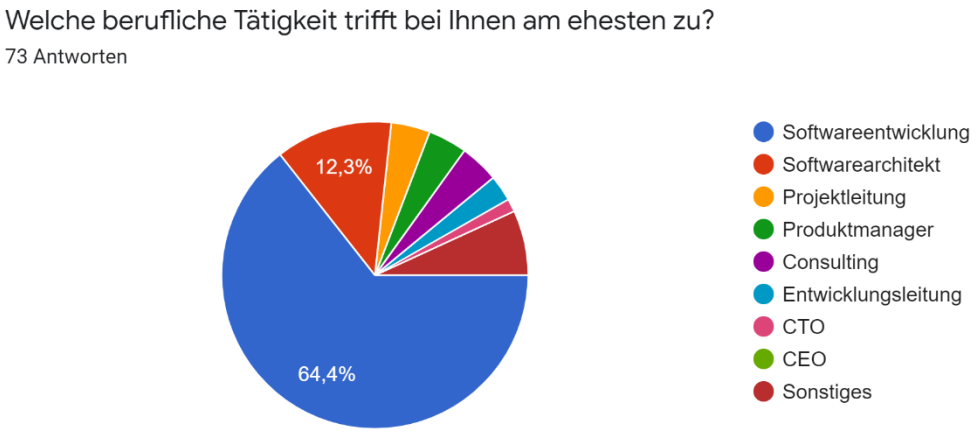


*Abbildung 32: Berufliche Tätigkeit der Umfrageteilnehmer*innen (Eigene Darstellung)*

Knapp ein Viertel der Teilnehmer*innen gab, wie in Abbildung 33 ersichtlich an, weniger als 2 Jahre im Bereich der Softwareentwicklung tätig zu sein. Die restlichen Teilnehmer gaben an, länger als 2 Jahre in diesem Beriech zu sein, wobei knapp 10% der Befragten angaben, mehr als 10 Jahre Erfahrung zu haben.

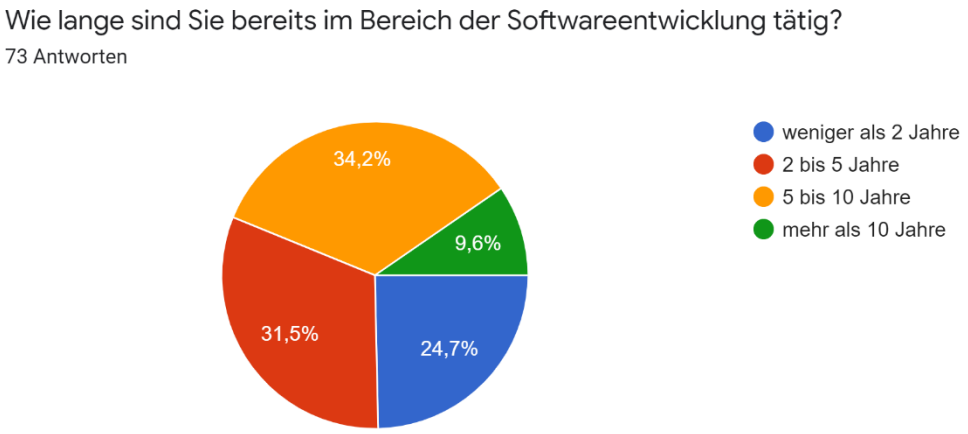


*Abbildung 33: Berufserfahrung der Umfrageteilnehmer*innen (Eigene Darstellung)*

Die Frage, ob die Teilnehmer*innen bereits Erfahrungen mit SPA-Frameworks gesammelt haben, bejahten mehr als zwei Drittel der Teilnehmer*innen. Mehr als die Hälfte, und damit die meisten der Befragten gaben an, mit Angular bereits Erfahrungen gesammelt zu haben. In der nachfolgenden Abbildung wird die Aufteilung der Erfahrungen mit teilweise von den Befragten selbst hinzugefügten Frameworks grafisch dargestellt.

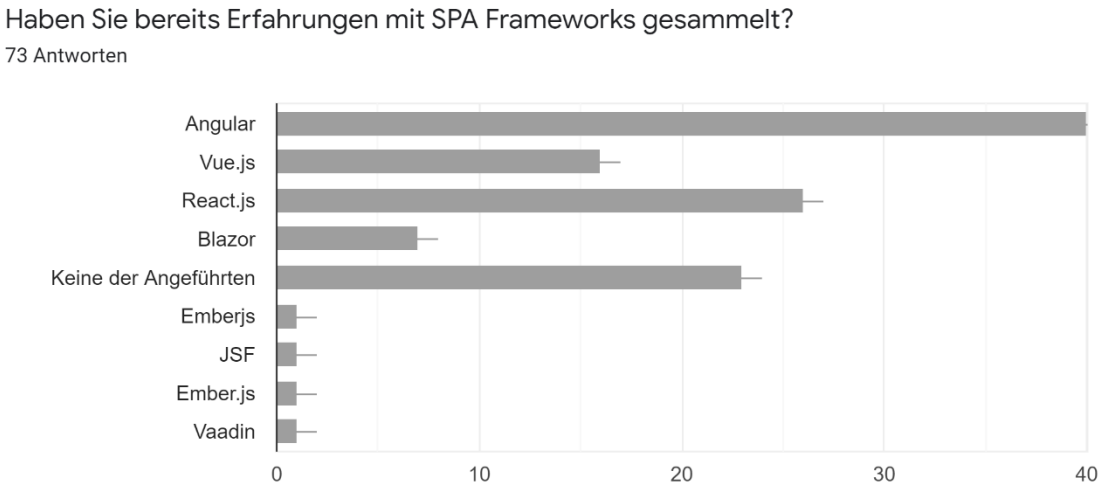


*Abbildung 34: Erfahrungen der Umfrageteilnehmer*innen mit SPA-Frameworks (Eigene Darstellung)*

Bei der Auswahl eines Frameworks gaben 41% der Teilnehmer*innen an, an einer solchen Auswahl beteiligt gewesen zu sein. Die Mehrheit der Befragten war dies jedoch nicht. In Abbildung 35 ist ersichtlich, dass fast die Hälfte und damit die Mehrheit der Teilnehmer*innen die Komplexität der Auswahl mit 4 von 5 bewerten.

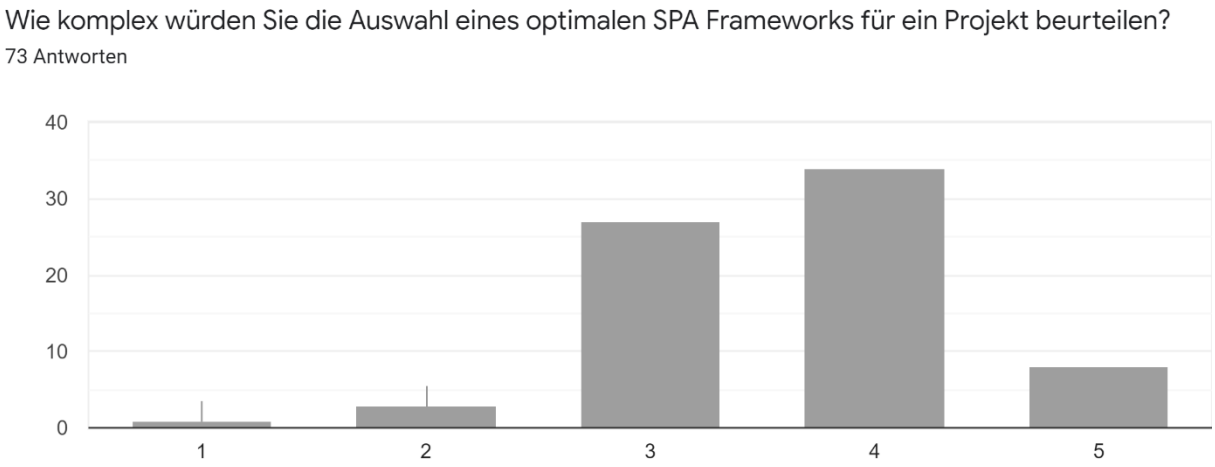


Abbildung 35: Komplexität der Auswahl eines SPA-Frameworks (Eigene Darstellung)

Aus Abbildung 36 geht hervor, dass lediglich drei Teilnehmer*innen würden für die Auswahl kein Entscheidungsmodell verwenden. Die meisten Teilnehmer*innen, etwa 47%, würden ein solches Modell als Entscheidungsgrundlage für die Auswahl eines Frameworks verwenden. Ein Grund gegen die Verwendung eines Entscheidungsmodells war, dass die objektiven Unterschiede der Frameworks zu gering seien.

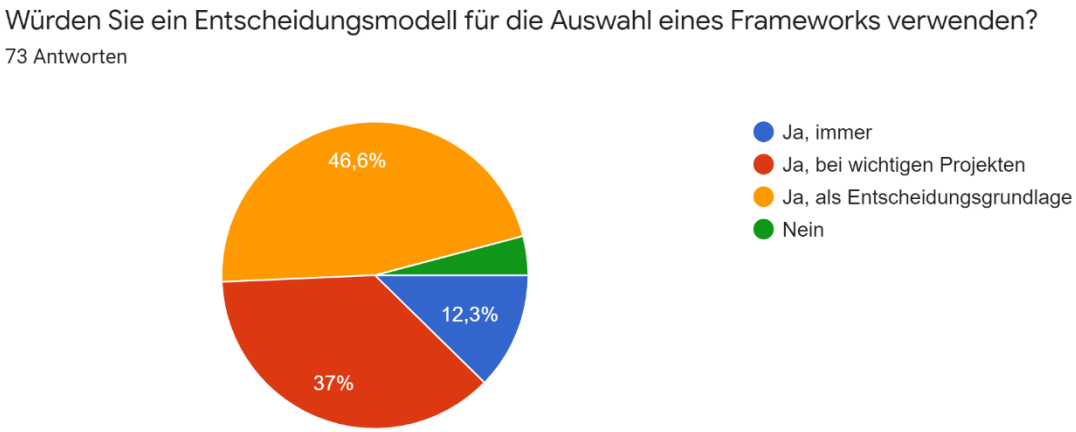


Abbildung 36: Bereitschaft zur Verwendung eines Entscheidungsmodells (Eigene Darstellung)

In der Umfrage wurde eruiert, welche Kriterien den Befragten bei der Auswahl eines Frameworks besonders wichtig sind. Dabei war mit 80 Antworten den meisten Teilnehmer*innen die Programmiersprache wichtig, gefolgt von der Performanz und einer guten Dokumentation des Frameworks mit jeweils 52 Antworten. In der nachfolgenden Abbildung werden alle Antworten aller Kriterien dargestellt.

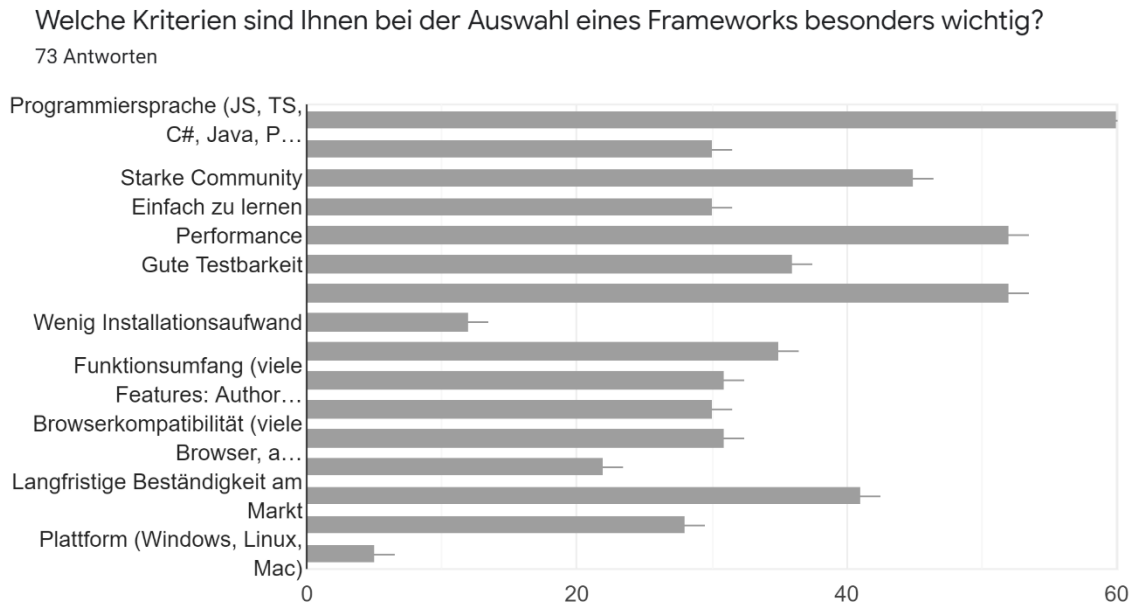


Abbildung 37: Wichtige Kriterien für die Auswahl eines Frameworks (Eigene Darstellung)

Bei der Umfrage konnten die Teilnehmer*innen ein SPA-Framework, das bereits für ein Projekt eingesetzt wurde, bewerten. Knapp 65% der Umfrageteilnehmer*innen haben ein Framework bewertet. Die meisten Bewertungen erhielt dabei Angular, gefolgt von React, Vue.js und Blazor. Die restlichen Teilnehmer*innen haben kein Framework bewertet. In der nachfolgenden Tabelle werden die gerundeten Mittelwerte der Framework-Bewertungen dargestellt. Die Bewertung, wie einfach das Framework zu installieren ist wurde dabei invertiert, da ein Framework mit weniger Installationsaufwand besser als ein Framework mit viel Installationsaufwand zu bewerten ist.

	Angular (N=21)	React (N=11)	Vue.js (N=8)	Blazor (N=6)
Installationsaufwand	3,6	4,2	4,1	3,8
Lernkurve	3,2	3,6	4,0	3,7
Lernunterstützung	4,5	4,3	4,3	3,5
Entwicklungsgeschwindigkeit	4,2	4,5	4,4	4,5
Dokumentation	4,2	4,5	4,5	3,8
Performanz	3,9	4,2	3,9	4,2
Testbarkeit	4,1	4,4	4,0	3,5
Support	4,2	3,7	3,5	3,3
Funktionsumfang	4,6	3,9	4,4	4,2

Tabelle 12: Gerundete Mittelwerte der in der Umfrage bewerteten Frameworks (Eigene Darstellung)

Zusammenfassend kann festgestellt werden, dass die Umfragen bei korrekter Angabe der beruflichen Tätigkeit und der Berufserfahrung vorwiegend durch Teilnehmer*innen aus der Zielgruppe beantwortet wurde. Weiters geht aus der Umfrage hervor, dass die Auswahl eines Frameworks komplex ist und dass die überwiegenden Mehrheit der Befragten ein Entscheidungsmodell für die Auswahl eines Frameworks verwenden würde. Bezugnehmend auf die Forschungsfrage untermauern diese Ergebnisse die Notwendigkeit eines solchen Modells.

9.4 Abgleich mit dem Entscheidungsmodell

In Kapitel 8 wurde ein Entscheidungsmodell für die Auswahl eines SPA-Frameworks entwickelt. Um dieses Modell hinsichtlich der Forschungsfrage zu validieren, ob ein Entscheidungsmodell bei der Auswahl eines Frameworks unterstützen kann, wird es anhand des in Kapitel 8.5 verwendeten Anwendungsbeispiels mit den bewerteten Kriterien der Umfrage angewendet und die Resultate verglichen.

Für die Anwendung des Entscheidungsmodells mit den Bewertungen aus den Umfrageergebnissen werden diese normalisiert. Anschließend werden diese anstelle der im Entscheidungsmodell vorhandenen Bewertungen eingesetzt. In der nachfolgenden Tabelle wird das Ergebnis der Anwendung des Entscheidungsmodells anhand der Bewertungen der Umfrageteilnehmer*innen sowie der Projektanforderungen aus dem exemplarischen Projekt dargestellt. Aus der Tabelle geht hervor, dass sich laut Entscheidungsmodell das Framework Angular am besten für das fiktive Projekt eignet. Am zweitbesten wäre Vue.js gefolgt von React und Blazor, wobei Letzteres ohnehin die Muss-Kriterien nicht erfüllt. Dasselbe Ergebnis resultierte auch aus der Anwendung des Entscheidungsmodells mit den originalen Parametern. Die Validität des Entscheidungsmodells konnte somit durch die Umfrage nicht widerlegt werden.

	Projektanforderung		Normalisierte Bewertung				Nutzwert			
			Angular	React	Vue.js	Blazor	Angular	React	Vue.js	Blazor
Programmiersprache										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
TypeScript	-	JA	JA	JA	JA	NEIN	10	10	10	
JavaScript	0	NEIN	NEIN	JA	JA	NEIN	0	10	10	
C#	0	NEIN	NEIN	NEIN	NEIN	JA	0	0	0	
Funktionsumfang										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Routing ohne Serveranfragen	7	NEIN	JA	JA	JA	JA	70	70	70	
Komponenten / Templates	10	NEIN	JA	JA	JA	JA	100	100	100	
Datenbindungen unidirektional	10	NEIN	JA	JA	JA	JA	100	100	100	
Datenbindungen bidirektional	7	NEIN	JA	NEIN	JA	JA	70	0	70	
Ereignisbindungen	8	NEIN	JA	JA	JA	JA	80	80	80	
Pipes	4	NEIN	JA	NEIN	JA	JA	40	0	40	
Direktiven	2	NEIN	JA	NEIN	JA	JA	20	0	20	
Dependency Injection	5	NEIN	JA	NEIN	JA	JA	50	0	50	
Internationalisierung	10	NEIN	JA	JA	JA	JA	100	100	100	
Formulare	9	NEIN	JA	JA	NEIN	NEIN	90	90	0	
Validierung	9	NEIN	JA	JA	JA	JA	90	90	90	
Animationen	3	NEIN	JA	JA	JA	NEIN	30	30	30	
HTTP-Client	6	NEIN	JA	NEIN	NEIN	JA	60	0	0	
Logging	5	NEIN	NEIN	NEIN	NEIN	JA	0	0	0	
Lifecycle	2	NEIN	JA	JA	JA	JA	20	20	20	
Authentifizierung	7	NEIN	NEIN	JA	NEIN	JA	0	70	0	
Browserkompatibilität										
Kriterium	**Bewertung 0 - 10**	**Muss**	**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Browser älter als 2017	-	JA	JA	JA	JA	NEIN	10	10	10	
Bewertete Kriterien										
Kriterium	**Bewertung 0 - 10**		**Angular**	**React**	**Vue.js**	**Blazor**	**Angular**	**React**	**Vue.js**	**Blazor**
Beliebtheit	6		3,9	9,1	10	0,5	23,3	54,5	60,0	
Community	8		8,4	10	2,5	0,5	67,2	80,0	20,1	
Lernkurve	9		6,4	7,2	8	7,3	57,4	64,8	72,0	
Lernunterstützung	9		9,0	8,5	8,5	7	81,4	76,9	76,5	
Performanz	7		7,7	8,4	7,7	8	53,9	58,5	54,0	
Testbarkeit	6		8,1	8,7	8	7	48,6	52,4	48,0	
Dokumentation	9		8,5	9,1	9	7,7	76,3	81,8	81,0	
Installationsaufwand	0		7,1	8,4	8,3	7,7	0,0	0,0	0,0	
Entwicklungsgeschwindigkeit	8		8,4	8,9	8,8	9	67,0	71,3	70,0	
Support	3		8,4	7,4	7	6,7	25,2	22,2	21,0	
Erweiterbarkeit	3		10	8	10	6	30,0	24,0	30,0	
Personalbedarf	1		10	6,4	3,5	0,1	10,0	6,4	3,5	
						SUMME:	1480,4	1372,9	1336,1	

Tabelle 13: Entscheidungsmodells anhand Bewertungen der Umfrage (Eigene Darstellung)

10 ERGEBNISSE

In diesem Kapitel sollen anhand der in dieser Masterarbeit erarbeiteten Kapitel die Ergebnisse dargestellt werden. Dies beinhaltet die Beantwortung der Forschungsfrage, die Einschränkungen hinsichtlich Limitierungen und ein Fazit samt Interpretation.

10.1 Beantwortung der Forschungsfrage

Im Kapitel 1.5 wurde hinsichtlich der Forschungsfrage folgende Hypothese H1 definiert:

Mithilfe eines systematischen Entscheidungsmodells kann die Auswahl eines geeigneten Single-Page-Application- Frameworks im Bereich Web-Frontendentwicklung getroffen werden.

Um diese Hypothese zu untersuchen wurde ein Entscheidungsmodell für eine solche Auswahl entwickelt und anhand eines fiktiven Beispiels angewendet. Es stellte sich im Zuge der schrittweisen Entwicklung des Modells heraus, dass die Erstellung eines systematischen Modells möglich ist. Die Frage des Nutzens im Sinne der Unterstützung bei der Auswahl eines Frameworks sowie der Eignung für die Praxis wird dabei noch nicht beantwortet.

Weiters wurde versucht, die Forschungsfrage mithilfe einer Umfrage mit Teilnehmer*innen aus der Softwareentwicklungsbranche zu beantworten. Die Umfrageergebnisse zeigten, dass der Bedarf und die Verwendungsbereitschaft eines Entscheidungsmodells als Unterstützung für die Auswahl eines Frameworks hoch sind. Dies Zeigt die Relevanz der Forschung im Rahmen dieser Arbeit auf.

Abschließend wurde durch eine Gegenüberstellung der Ergebnisse der Forschungsfrage mit dem davor entwickelten Entscheidungsmodell versucht, die Forschungsfrage zu beantworten. Dabei wurde das Entscheidungsmodell mit den Parametern der Umfrage angewandt und die Ergebnisse der Anwendungen des Modells verglichen. Dabei stellte sich heraus, dass bei beiden Anwendungen des Modells auf das fiktive Projekt jeweils dasselbe Framework als am besten geeignete vorgeschlagen wurde. Somit konnte die Hypothese H1 bestätigt werden.

10.2 Limitierungen

Im Rahmen der Masterarbeit wurde die Entwicklung des Entscheidungsmodells auf 4 Frameworks eingegrenzt. Die ausgewählten Frameworks weisen dabei von Haus aus nur geringe objektive Unterschiede auf, was eine klare Abgrenzung bei der Entscheidung erschwert.

Die Bewertung nicht messbarer Kriterien kann nicht vollständig objektiv durchgeführt werden, da die Bewertung durch Faktoren, wie beispielsweise unterschiedliche Erfahrungen mit verschiedenen Konzepten der Frameworks, beeinflusst werden kann.

Die Festlegung der Projektanforderungen wird in der Praxis zum Zeitpunkt der Auswahl des Frameworks nicht immer eindeutig und rational möglich sein. Diese haben aber einen direkten Hebel bei der Auswertung des Entscheidungsmodells.

Zusätzliche Einflüsse, bezogen auf die Auswahl eines Frameworks, wie beispielsweise vorhandene Komponenten für ein bestimmtes Framework können mit dem Entscheidungsmodell nicht abgebildet werden. Diese könnten aber starken Einfluss auf die Eignung des Frameworks haben, da potentiell bereits vorhandene Komponenten für ein bestimmtes Framework viel Zeit bei der Projektumsetzung einsparen könnten.

Bei bestimmten Projekten könnte es zum Teil sinnvoll sein, mehrere Frameworks für einzelne Module der Webapplikation einzusetzen oder miteinander zu kombinieren. Dies kann durch das Entscheidungsmodell nur bedingt abgebildet werden. Eine mögliche Vorgehensweise wäre die Anwendung des Modells je getrenntem Modul der Applikation. Ein anderer Ansatz wäre die Berücksichtigung der Reihung der Frameworks der Auswertung bezogen auf die gesamte Webapplikation. Fehlende Funktionen des geeignetsten Frameworks könnten möglicherweise durch die Integration des zweitbesten ergänzt werden.

10.3 Fazit

Durch die empirische Untersuchung der Forschungsfrage stellte sich heraus, dass die Auswahl eines geeigneten Single Page Application Frameworks im Bereich Web-Frontendentwicklung durch ein systematisches Entscheidungsmodell unterstützt werden kann. Voraussetzung für eine hohe Praxisrelevanz sind dabei die Objektivität bei der Bewertung der Frameworks und der Bewertung der individuellen Projektanforderungen. Besonders gut funktioniert das Modell bei der Ausscheidung von Frameworks durch die Angabe von Muss-Kriterien, da diese messbar und dadurch objektiv sind. Aus der Umfrage geht hervor, dass die Mehrheit der Teilnehmer*innen ein Entscheidungsmodell als Grundlage für die Auswahl eines Frameworks verwenden würde. Auch dies kann mithilfe des entwickelten Entscheidungsmodells gut umgesetzt werden, da als Ergebnis des Modells nicht nur ein einzelnes Framework vorgeschlagen wird, sondern mehrere Frameworks hinsichtlich ihrer Eignung für die Projektanforderungen in absteigender Reihenfolge aufgelistet werden. Die Reihung erfolgt anhand des Nutzwertes, mit dem zusätzlich der Abstand der Eignung zwischen den Frameworks ersichtlich ist. Haben beispielsweise die ersten beiden Frameworks eine geringe Differenz im Vergleich zum dritten, so könnten diese beiden als Grundlage für eine finale Entscheidung verwendet werden.

11 AUSBLICK

Abschließend soll anhand der Ergebnisse ein Ausblick gegeben werden. Dies beinhaltet Handlungsempfehlungen sowohl für die Anwendung und Erweiterung des Entscheidungsmodells als auch für die Identifikation von Forschungsfragen, die sich aus den Ergebnissen ableiten lassen.

Im Rahmen dieser Masterarbeit wurde das Themengebiet in Kapitel 3 stark eingegrenzt. Als Handlungsempfehlung für die weitere Forschung bietet sich die Anwendung der Forschungsfrage auch auf andere Themenfelder an. Beispielsweise könnte eine Eingrenzung auf Frameworks für die Desktop-Frontendentwicklung erfolgen.

Um das entwickelte Entscheidungsmodell zu optimieren, sollte es um weitere Frameworks erweitert werden. Zusätzlich wäre es denkbar, die Kategorien zu erweitern. Die Methodik zur Anwendung des Modells bleibt dabei unabhängig von der Anzahl der Frameworks und der Anzahl der Kriterien gleich.

Die Parameter des Entscheidungsmodells sollten nicht, wie in dieser Masterarbeit, durch die Bewertung seitens einzelner Personen definiert werden. Stattdessen sollten die Parameter durch kontinuierliche Umfragen mit möglichst hohem Beteiligungsgrad festgelegt und laufend optimiert werden.

Zur Verringerung der Komplexität und somit zur Steigerung der Praxistauglichkeit könnte das Entscheidungsmodell über eine Webapplikation mit einfach aufgebauten Wizards umgesetzt werden. Dies hätte zusätzlich den Vorteil, dass durch die Zurverfügungstellung des Entscheidungsmodells gleichzeitig Statistiken über die am häufigsten benötigten Kriterien bei der Auswahl von Frameworks erfasst werden können.

ANHANG A - Antwortmöglichkeiten der Umfrage

Frage	Antwortmöglichkeit
1	Single-Choice
2	Single-Choice
3	Multiple-Choice
4	Single-Choice
5	Skala
6	Single-Choice
7	Freitext
8	Multiple-Choice
9	Single-Choice
10	Freitext
11	Single-Choice
12	Freitext
13	Multiple-Choice
14	Skala
15	Skala
16	Skala
17	Skala
18	Skala
19	Skala
20	Skala
21	Skala
22	Skala
23	Freitext

Tabelle 14: Ausgewählte Antwortmöglichkeiten (Eigene Darstellung)

Frage	Antwortmöglichkeiten
1	• Softwareentwicklung • Softwarearchitekt • Projektleitung • Produktmanager • Consulting • Entwicklungsleitung • CTO • CEO • Sonstiges
2	• Weniger als 2 Jahre • 2 bis 5 Jahre • 5 bis 10 Jahre • Mehr als 10 Jahre
3	• Angular • Vue.js • React • Blazor • Keine der Angeführten • *Eigene Antwort
4	• Ja • Nein
6	• Ja, immer • Ja, bei wichtigen Projekten • Ja, als Entscheidungsgrundlage • Nein
8	• Programmiersprache (JS, TS, C#, Java, Phyton) • Beliebtheit (Framework wird häufig verwendet) • Starke Community

	• Einfach zu lernen • Performance • Gute Testbarkeit • Gute Dokumentation des Frameworks vorhanden • Wenig Installationsaufwand • Entwicklungsgeschwindigkeit (Unterstützung durch CLI, Template, Codegenerator) • Funktionsumfang (viele Features: Authorization, Multilanguage…) • Erweiterbarkeit/Plug-Ins • Browserkompatibilität (viele Browser, alte Browserversionen, IE) • Support (einfach zugänglich, kostenlos) • Langfristige Beständigkeit am Markt • Personal mit Knowhow am Arbeitsmarkt verfügbar
9	• Angular • React • Vue.js • Blazor • Keines
11	• Ja • Nein
13	• Knowhow war vorhanden • Hohe Marktpopulation • Große Community • Viele Features • Hohe Entwicklungsgeschwindigkeit • Programmiersprache

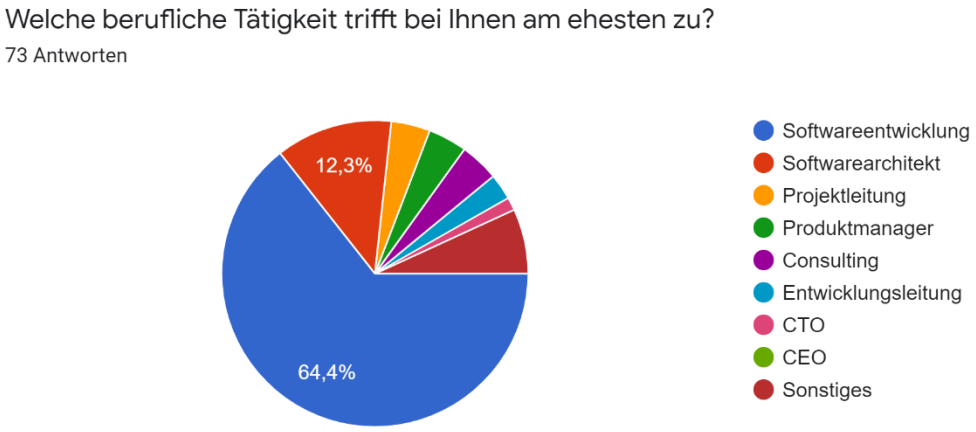
	• Schneller Einstieg ins Framework • Ausreichend Tutorials vorhanden • Hohe Performanz • Gute Testbarkeit • Gute Dokumentation • Ausreichend Support • Mitarbeiter für Arbeitsmarkt vorhanden • *Eigene Antwort

Tabelle 15: Vorgaben bei Single-Choice und Multiple-Choice Fragen (Eigene Darstellung)

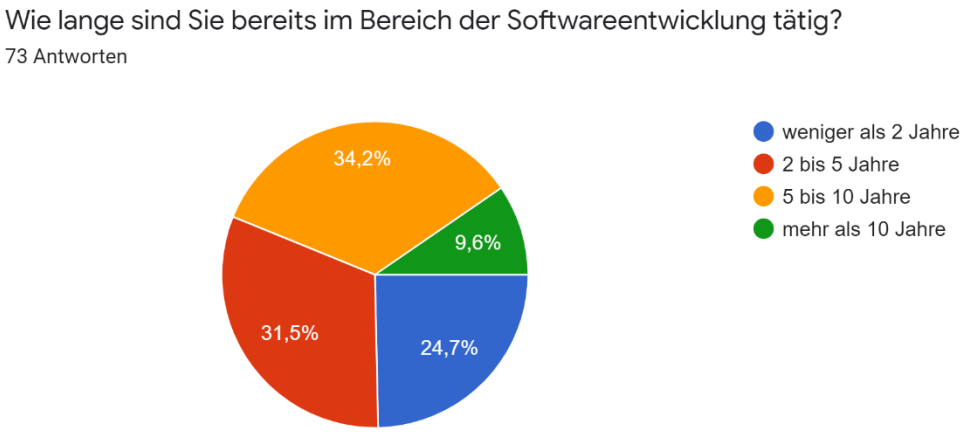
ANHANG B - Umfrageergebnisse

In diesem Anhang werden die erfassten Ergebnisse der Umfrage dargestellt. Die dazugehörigen Abbildungen sind eine eigene Darstellung und verfügen daher über keine explizite Quellenangabe.

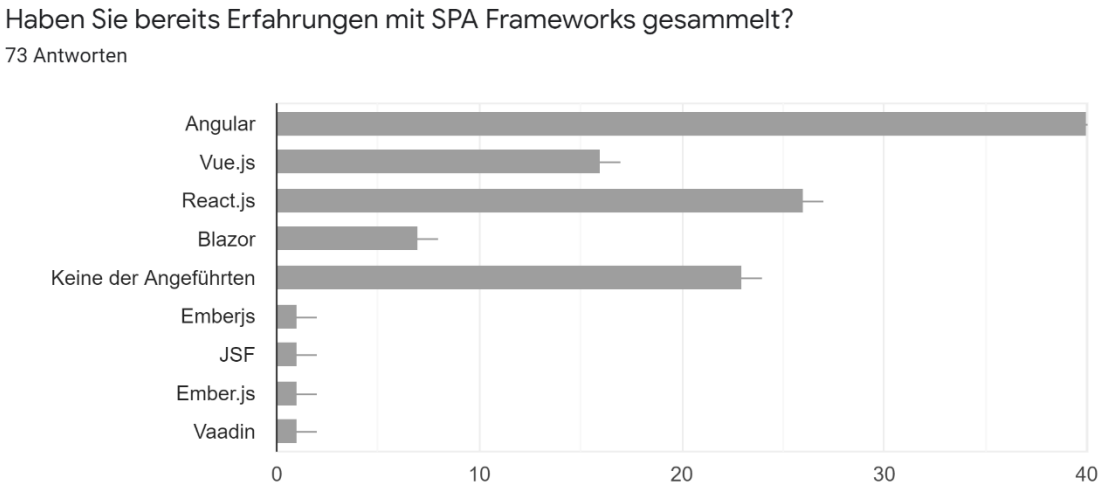
Frage 1:



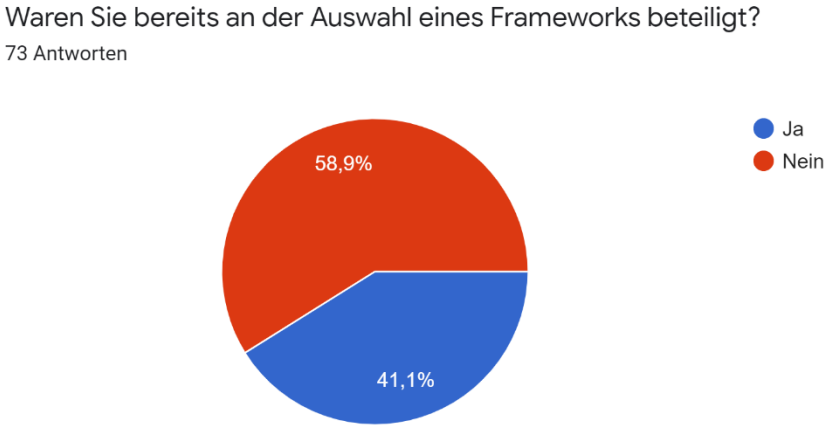
Frage 2:



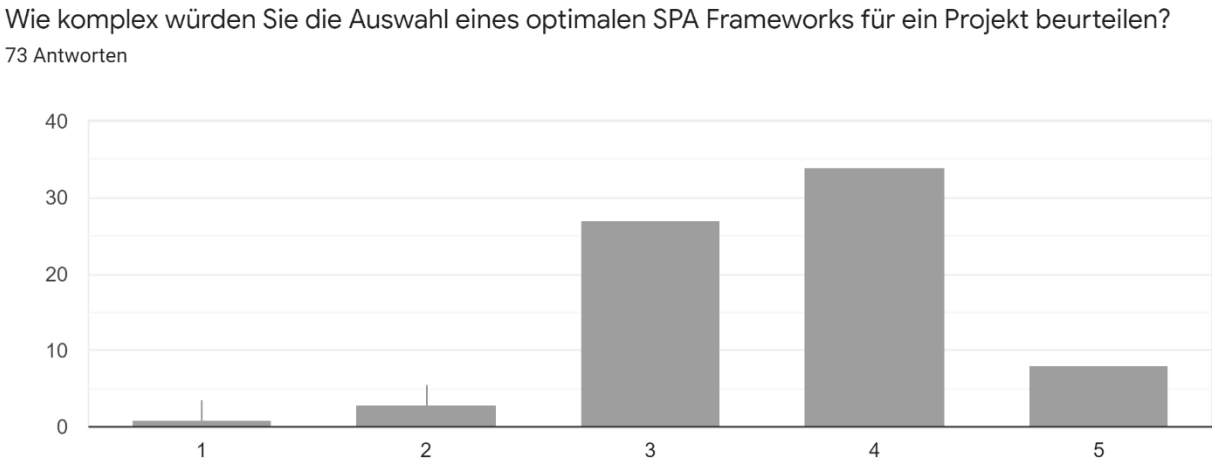
Frage 3:



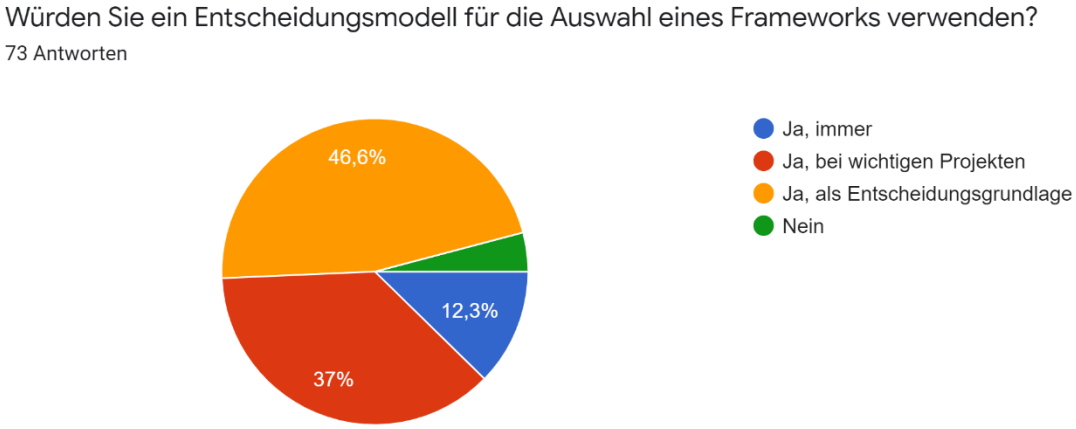
Frage 4:



Frage 5:



Frage 6:

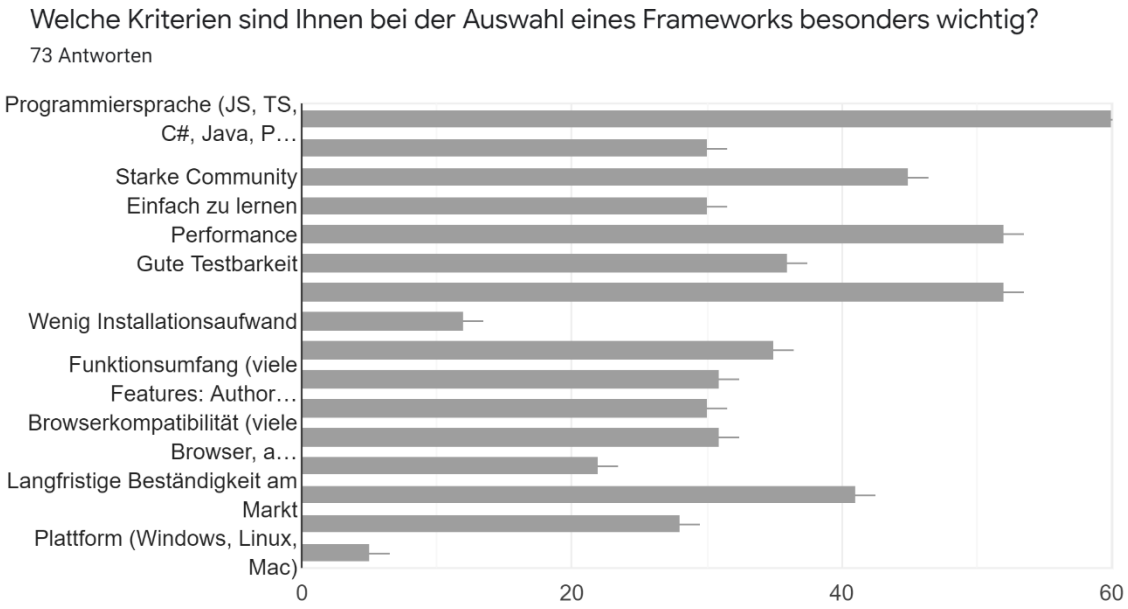


Frage 7:

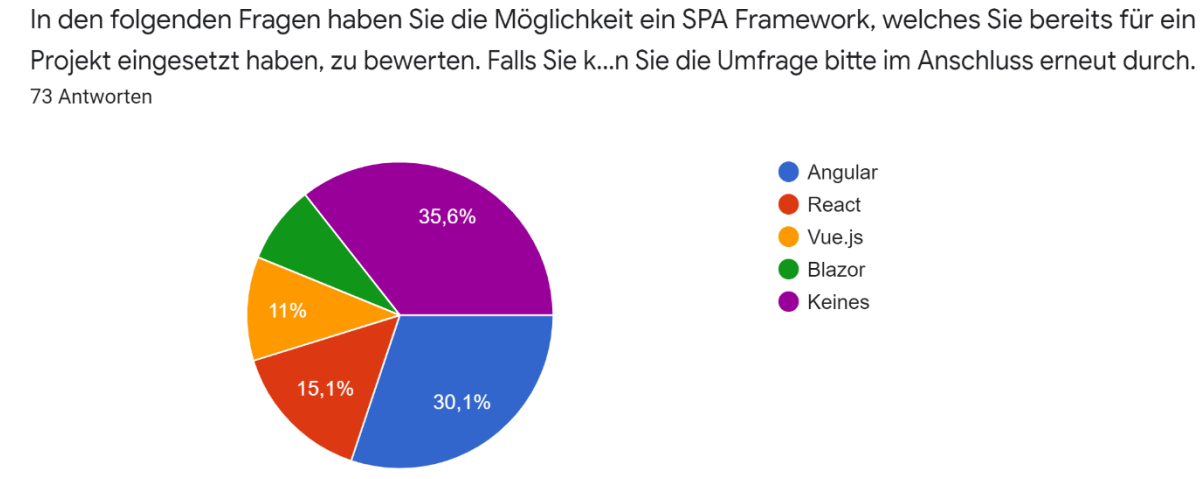
Bitte begründen Sie kurz, weshalb Sie kein Entscheidungsmodell verwenden würden.:

- Zu geringe „objektive“ Unterschiede zwischen den Frameworks
- Keine Kenntnisse

Frage 8:



Frage 9:



Um die Fragennummern mit dem Fragenkatalog in Kapitel 9.2.1 konsistent zu halten sind nachfolgende Fragen mit dem jeweiligen Framework gekennzeichnet.

Frage 10 (Angular):

Was wurde mit dem Framework umgesetzt? (z.B. Webshop, Dashboard für Maschinenübersicht etc.)

- Business Support System
- Social Media Platform für spezielle Zielgruppen
- Vollständige Web Applikation (WMS)
- Single Page Application
- Web Visualisierungs für Automatisierungslösungen
- interne s7chemaschine
- Dashboard für Lagerverwaltung
- Dashboard
- Visualisierung von einem Automatiklager
- Computerized maintenance management system
- MES System
- Business-Lösung für Überweisungen
- Enterprise Applikation
- Visualisierung einer Automatisierungssoftware
- Dashboard für Personalplanung
- Webapplikation
- Einfache Übersicht von Reports
- Smart Home Visualisierung
- Dashboard für Smart Home
- Webshop

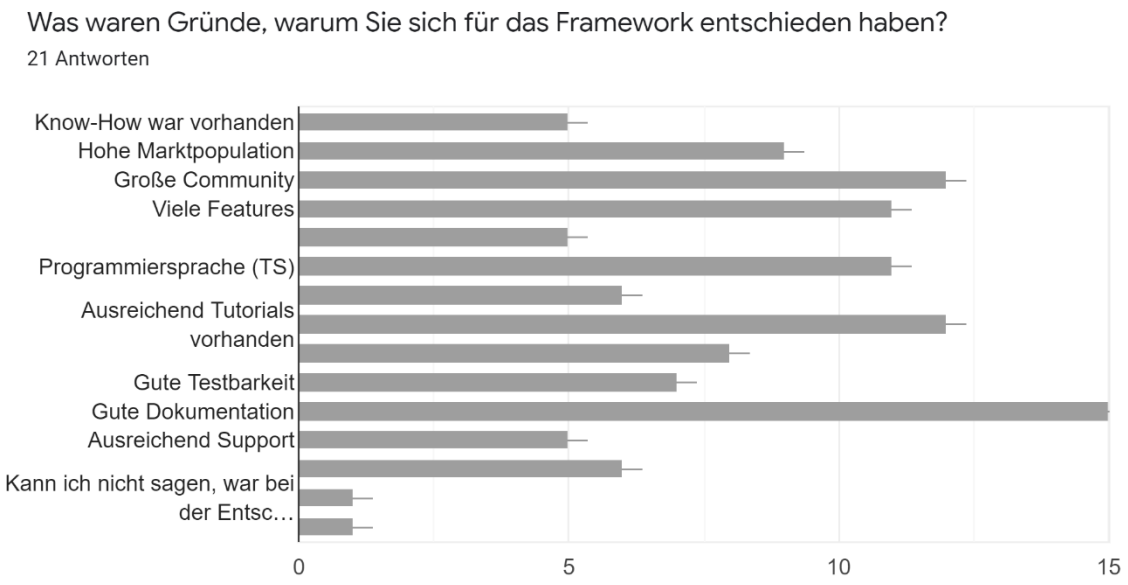
Frage 11 (Angular):

Frage 12 (Angular):

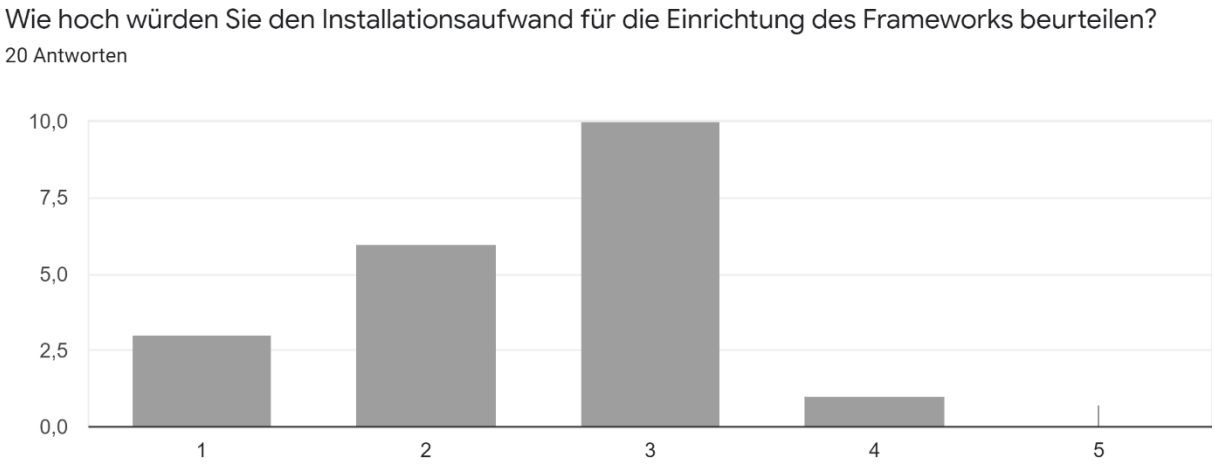
Bitte begründen Sie kurz ihre Auswahl.

- Wiederverwendbarkeit von Komponenten und einfache Erweiterbarkeit für die Integration von Standardprodukten
- Durch den großen Know-How-Gain wurden auch andere Projektofferten möglich
- Großer Umfang, community und stabiles framework
- es hat sich als Framework noch weiter etabliert
- rapit prototyping war dadurch möglich
- Für den Entwicklungszeitraum den wir hatten war die Lernkurve zu hoch.
- Viele Features, einfach zur bedienen
- Features aus einer Hand, Leichte Skalierung,
- Nutzbarkeit der Fachabteilung war effizient gegeben und konnte angepasst werden.
- Gute Quellen zur Umsetzung vorhanden, gute Webseite & Community
- Es war einfach zu verwenden und ist sehr gut dokumentiert

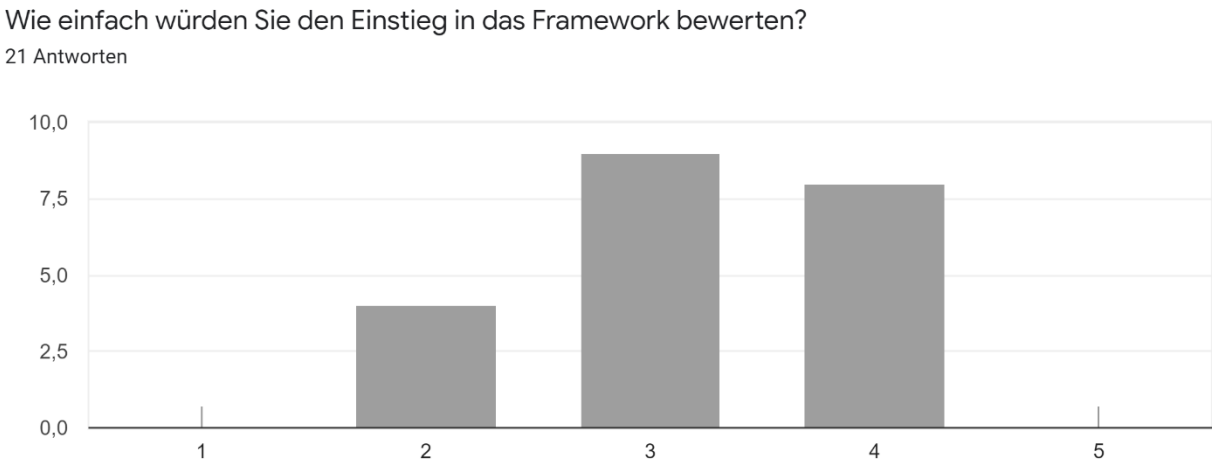
Frage 13 (Angular):



Frage 14 (Angular):



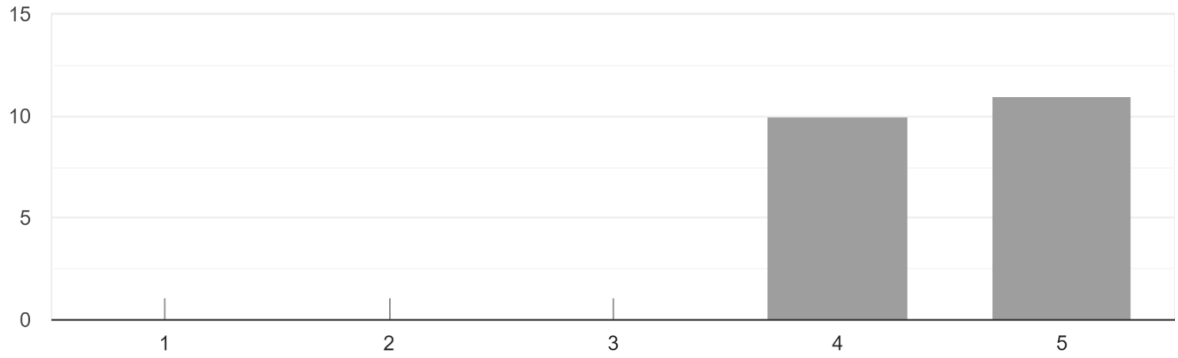
Frage 15 (Angular):



Frage 16 (Angular):

Wie gut wurden Unterstützungen wie Tutorials, Beispiele und Schulungen vom Framework angeboten?

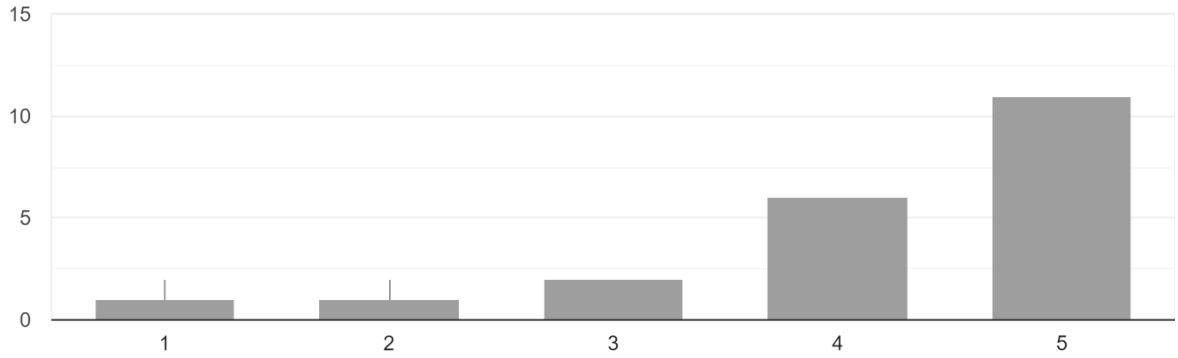
21 Antworten



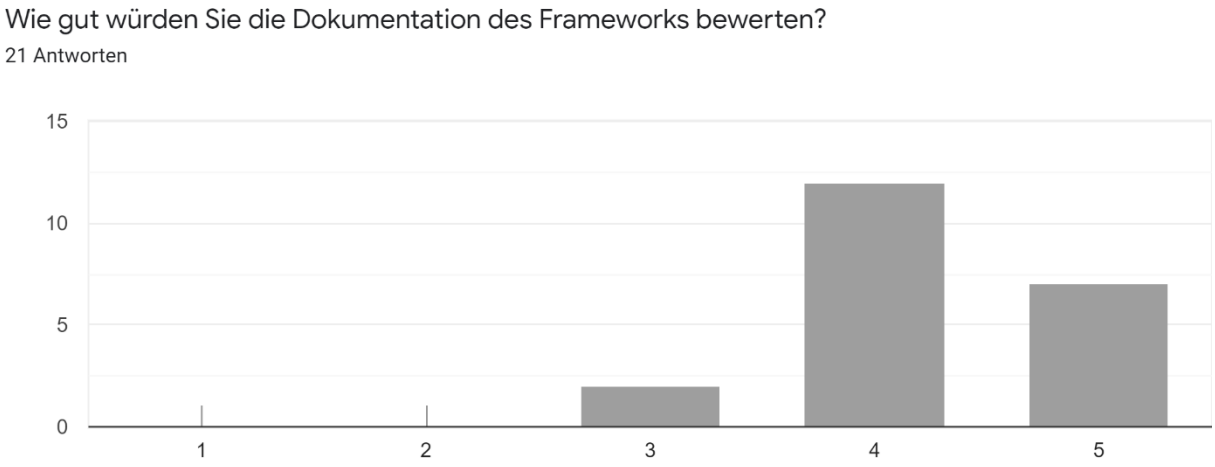
Frage 17 (Angular):

Wie würden Sie die Zeitersparnis bei der Projektumsetzung durch das Framework im Vergleich zur Entwicklung ohne Framework bewerten?

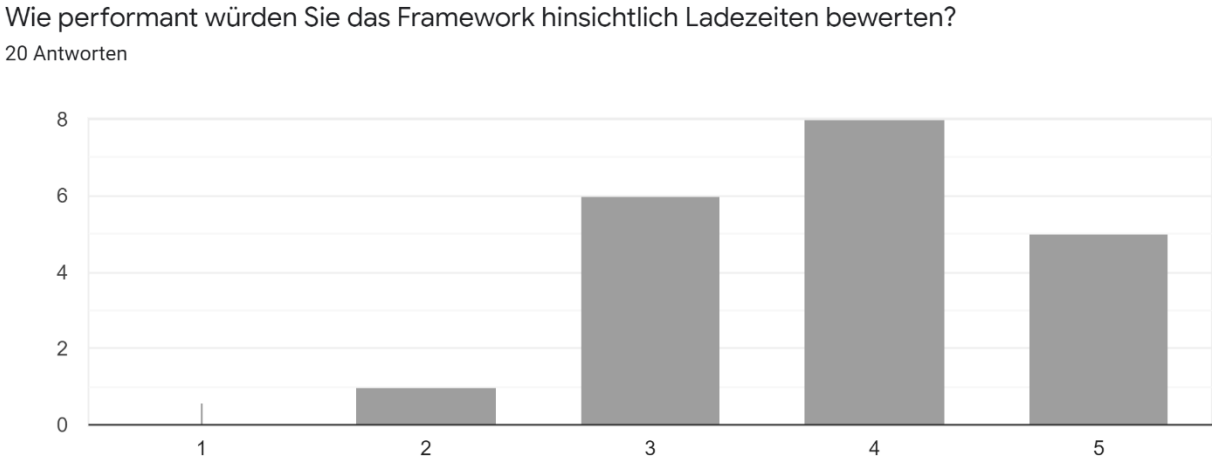
21 Antworten



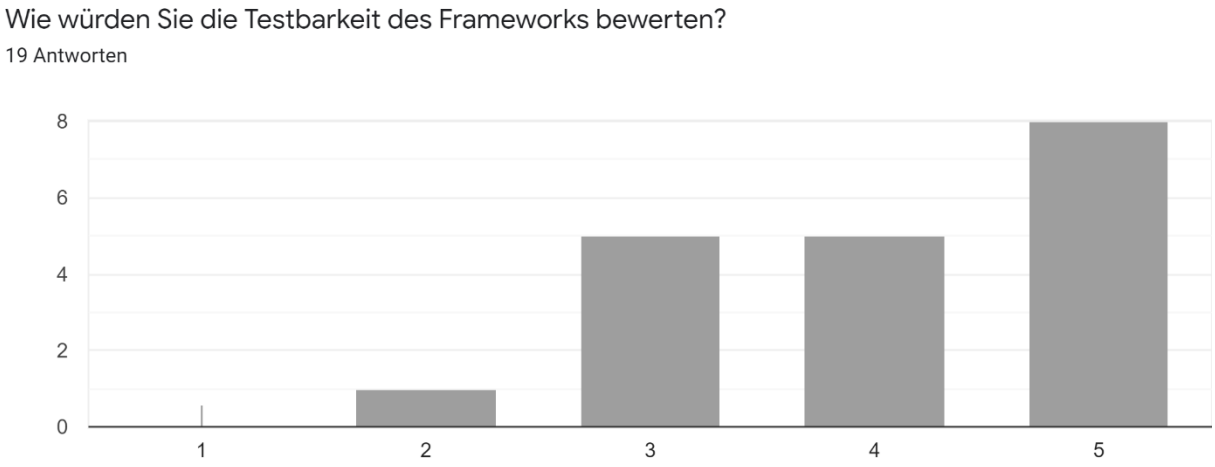
Frage 18 (Angular):



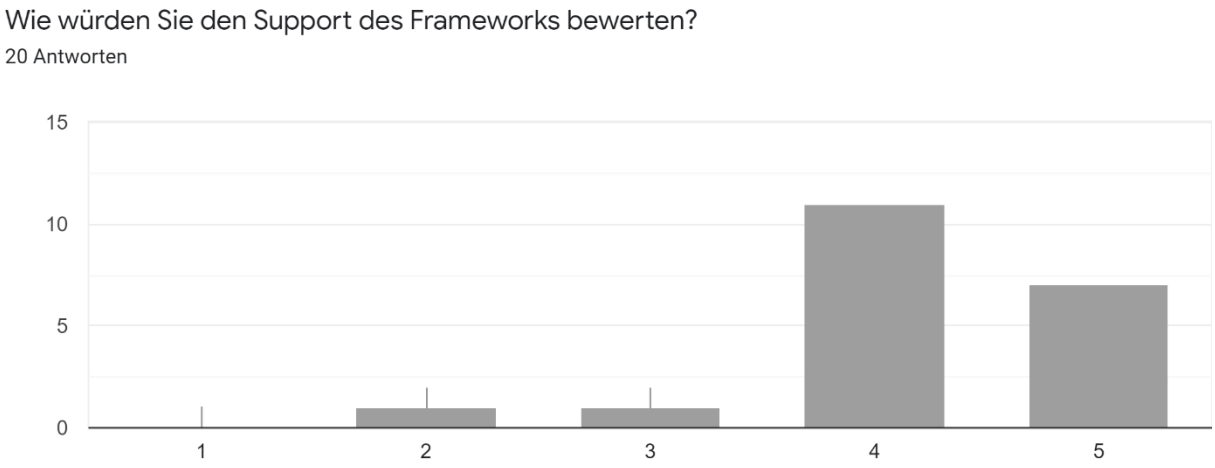
Frage 19 (Angular):



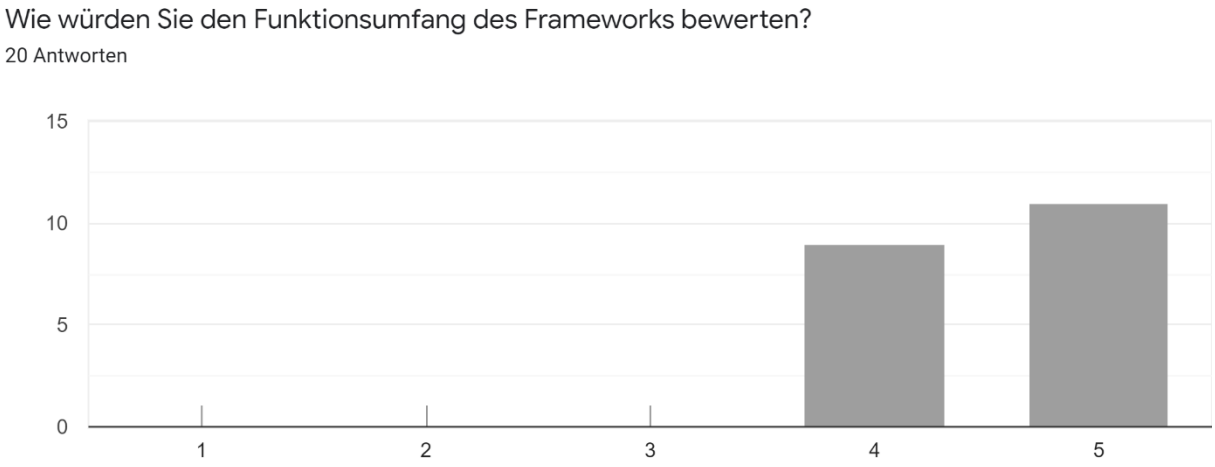
Frage 20 (Angular):



Frage 21 (Angular):



Frage 22 (Angular):

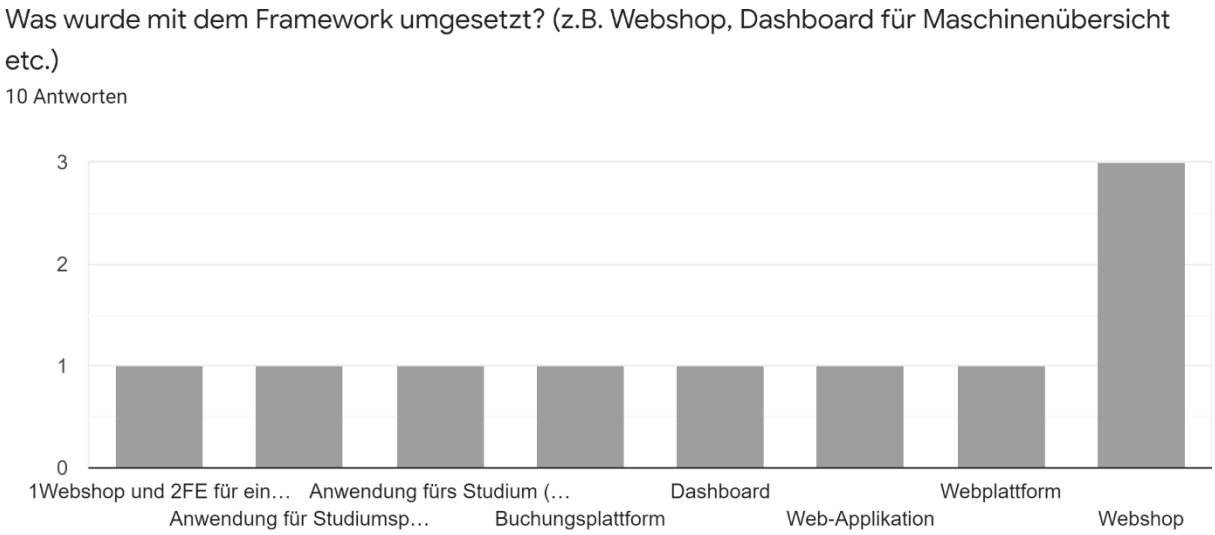


Frage 23 (Angular):

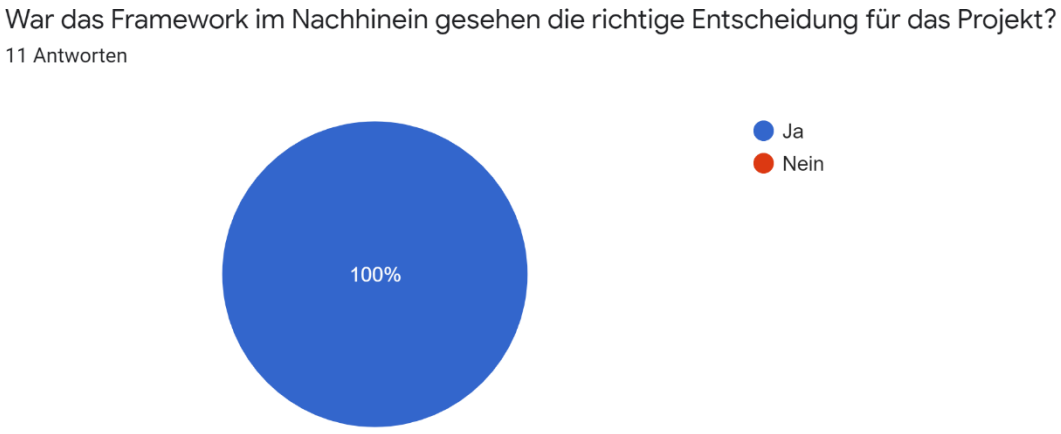
Haben Sie noch zusätzliche Anmerkungen?

- Antworten hängen stark von der Ausprägung der Applikation wie beispielsweise bei Integrationsmöglichkeiten einer Standardsoftware mit Multi-Tenancy
- Jede Major-Version hat sehr viele Breaking-Changes.

Frage 10 (React):



Frage 11 (React):

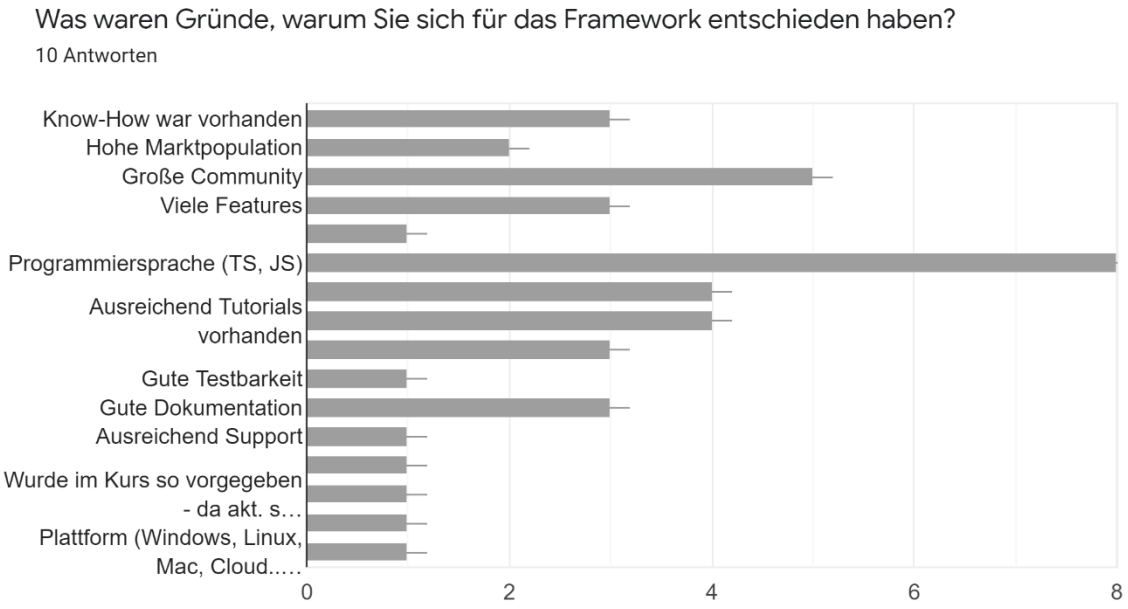


Frage 12 (React):

Bitte begründen Sie kurz Ihre Auswahl.

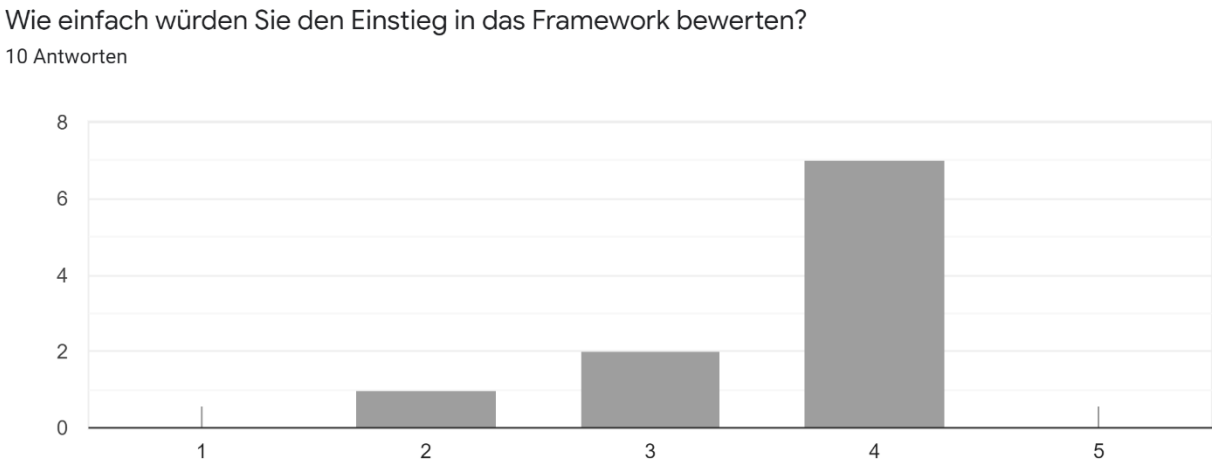
- War damit gut umsetzbar, gute Community bei Fragen
- Performance
- Aufgrund der komponentenbasierten Architektur konnte die Komplexität gut strukturiert werden.
- Da es viele Freiheiten bietet und man sich die Dinge selbst zusammenstellen kann, wie man es benötigt. Auch war es damals für die Schrittweise Migration einfacher.
- Strukturierter Aufbau möglich, Hohe Flexibilität
- Flexibel und einfach zu erlernen
- für 2.: Vorerfahrung durch 1 vorhanden und dadurch noch schnellere Umsetzung des schon vorab relativ simplen Setups.
- Ist gut erweiterbar

Frage 13 (React):



Frage 14 (React):

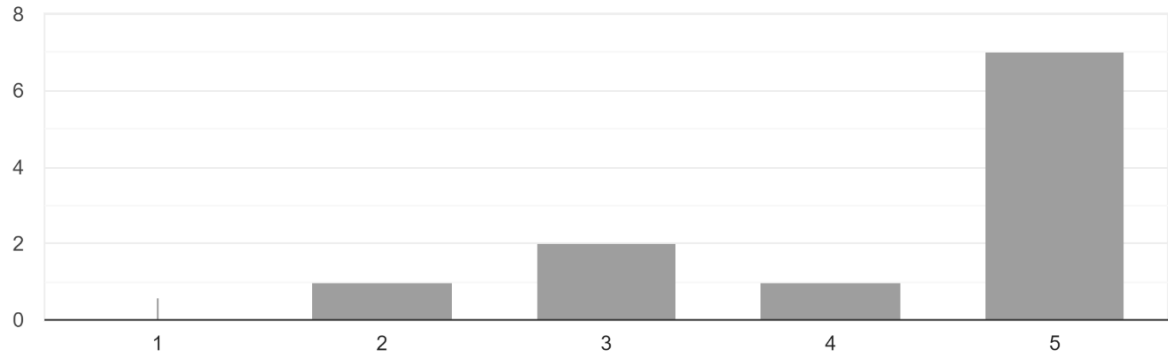
Frage 15 (React):



Frage 16 (React):

Wie gut wurden Unterstützungen wie Tutorials, Beispiele und Schulungen vom Framework angeboten?

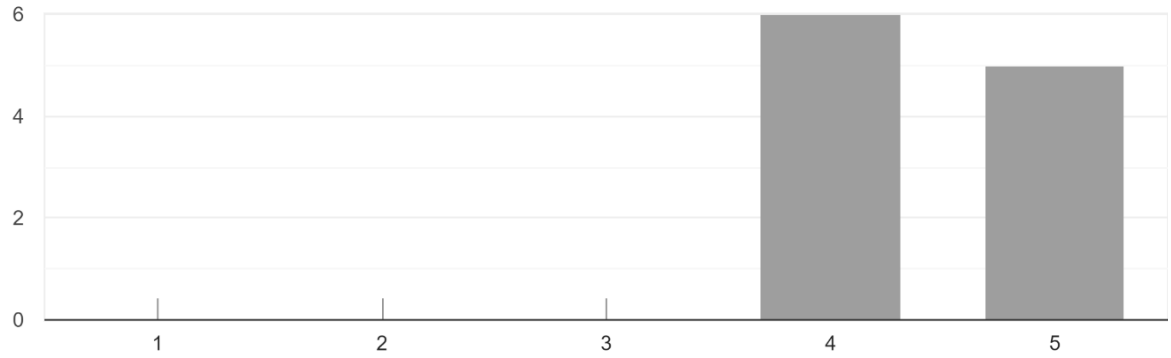
11 Antworten



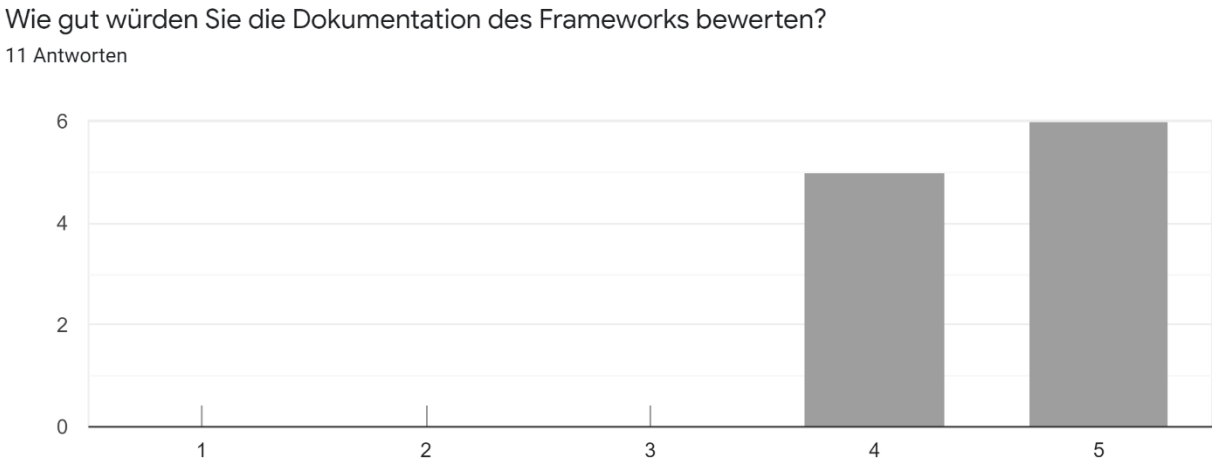
Frage 17 (React):

Wie würden Sie die Zeitersparnis bei der Projektumsetzung durch das Framework im Vergleich zur Entwicklung ohne Framework bewerten?

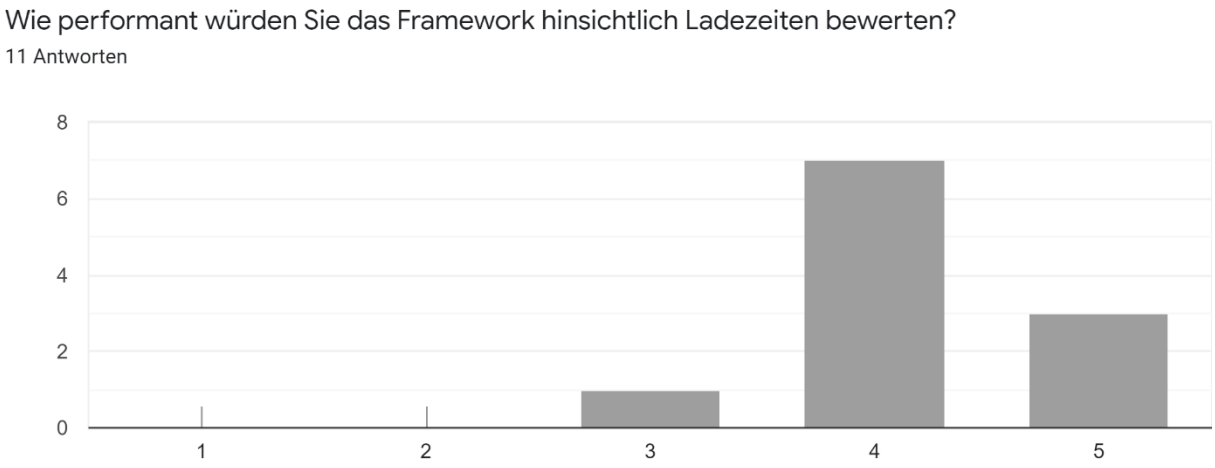
11 Antworten



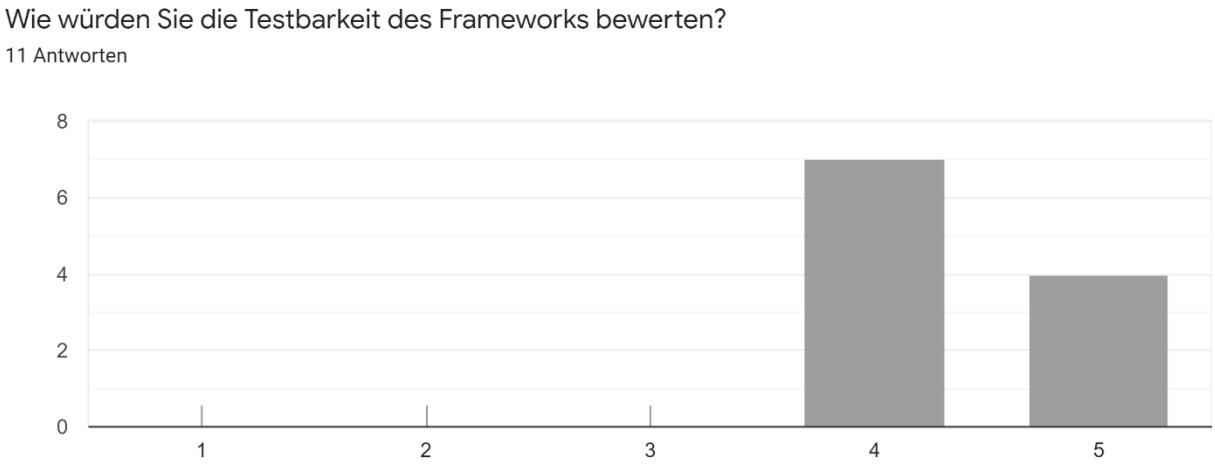
Frage 18 (React):



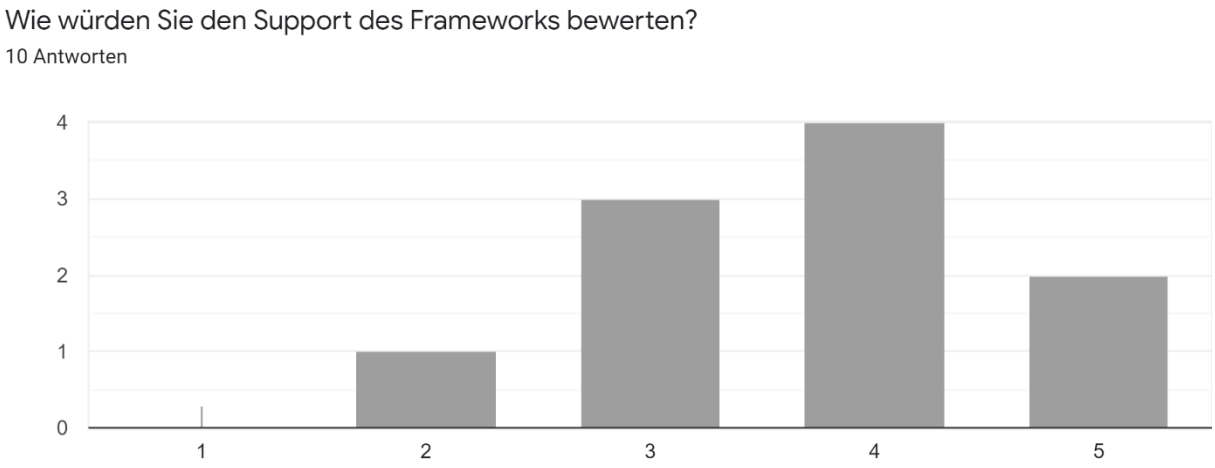
Frage 19 (React):



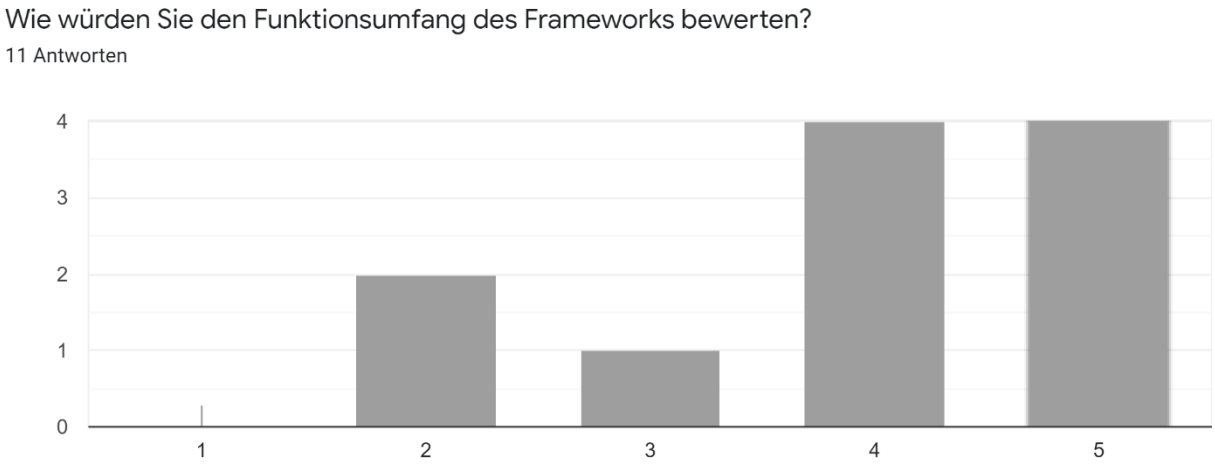
Frage 20 (React):



Frage 21 (React):



Frage 22 (React):



Frage 23 (React):

Haben Sie noch zusätzliche Anmerkungen?

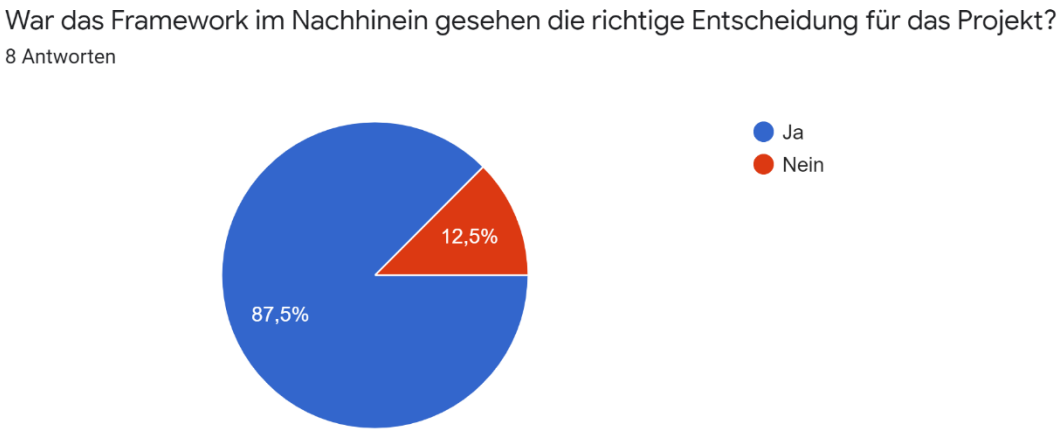
- -
- React ist eine Library :-)

Frage 10 (Vue.js):

Was wurde mit dem Framework umgesetzt? (z.B. Webshop, Dashboard für Maschinenübersicht etc.)

- Mehrere Firmen interne Apps
- Komplexe Herstellungs-Prozess-Überwachung
- Modellierungswerkzeug
- Webshop
- Order Man für kleine Feste

Frage 11 (Vue.js):

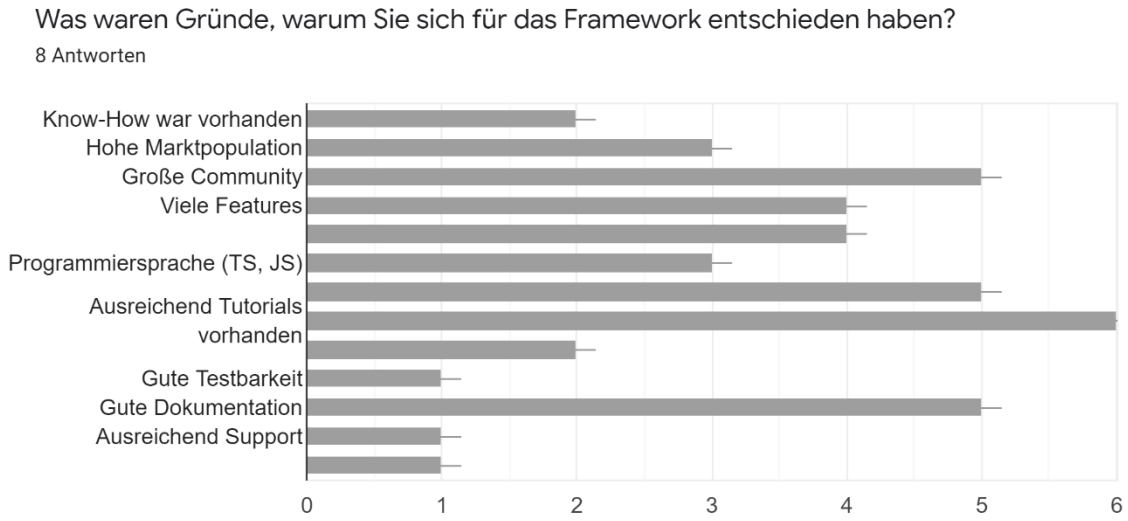


Frage 12 (Vue.js):

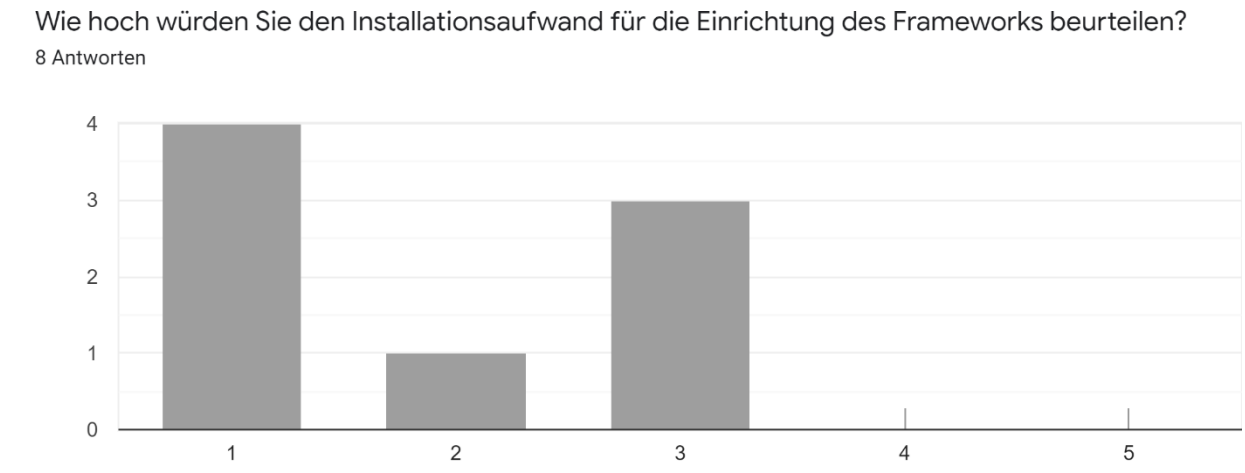
Bitte begründen Sie kurz Ihre Auswahl.

- Schnell erlernbar und schnelle Ergebnisse
- Es hat funktioniert.
- .
- Einfach und schnell gute Ergebnisse

Frage 13 (Vue.js):



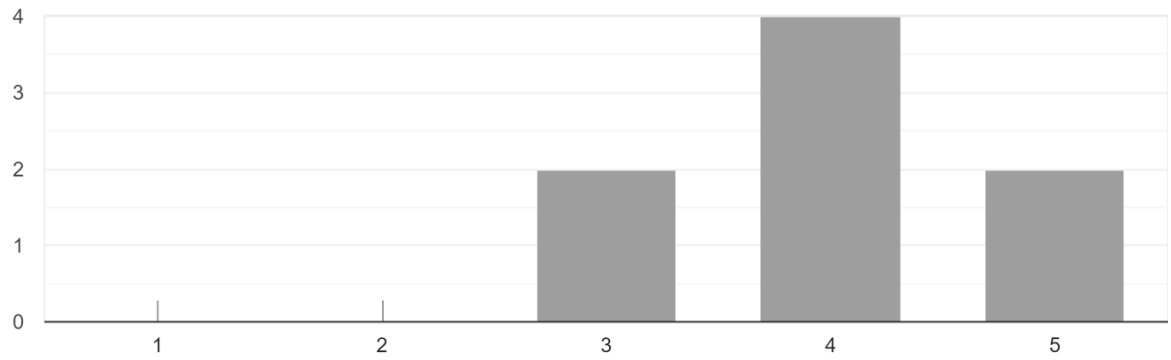
Frage 14 (Vue.js):



Frage 15 (Vue.js):

Wie einfach würden Sie den Einstieg in das Framework bewerten?

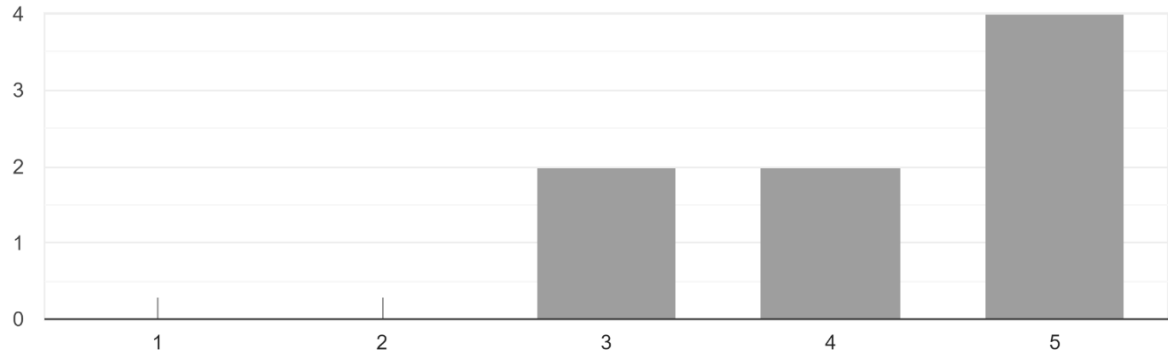
8 Antworten



Frage 15 (Vue.js):

Wie gut wurden Unterstützungen wie Tutorials, Beispiele und Schulungen vom Framework angeboten?

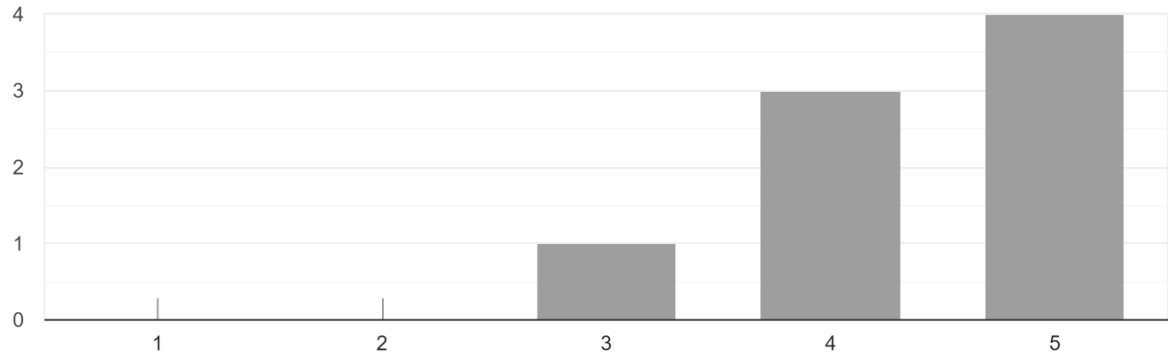
8 Antworten



Frage 16 (Vue.js):

Wie würden Sie die Zeitersparnis bei der Projektumsetzung durch das Framework im Vergleich zur Entwicklung ohne Framework bewerten?

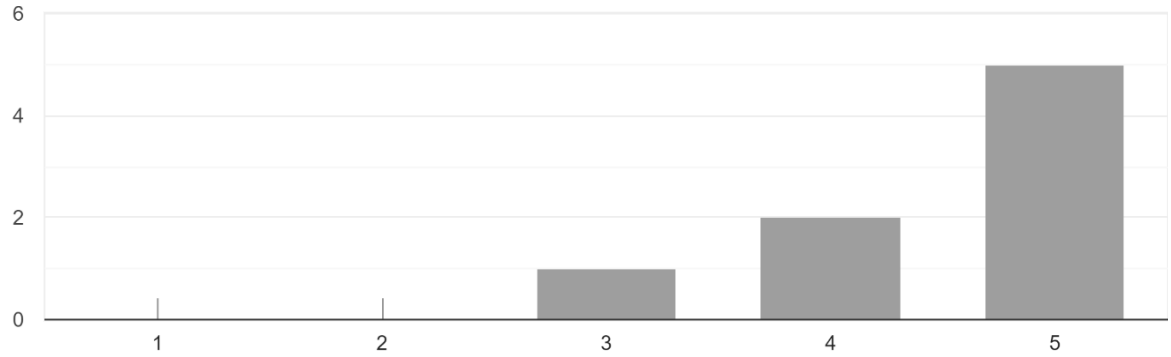
8 Antworten



Frage 17 (Vue.js):

Wie gut würden Sie die Dokumentation des Frameworks bewerten?

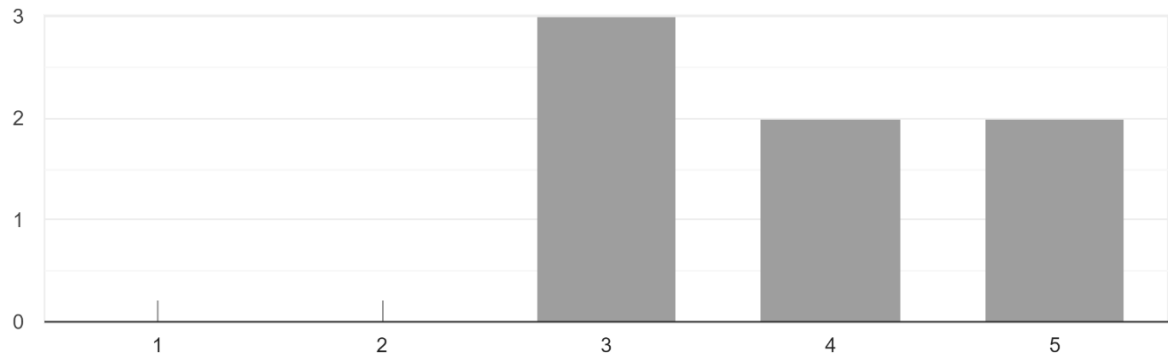
8 Antworten



Frage 18 (Vue.js):

Wie performant würden Sie das Framework hinsichtlich Ladezeiten bewerten?

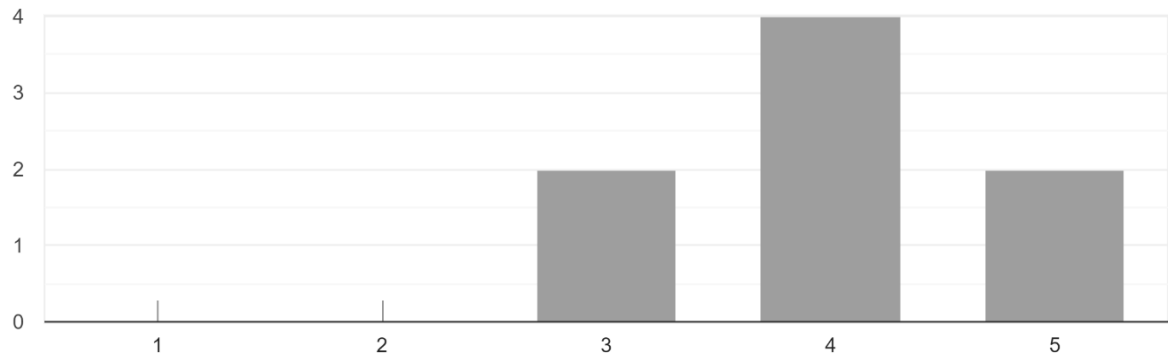
7 Antworten



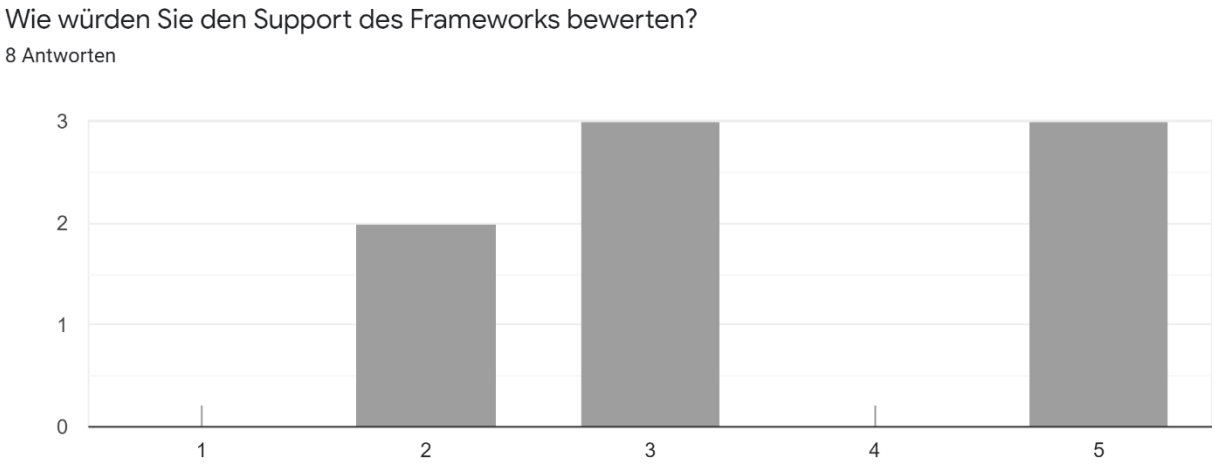
Frage 19 (Vue.js):

Wie würden Sie die Testbarkeit des Frameworks bewerten?

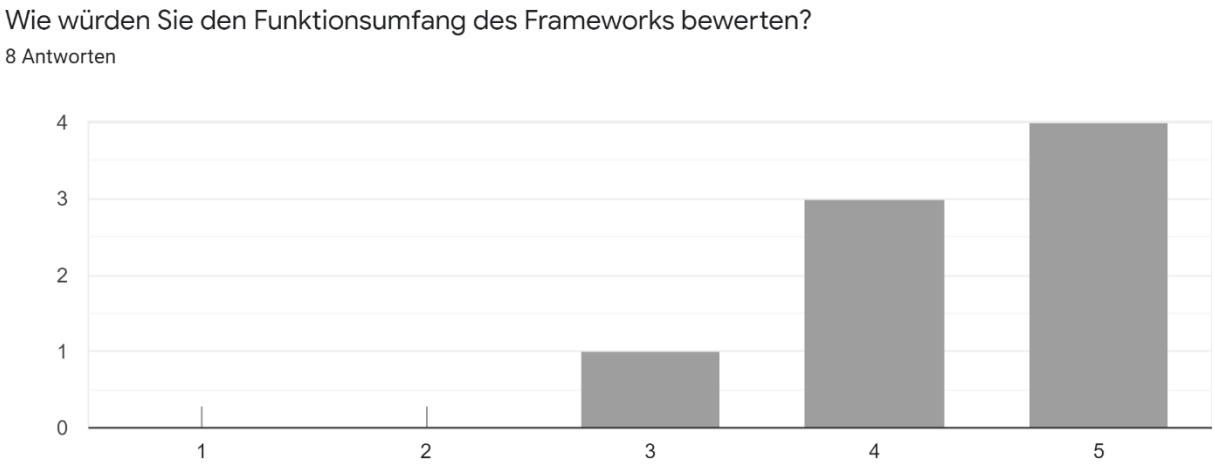
8 Antworten



Frage 20 (Vue.js):



Frage 21 (Vue.js):

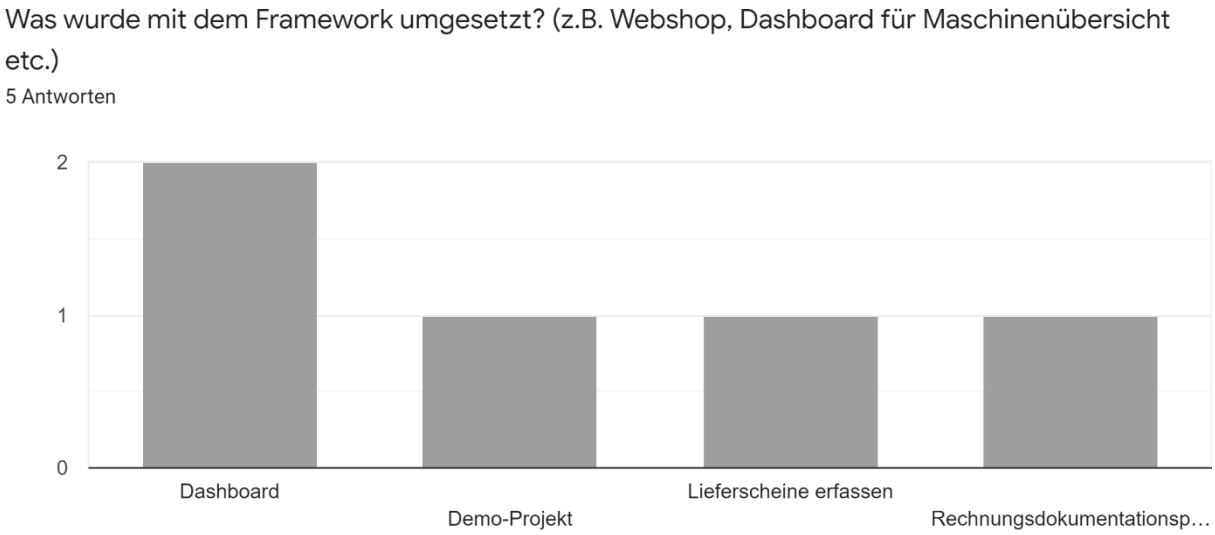


Frage 10 (Vue.js):

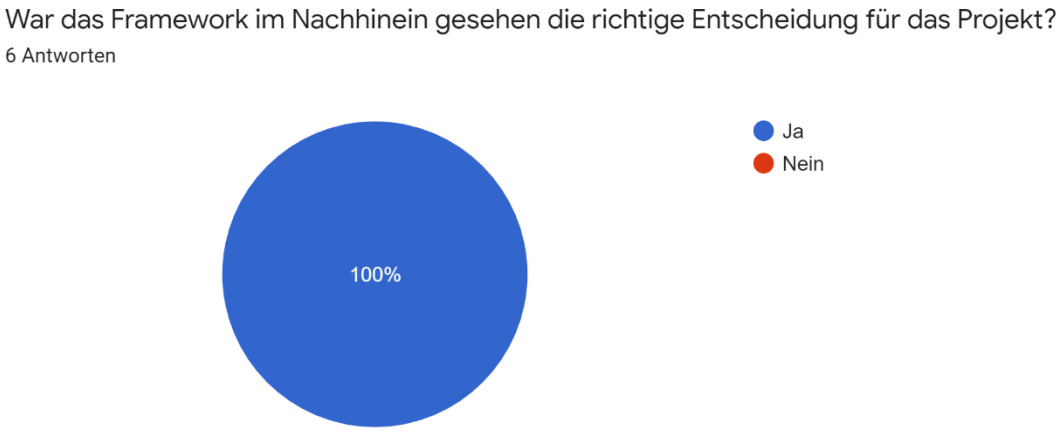
Haben Sie noch zusätzliche Anmerkungen?

Keine Antworten

Frage 10 (Blazor):



Frage 11 (Blazor):

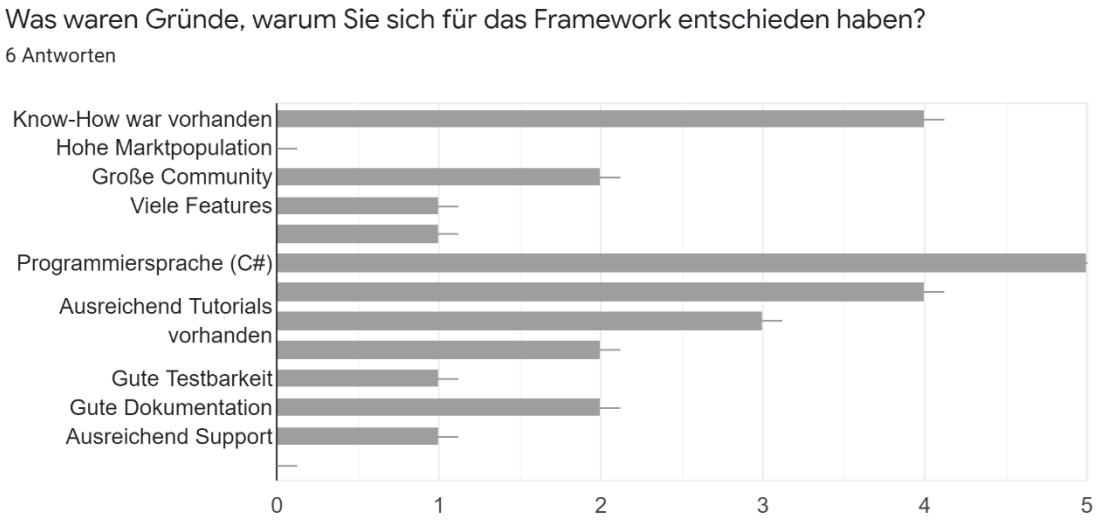


Frage 12 (Blazor):

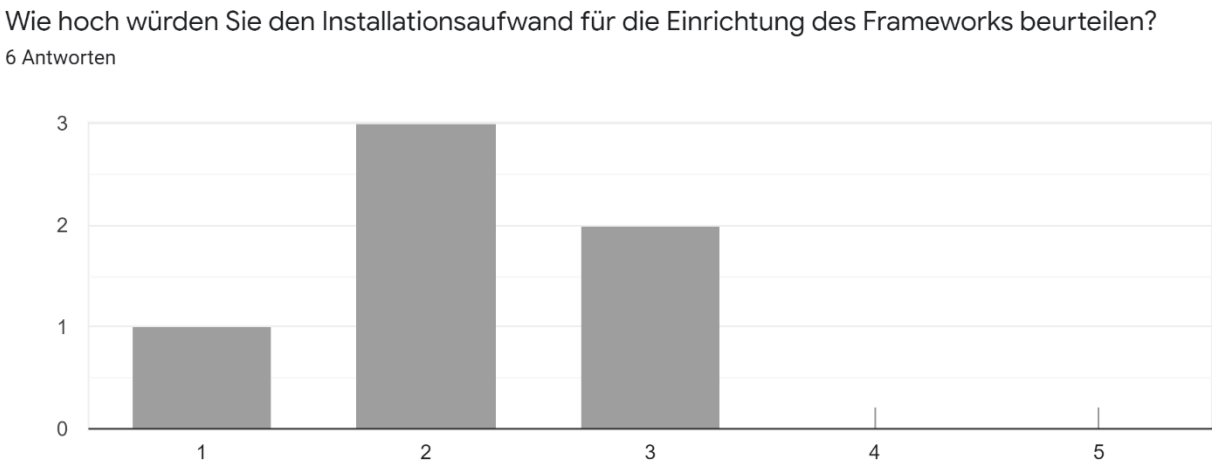
Bitte begründen Sie kurz Ihre Auswahl.

- Erfüllte den Zweck des Projekts vollumfänglich
- Wurde verwendet, weil das Produkt auf Azure laufen soll und Azure Functions verwendet. Dadurch war es einfach die Datenmodelle von Back und Frontend zu teilen.
- Open Source und im gewohnten .NET Umfeld

Frage 13 (Blazor):



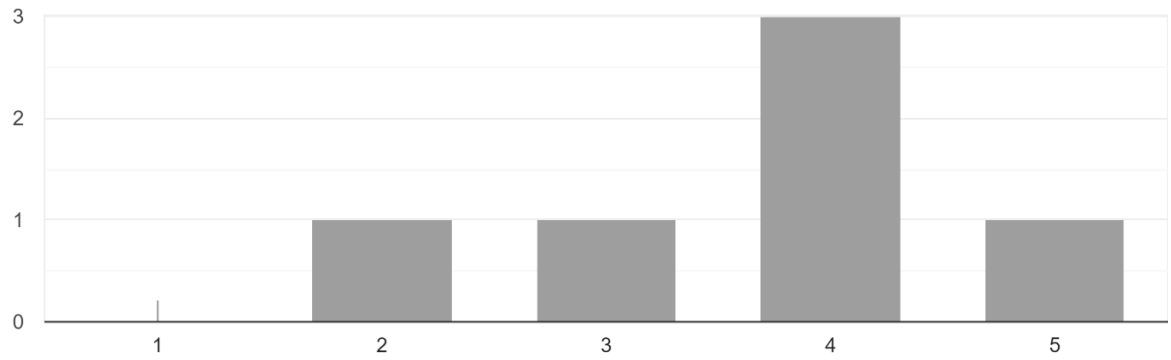
Frage 14 (Blazor):



Frage 15 (Blazor):

Wie einfach würden Sie den Einstieg in das Framework bewerten?

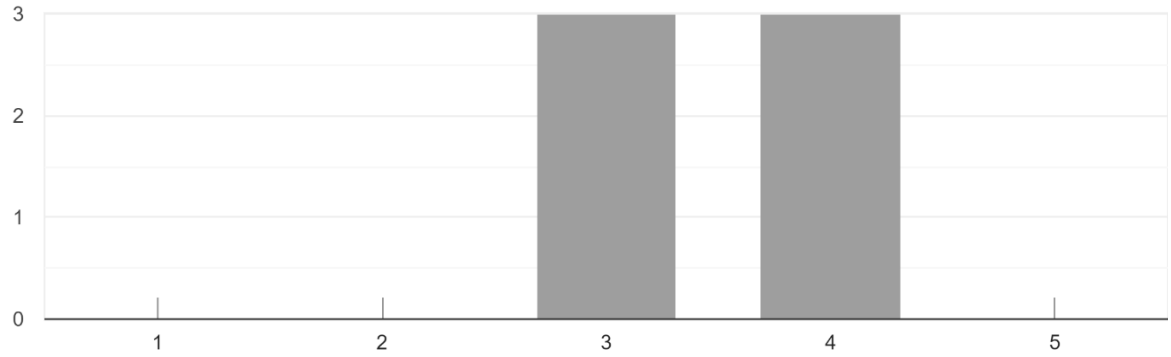
6 Antworten



Frage 16 (Blazor):

Wie gut wurden Unterstützungen wie Tutorials, Beispiele und Schulungen vom Framework angeboten?

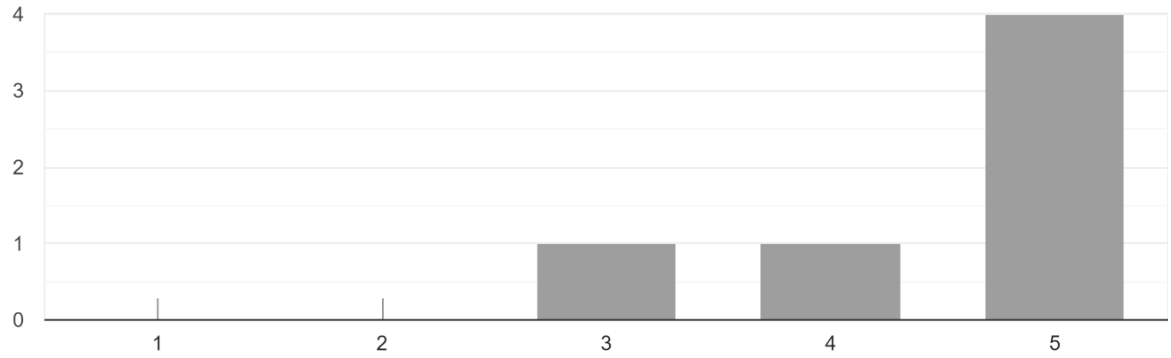
6 Antworten



Frage 17 (Blazor):

Wie würden Sie die Zeitersparnis bei der Projektumsetzung durch das Framework im Vergleich zur Entwicklung ohne Framework bewerten?

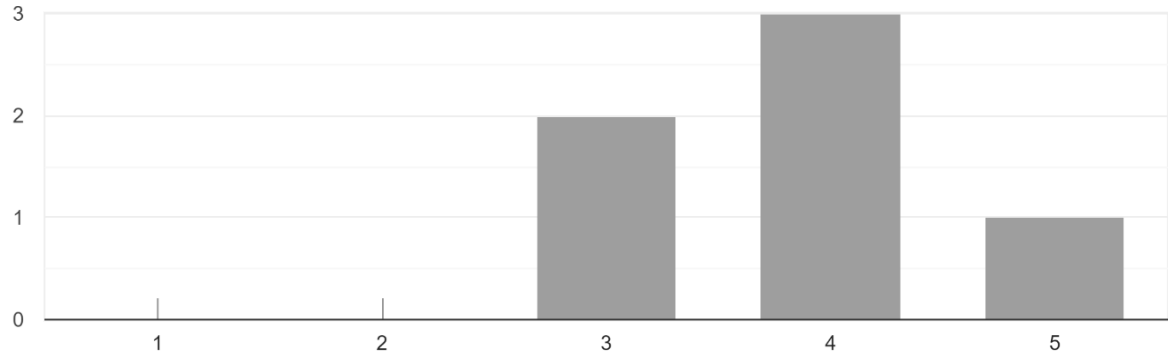
6 Antworten



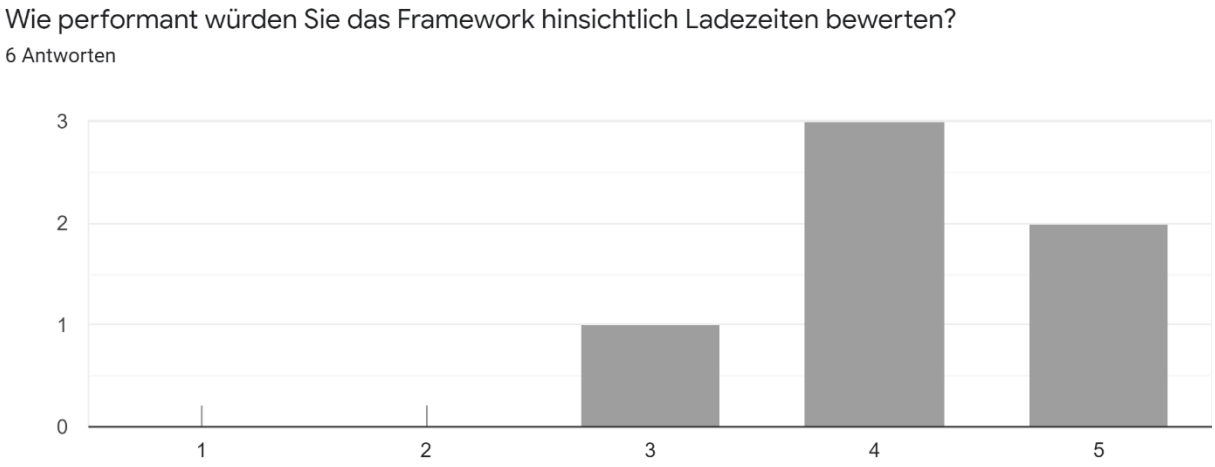
Frage 18 (Blazor):

Wie gut würden Sie die Dokumentation des Frameworks bewerten?

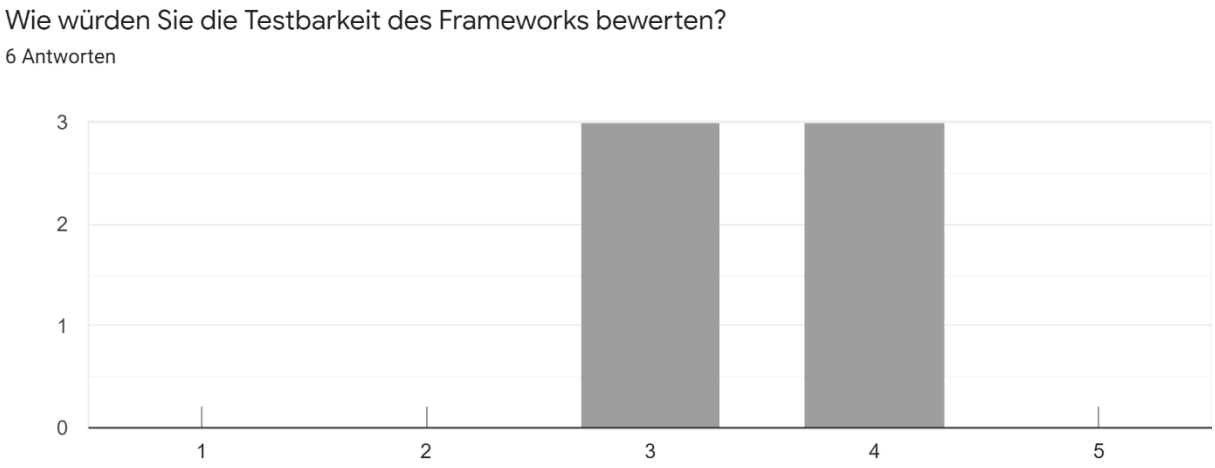
6 Antworten



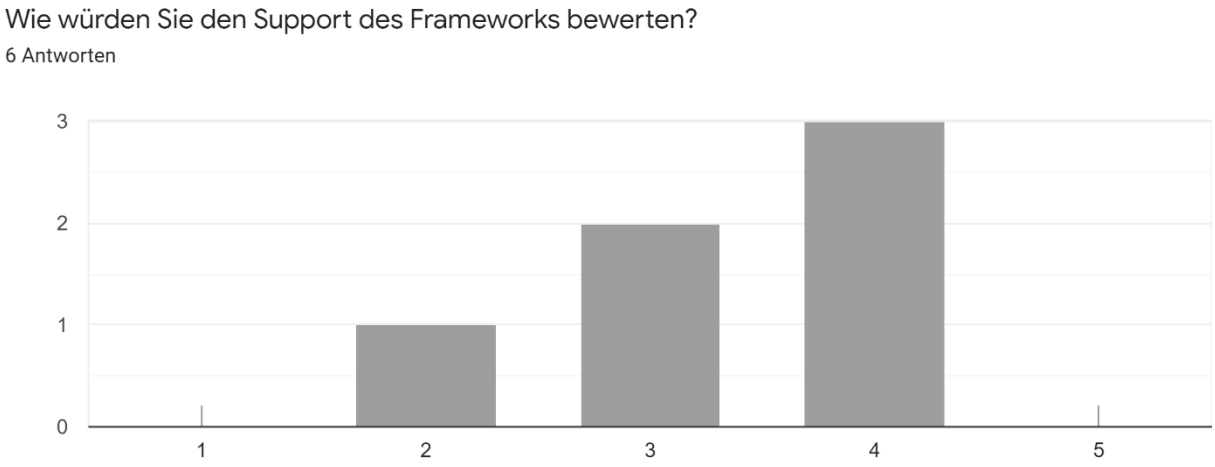
Frage 19 (Blazor):



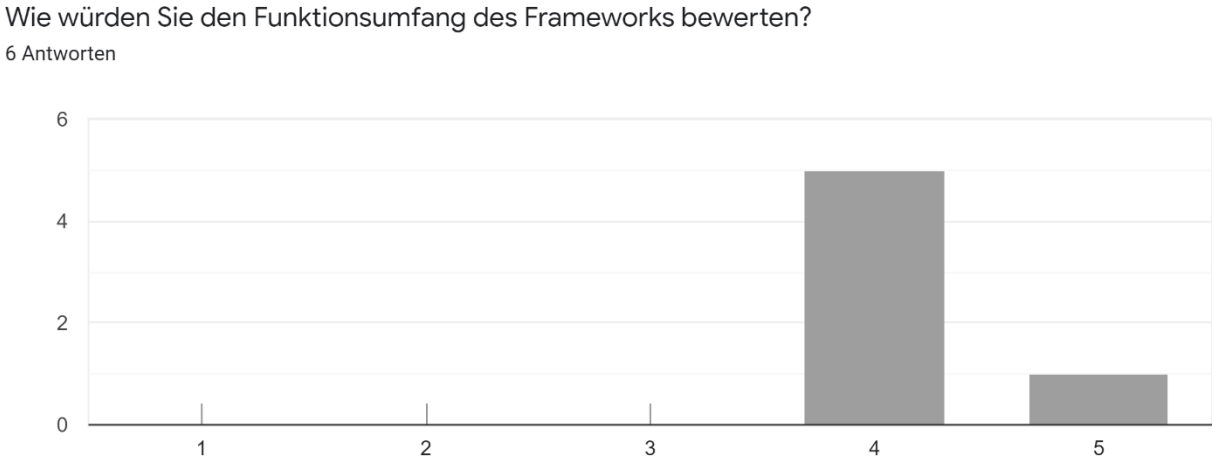
Frage 20 (Blazor):



Frage 21 (Blazor):



Frage 22 (Blazor):



Frage 23 (Blazor):

Haben Sie noch zusätzliche Anmerkungen?

- Ideal, wenn man aus dem .NET Umfeld kommt

ABKÜRZUNGSVERZEICHNIS

CLI ..... *Command-Line-Interface*
CLR..... *Common-Language-Runtime*
CSS ..... *Cascading-Style-Sheets*
DNS ..... *Domain-Name-Service*
DOM ..... *Document-Object-Model*
HTML ..... *Hypertext-Markup-Language*
HTTP ..... *Hypertext-Transfer-Protocol*
IDE..... *Integrated Development Environment*
IT..... *Informationstechnik*
MVC..... *Model-View-Controller*
npm..... *node package manager*
OOP..... *Objektorientierte Programmierung*
PWA..... *Progressive-Web-Apps*
SDK ..... *Software-Development-Kit*
SPA..... *Single-Page-Application*
UI ..... *User-Interface*
URL..... *Uniform-Resource-Locator*
Wasm..... *WebAssembly*

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LISTINGS

LITERATURVERZEICHNIS

Aggarwal, Shanal (2020): Single Page Application: Hottest trend to watch for in 2020. Online verfügbar unter https://www.techaheadcorp.com/blog/single-page-application/, zuletzt aktualisiert am 06.03.2020, zuletzt geprüft am 14.10.2020.

Amann, Erwin (2019): Entscheidungstheorie. Individuelle, Strategische und Kollektive Entscheidungen. Wiesbaden: Springer Gabler. in Springer Fachmedien Wiesbaden GmbH (Studienbücher Wirtschaftsmathematik Ser).

Aponte, Michele (2020): Building single page applications in.NET Core 3. Jumpstart coding using Blazor and C#. [Berkeley, California?]: Apress.

Berners-Lee, Tim; Fischetti, Mark (1999): Der Web-Report. Der Schöpfer des World Wide Webs über das grenzenlose Potential des Internets. München: Econ.

Bitz, Michael (1977): Die Strukturierung ökonomischer Entscheidungsmodelle. Wiesbaden: Gabler Verlag.

Böhm, Robin (2017): Was sind Angular und AngularJS? - Angular.DE. Online verfügbar unter https://angular.de/artikel/was-ist-angular/, zuletzt aktualisiert am 11.10.2020, zuletzt geprüft am 11.10.2020.

Bourier, Günther. (2018): Beschreibende Statistik. Praxisorientierte Einführung - Mit Aufgaben und Lösungen. 13. Aufl. 2018. Wiesbaden: Springer Fachmedien Wiesbaden; Imprint: Springer Gabler (Lehrbuch).

Bretzke, Wolf-Rüdiger (1980): Der Problembezug von Entscheidungsmodellen. Zugl.: Köln, Univ., Habil.-Schr., 1978. Tübingen: Mohr (Die Einheit der Gesellschaftswissenschaften, 29).

Cordts, Jürgen; Lensing, Hans-Joachim (1992): ABC-Analyse. Preisanalyse für Einkäufer. Wiesbaden, s.l.: Gabler Verlag (Gabler-Studientexte).

Cwalina, Krzysztof; Abrams, Brad (2007): Richtlinien für das Framework-Design. Konventionen, Ausdrücke und Muster für wiederverwertbare.NET-Bibliotheken. München: Addison-Wesley (Microsoft.NET development series).

RFC 1034, 1987: Domain Names - Concepts and facilities. Online verfügbar unter https://tools.ietf.org/html/rfc1034, zuletzt geprüft am 17.10.2020.

Doyle, Barry; Lopes, Cristina Videira (2008): Survey of Technologies for Web Application Development. Online verfügbar unter https://arxiv.org/pdf/0801.2618.

Facebook Inc. (2020): React – A JavaScript library for building user interfaces. Online verfügbar unter https://reactjs.org/, zuletzt aktualisiert am 17.10.2020, zuletzt geprüft am 17.10.2020.

Fischer, Peter; Hofer, Peter (2011): Lexikon der Informatik. Berlin, Heidelberg: Springer Berlin Heidelberg.

Flick, Uwe; Kardoff, Ernst von; Steinke, Ines (2004): A Companion to Qualitative Research. London: Sage Publications Ltd.

Gackenheimer, Cory (2015): Introduction to React. Berkeley, CA: Apress (The expert's voice in web development).

GitHub (2020a): angular/angular. Online verfügbar unter https://github.com/angular/angular, zuletzt aktualisiert am 28.11.2020, zuletzt geprüft am 28.11.2020.

GitHub (2020b): dotnet/blazor. Online verfügbar unter https://github.com/dotnet/blazor, zuletzt aktualisiert am 28.11.2020, zuletzt geprüft am 28.11.2020.

GitHub (2020c): facebook/react. Online verfügbar unter https://github.com/facebook/react, zuletzt aktualisiert am 28.11.2020, zuletzt geprüft am 28.11.2020.

GitHub (2020d): vuejs/vue. Online verfügbar unter https://github.com/vuejs/vue, zuletzt aktualisiert am 28.11.2020, zuletzt geprüft am 28.11.2020.

Google (2020a): Angular. Online verfügbar unter https://angular.io/, zuletzt aktualisiert am 09.10.2020, zuletzt geprüft am 11.10.2020.

Google (2020b): Angular - FEATURES & BENEFITS. Online verfügbar unter https://angular.io/features, zuletzt aktualisiert am 09.10.2020, zuletzt geprüft am 11.10.2020.

Haun, Matthias (2016): Cognitive Organisation. Prozessuale und funktionale Gestaltung von Unternehmen. Berlin, Heidelberg: Springer Berlin Heidelberg.

RFC 1945, 1996: Hypertext Transfer Protocol -- HTTP/1.0. Online verfügbar unter https://tools.ietf.org/html/rfc1945, zuletzt geprüft am 17.10.2020.

ISO/IEC 2382:2015-05, Mai 2015: Information technology -- Vocabulary.

Ismail, Kaya (2019): Why Single Page Apps Are the Hottest Trend of 2020. In: *CMSWire.com*, 09.12.2019. Online verfügbar unter https://www.cmswire.com/digital-experience/why-single-page-apps-are-the-hottest-trend-of-2020/, zuletzt geprüft am 14.10.2020.

Janson, Matthias (2020): Die Welt geht ins Netz. In: *Statista*, 12.10.2020. Online verfügbar unter https://de.statista.com/infografik/23170/prognose-zur-zahl-der-weltweiten-internetnutzer/, zuletzt geprüft am 14.10.2020.

Johnson, Ralph E.; Foote, Brian (1988): Designing Reusable Classes. In: *Journal of Object-Oriented Programming*, S. 3.

Keith, Jeremy (2005): DOM Scripting. Web Design with JavaScript and the Document Object Model. Berkeley, CA: Jeremy Keith (Safari Books Online).

Kölpin, Sven (2017): SPA: Wann lohnen sich Single Page Applications - und wann nicht? Online verfügbar unter https://jaxenter.de/enterprisetales-auf-dem-weg-zur-spa-58941, zuletzt aktualisiert am 03.07.2017, zuletzt geprüft am 14.10.2020.

Krause, Stefan (2020): Interactive Results. Online verfügbar unter https://krausest.github.io/js-framework-benchmark/2020/table_chrome_87.0.4280.66.html, zuletzt aktualisiert am 20.11.2020, zuletzt geprüft am 06.12.2020.

Kurbel, Karl E. (2008): The Making of Information Systems. Software Engineering and Management in a Globalized World. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.

Lackes, Richard (2018): Definition: Softwareentwicklung. In: *Springer Fachmedien Wiesbaden GmbH*, 19.02.2018. Online verfügbar unter https://wirtschaftslexikon.gabler.de/definition/softwareentwicklung-44040/version-267361, zuletzt geprüft am 11.10.2020.

Laux, Helmut; Gillenkirch, Robert M.; Schenk-Mathes, Heike Y. (2012): Entscheidungstheorie. Berlin, Heidelberg: Springer Berlin Heidelberg.

Laux, Helmut; Liermann, Felix (2005): Grundlagen der Organisation. Die Steuerung von Entscheidungen als Grundproblem der Betriebswirtschaftslehre. 6. Aufl. Berlin: Springer (Springer-Lehrbuch).

Massol, Vincent; Husted, Ted (2004): JUnit in action. Greenwich, Conn.: Manning.

MDN Web Docs (2016): Introduction to the server side. Online verfügbar unter https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction, zuletzt aktualisiert am 29.11.2019, zuletzt geprüft am 14.10.2020.

Meinel, Christoph; Sack, Harald (2004): WWW. Berlin, Heidelberg: Springer Berlin Heidelberg.

Meyer, Roswitha (2000): Entscheidungstheorie. Ein Lehr- und Arbeitsbuch. 2., durchgesehene Auflage. Wiesbaden: Gabler Verlag.

Microsoft (2020): Blazor | Build client web apps with C# |.NET. Online verfügbar unter https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor, zuletzt aktualisiert am 17.10.2020, zuletzt geprüft am 17.10.2020.

Nixon, Robin (2014): Learning PHP, MySQL, JavaScript, CSS & HTML5. Sebastopol, CA: O'Reilly Media.

Ramb, Bernd-Thomas (2018): Definition: Objektivität. In: *Springer Fachmedien Wiesbaden GmbH*, 19.02.2018. Online verfügbar unter

https://wirtschaftslexikon.gabler.de/definition/objektivitaet-45537/version-268829, zuletzt geprüft am 11.10.2020.

Schawel, Christian; Billing, Fabian (2009): Top 100 Management Tools. Das wichtigste Buch eines Managers. 2., überarbeitete Aufl. Wiesbaden: Gabler Verlag / GWV Fachverlage, Wiesbaden.

Schneck, Ottmar; Hahn, Klaus; Schramm, Uwe; Stelzer, Matthias (2015): Lexikon der Betriebswirtschaft. 3.000 grundlegende und aktuelle Begriffe für Studium und Beruf. 9. Auflage. München: Deutscher Taschenbuch Verlag (Beck-Wirtschaftsberater im dtv, v.50942).

Schneider, Willy (2019): Praxisleitfaden SWOT-Analyse. 1. Auflage. Norderstedt: BoD – Books on Demand (Fachbuchreihe "Management-Kompetenz kompakt", 1).

Schupp, Florian (2004): Versorgungsstrategien in der Logistik. Konzeption eines modularen Entscheidungsmodells. Gabler Edition Wissenschaft. Wiesbaden: Deutscher Universitätsverlag (Logistik-Management).

Sharma, Sagar (2020): Single Page Application Development: How is the impact of growing SPA trend in 2020? Online verfügbar unter https://www.credencys.com/blog/single-page-application-development/, zuletzt aktualisiert am 14.10.2020, zuletzt geprüft am 14.10.2020.

Siepermann, Markus (2018): Definition: Single Page Application. In: *Springer Fachmedien Wiesbaden GmbH*, 19.02.2018. Online verfügbar unter https://wirtschaftslexikon.gabler.de/definition/single-page-application-54485/version-277514, zuletzt geprüft am 18.10.2020.

Stack Overflow (2020): Stack Overflow Developer Survey 2020. Online verfügbar unter https://insights.stackoverflow.com/survey/2020, zuletzt aktualisiert am 14.10.2020, zuletzt geprüft am 14.10.2020.

StatCounter Global Stats (2020): Desktop Browser Version Market Share Worldwide | StatCounter Global Stats. Online verfügbar unter https://gs.statcounter.com/browser-version-market-share/desktop/worldwide/#monthly-202009-202009-bar, zuletzt aktualisiert am 14.11.2020, zuletzt geprüft am 14.11.2020.

Steyer, Manfred; Schwab, Daniel (2017): Angular. Das Praxisbuch zu Grundlagen und Best Practices, ab Version 4. 2nd ed. Heidelberg: O'Reilly (Animals).

Steyer, Ralph (2019): Webanwendungen erstellen mit Vue.js. MVVM-Muster für konventionelle und Single-Page-Webseiten. Wiesbaden: Springer Vieweg.

Suh, Woojong (Hg.) (2005): Web engineering. Principles and techniques. Hershey: Idea Group Publ.

Sztuka, Achim (2020): Entscheidungsmodelle zur Analyse von Entscheidungsproblemen. Online verfügbar unter http://www.manager-wiki.com/methodik/37-entscheidungsmodelle, zuletzt aktualisiert am 28.11.2020, zuletzt geprüft am 29.11.2020.

Tachtsoglou, Sarantis; König, Johannes (2017): Statistik für Erziehungswissenschaftlerinnen und Erziehungswissenschaftler. Konzepte, Beispiele und Anwendungen in SPSS und R. Wiesbaden: Springer VS.

Tech Differences (2018): Difference Between Server-side Scripting and Client-side Scripting (with Comparison Chart) - Tech Differences. Online verfügbar unter https://techdifferences.com/difference-between-server-side-scripting-and-client-side-scripting.html, zuletzt aktualisiert am 17.12.2019, zuletzt geprüft am 14.10.2020.

Vahs, Dietmar; Burmester, Ralf (2009): Innovationsmanagement. Von der Produktidee zur erfolgreichen Vermarktung. 4., überarb. Aufl. Stuttgart: Schäffer-Poeschel (Praxisnahes Wirtschaftsstudium).

Vesic, Slavimir; Minović, Miroslav (2015): Single Page Applications: Trend or Future. In: *Info M*.

Wessler, Markus (2012): Entscheidungstheorie. Von der klassischen Spieltheorie zur Anwendung kooperativer Konzepte. Wiesbaden: Springer Gabler (Lehrbuch).

You, Evan (2020): Vue.js. Online verfügbar unter https://vuejs.org/, zuletzt aktualisiert am 17.10.2020, zuletzt geprüft am 17.10.2020.

Zangemeister, Christof (2014): Nutzwertanalyse in der Systemtechnik. Eine Methodik zur multidimensionalen Bewertung und Auswahl von Projektalternativen. Teilw. zugl.: Berlin, Univ., Diss., 1970. 5., erw. Aufl. Norderstedt: Books on Demand.