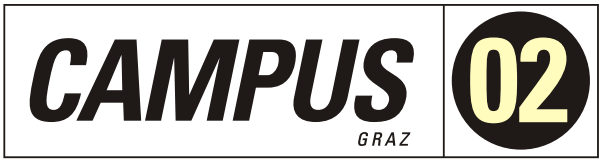
MASTERARBEIT

Full Stack Web-Applications - Vom Client bis zur Datenbank

Analysen und Kombinationen unterschiedlichster Technologien zur Ableitung einer Handlungsempfehlung

ausgeführt am



Studiengang
Informationstechnologien und Wirtschaftsinformatik

Von: Florian Götz, BSc
Pers. Kennz. 1810320006

Graz, am 7. Januar 2020

.....
Florian Götz, BSc

Ehrenwörtliche Erklärung

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

.....

Florian Götz, BSc

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich während meines Studiums und beim Erstellen dieser Arbeit unterstützt haben.

Allen voran Herrn Günther Zwetti für seine engagierte, fachliche und hilfreiche Betreuung, die oftmals über den Inhalt der Arbeit hinausging.
Des Weiteren bei meiner Schwester Nadine für das ständige Korrekturlesen.

Ebenfalls möchte ich mich beim Lehrpersonal der Fachhochschule Campus02, von dem ich in den vergangen eineinhalb Jahren sehr viel lernen durfte, bedanken.

Ganz besonders danken möchte ich meiner Familie und meinen Freunden, die mich im Laufe meines Studiums und der Entstehungszeit dieser Arbeit mit viel Geduld und Ausdauer begleitet und unterstützt haben.

Florian Götz, BSc

Graz, am 7. Januar 2020

Kurzfassung

Mit der immer größer werdenden Abhängigkeit von der Nutzung des Webs – sei es mittels Smartphones, Tablets oder Laptops – steigt folglich auch die Bedeutsamkeit von Web-Applikationen immer mehr an. Aufgrund dieser steigenden Popularität vervielfachte sich parallel zu dieser Entwicklung auch die Anzahl an Frameworks für diverse Programmiersprachen, welche im Zusammenhang mit Web-Applikationen genutzt werden.
Um das passende Framework zu finden, gibt es kritische und objektive Kriterien wie Sicherheit, Wartbarkeit, Performance oder „Time to Market“, die ebenso zu beachten sind wie die Zukunftssicherheit und die Modernität der gewählten Frameworks. Durch die Vielzahl an unterschiedlichen Frameworks wird es immer komplexer, schwieriger sowie zeitintensiver, die richtige Wahl zu treffen.
Um diese Wahl zu vereinfachen, werden in dieser Arbeit Technologien aus den einzelnen Layern miteinander kombiniert, evaluiert und auf deren Zusammenspiel hin untersucht und bewertet. Weiters wird es zu einer Verbindung zwischen Theorie und Praxis kommen, indem in diesem Arbeitsschritt die im Theorieteil gewonnen Evaluierungskriterien herangezogen und mit der Praxis in Verbindung gesetzt werden.

Abstract

As the dependency regarding the usage of the web is constantly growing – be it with smartphones, tablets or laptops – the significance of web applications is also increasing steadily. Due to this rising popularity, the amount of frameworks of various programming languages, which can be used within the context of web applications, are also multiplying at the same time.
In order to find the most suitable framework, critical and objective criteria such as security, maintainability, performance or “time to market” as well as the long-term

availability and the modernity of the frameworks should be observed in this thesis. Due to the vast number of various frameworks it has become more and more complex and time-consuming to make the right decision.
To simplify the choice, technologies from different layers will be combined, evaluated as well as examined and rated with reference to their interaction. During this phase, the previously evaluated criteria from the theoretical part will be used in order to assess the usability of these combined frameworks.

Inhaltsverzeichnis

1 Einleitung

Im Gegensatz zu den Anfängen des Word Wide Web (WWW), in welcher Websites lediglich statisch und zur reinen Informationsgewinnung seitens der/des BesucherIn genutzt wurden, drangen Mitte 2000 die ersten Smartphones in die Online-Welt vor und veränderten das Userverhalten grundlegend. Schon zu Beginn der 2010er Jahre wurde der Fokus auf Mobile First bzw. auf repsonsive Designs gelegt. Unter responsive Design versteht man, dass sich die Website je nach Displaygröße des Endgerätes automatisch anpasst, um der/dem NutzerIn das beste Nutzererlebnis unabhängig vom verwendeten Gerät zu bieten.

Aufgrund der Unausweichlichkeit eines Webauftrittes wurde das Angebot zur einfachen Erstellung von Websites immer größer. Parallel dazu änderte sich der Schwerpunkt bei der Entwicklung einer Website weg von der simplen Rechnung Hypertext Markup Language (HTML)+Cascading Style Sheets (CSS) = Webseitengestaltung, hin zu Usability, User Experience und Interaktion.(Harringer, 2018)

Wird den NutzerInnen zusätzlich zu Informationen auch Service wie beispielsweise Suchmaschinen, Webmail, Online-Spiele oder Shops angeboten, spricht man von Web-Applikationen; meist werden hierfür interaktive Elemente verwendet.(Augsten, 2017)

1.1 Hypothesen und Forschungsfrage

In welchem Umfang ist es möglich, eine einheitliche Handlungsempfehlung für die vollständige Entwicklung einer Web-Applikation in Bezug auf den Einsatz unterschiedlicher Frameworks aus verschiedenen Layern abzugeben, aus welcher der/die LeserIn Vorteile ziehen und den er/sie als Leitfaden nutzen kann?

Veränderungshypothese

H1: Eine einheitliche Handlungsempfehlung für die Entwicklung von Web-Applikationen wirkt sich positiv auf die Effizienz des Entwicklungsprozesses aus.

H2: Eine einheitliche Handlungsempfehlung für die Entwicklung von Web-Applikationen wirkt sich negativ auf die Effizienz des Entwicklungsprozesses aus.

Zusammenhangshypothese

H1: Durch die Zuhilfenahme einer Handlungsempfehlung während der Entwicklung einer Web-Applikation wird die Übersicht über die einzelnen Technologien und Teilbereiche signifikant verbessert und die Komplexität deutlich verringert.

H2: Die Zuhilfenahme einer Handlungsempfehlung während der Entwicklung einer Web-Applikation hat weder eine Auswirkung auf eine verbesserte Übersicht über die einzelnen Technologien sowie Teilbereiche, noch verringert sie die Komplexität.

1.2 Ausgangssituation

Mit der immer größer werdenden Abhängigkeit und Nutzung des Webs – sei es mittels Smartphones, Tablets oder Laptops – steigt die Bedeutsamkeit von Web-Applikationen immer mehr an.
Aufgrund dieser steigenden Popularität vervielfachte sich parallel dazu auch die Anzahl an Frameworks für diverse Programmiersprachen, welche im Zusammenhang mit Web-Applikationen genutzt werden.

Kritische und objektive Kriterien wie Sicherheit, Wartbarkeit, Performance oder „Time to Market“sind ebenso zu beachten wie die Zukunftssicherheit und die Modernität der gewählten Frameworks. Durch die Vielzahl an unterschiedlichen Frameworks wird es immer komplexer, schwieriger sowie zeitintensiver, die richtige Wahl zu treffen.

1.3 Zielsetzung der Arbeit

Im Zuge dieser Masterarbeit sollen diverse Technologien aus den unterschiedlichsten Layern analysiert und anschließend miteinander kombiniert werden, um herauszufinden, ob sich diese für eine gemeinsame Nutzung eignen würden. Weiters soll eine individuelle Analyse der Technologien hinsichtlich ihrer Nutzbarkeit durchgeführt werden.
Dabei soll eine Handlungsempfehlung entstehen, welche dem/der LeserIn bei den Entscheidungen bezüglich der Wahl der Frameworks in den unterschiedlichen Schichten (Model - View - Controller) unterstützt.

1.4 Vorgehen und Methodik

Im Theorieteil der Masterarbeit werden - unter Hilfenahme von Fachliteratur - die wichtigsten und relevantesten Frameworks der einzelnen Schichten identifiziert, um eine sinnvolle Weiterarbeit zu ermöglichen. Anhand dieser Recherche und den daraus gewonnenen Erkenntnissen werden für die Technologien objektive Faktoren ausgewählt, welche für die Evaluierung herangezogen werden.

Der darauffolgende Praxisteil der Arbeit stützt sich auf die gewonnenen Erkenntnisse des Theorieteils. Dabei werden Technologien aus den einzelnen Layern miteinander kombiniert, evaluiert und auf deren Zusammenspiel hin untersucht und bewertet. Hier kommt es zu der Verbindung zwischen Theorie und Praxis, denn dieser Arbeitsschritt erfolgt unter der Zuhilfenahme der im Theorieteil gewonnen Evaluierungskriterien.

Im ersten Schritt erfolgt eine Literaturrecherche, um die relevantesten Frameworks für Web-Applikationen zu erheben und sich einen Überblick über die verfügbaren Möglichkeiten zu schaffen. Weiters werden im Zuge dieser Recherche kritische sowie objektive Faktoren für die Evaluierung der Brauchbarkeit dieser Technologien bestimmt. In weiterer Folge werden auch die einzelnen Schichten, welche bei „Full Stack Development“einer Web-Applikation von Nöten sind, aufgelistet und kurz beschrieben, um eine fundierte Basis für die weitere Arbeit zu schaffen.

Weiters wird eine kurze Marktanalyse Aufschluss darüber geben, ob eventuell bereits ein bestehendes Angebot existiert, welches eine vollständige Entwicklung einer Web- Applikation ermöglicht.
Aufbauend auf den vorangegangenen Schritt wird im Praxisteil genauer auf die einzelnen Frameworks und deren Zusammenspiel mit Frameworks aus anderen Layern eingegangen. Hierzu werden jene Technologien miteinander kombiniert und mit Hilfe der davor ausgearbeiteten Faktoren evaluiert. Neben der Kompatibilität untereinander wird auch jede Technologie individuell beleuchtet, um zu sehen, ob das gewählte Framework auch die zuvor erhobenen Kriterien erfüllt.
Als Abschluss der Arbeit erfolgt eine Diskussion, welche auf die zuvor gewonnenen Erkenntnisse aufbauen wird. Hier wird unter anderem beurteilt, inwieweit es überhaupt möglich ist, eine einheitliche Handlungsempfehlung im Bereich des „Full Stack Developments“einer Web-Applikation abzugeben.

2 Allgemeines zu Web-Applikationen

Allgemein versteht man unter einer Web-Applikation ein interaktives Programm, welches Webbrowser und Web-Technologien verwendet, um gewisse Aufgaben über das Internet zu erledigen.

Serverseitige Skripts, wie beispielsweise PHP, welche für das Speichern sowie das Abrufen der Daten zuständig sind, werden ebenso von Web-Applikationen verwendet wie clientseitige Skripts, wie zum Beispiel JavaScript oder HTML, die für die Darstellung der Informationen zuständig sind. Beispiele für solche Anwendungen sind unter anderem Online-Formulare, Web-Shops oder auch Suchmaschinen. Anders als bei herkömmlichen Programmen muss für die Verwendung von einer Web-Applikation nichts am Rechner der NutzerInnen installiert werden. Dies ermöglicht es, Daten unterschiedlichster Art plattformunabhängig und dezentralisiert zu be- und verarbeiten.

Unterschieden wird zwischen dynamischen und statischen Applikationen, wobei der Unterschied darin liegt, dass bei einer dynamischen Web-App zusätzlich serverseitige Prozesse von Nöten sind.
Der grundsätzliche Aufbau einer solchen Anwendung sieht meist folgendermaßen aus:

- Web-Server um die Daten der Userin/des Users zu verwalten
- Einen Application-Server um den angeforderten Task durchzuführen
- Wenn Informationen respektive Daten gespeichert werden sollen, ist auch noch zusätzlich eine Datenbank nötig

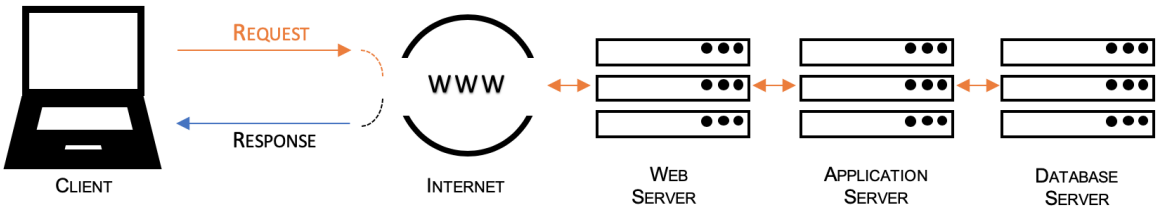


Abbildung 2.1: Aufbau einer Web-Applikation
(vgl. Rawindu, 2019)

Web-Applikationen bringen große Vorteile mit sich; diese reichen von der Plattformunabhängigkeit bis hin zu der Reduzierung von illegaler Verwendung der Software mittels eines Abonnement-Modells, bei dem sich die/der NutzerIn online registrieren muss. (Augsten, 2017)

Wie der Großteil der Technologien, bzw. Techniken in der IT, sind auch Web-Applikationen einem stetigen Wandel ausgesetzt. Neue Web-Technologien, wie in letzter Zeit beispielsweise Vue.js, drängen in den Markt und Useranforderungen wie etwa Reaktionsgeschwindigkeit, Stabilität und Sicherheit nehmen kontinuierlich zu. Aufgrund der rasanten Geschwindigkeit mit der sich das WWW verändert, ist es vielen Technologien nicht möglich sich in dieser kurzen Zeit dahingehend weiterzuentwickeln und den EntwicklerInnen die neuesten Features bereitzustellen.

Viele dieser Frameworks, welche für eine kurze Dauer Big Players in der Web-Entwicklung waren, wie z.B. Java Applets, sind nun nahezu aus der Web-Programmierung verschwunden. Parallel dazu haben sich andere Technologien von einfachen Werkzeugen zu mächtigen und wichtigen Werkzeugen entwickelt; als Beispiel ist hier JavaScript zu nennen. (Pop & Altar, 2013)

Neue Technologien bringen neue Möglichkeiten für EntwicklerInnen auf den Markt und können zu einer Simplifikation des Entwicklungsprozesses führen. Ein Beispiel sind etwa Progressive Web Applications, mithilfe dieser ist es möglich, Web-Applikationen die Mehrzahl des Funktionsumfanges einer nativen App mitzugeben.

2.1 Aktuelle Marktsituation

Individuelle Technologien, exemplarisch eine reine Clienttechnologie wie React oder Datenbanktechnologien wie MySQL, werden am Markt hinreichend angeboten.
Durch die Schnelllebigkeit der Branche, die den hohen Anforderungen der/des EndnutzerIn und dem unterschiedlichen Verhalten bei der Kombination der unterschiedliche Layertechnologien (Client, Server und Datenbank) untereinander, existiert eine steigende Nachfrage nach ganzheitlichen Softwarelösungen. Das Kernproblem weshalb - auch nach gründlicher Marktrecherche - keine ganzheitliche Softwarelösung gefunden wurde, ist, dass je nach Art der Applikation Anforderungen stark variieren.

Aus diesem Grund gibt es meist auch nur Empfehlungen welches Framework sich für Back-End, Front-End und Datenbank eignen können. Auch eine Kombination der Frameworks untereinander wurde während der Recherche selten angefunden. Weil Anwendungen wechselnde Anforderungen haben und sich immer neue Technologien entwickeln, wird im Praxisteil dieser Arbeit auch darauf geachtet, dass die einzelnen Komponenten ohne komplexen Aufwand ausgetauscht werden können.

Weil gerade bei der Entwicklung einer Web-Applikation EntwicklerInnen Wissen bezüglich Front-End und Back-End mitbringen müssen und sich die Technologien im WWW rapide ändern, ist es relativ komplex, wartbaren Code sowie eine stabile Web-Anwendung zu erzeugen. Das unausweichliche Zusammenspiel zwischen Client, Server und Datenbank macht es verhältnismäßig schwierig eine gut wartbare und erweiterbare Applikation zu schreiben. Wird auf der Front-End Seite etwas verändert, muss darauf geachtet werden, dass diese Änderung mit dem Server Code kompatibel ist. Damit die Kommunikation zwischen den einzelnen Schichten reibungslos von Statten gehen kann, wird zur Zeit am Markt empfohlen, den Code mittels dem MVC Pattern zu splitten. (Pop & Altar, 2013)

2.2 Front End

Als Front-End versteht man alles, was die/der UserIn sieht, also das Design, welches mittels HTML und CSS gestaltet wird. Doch das Hauptziel, speziell bei Web-Applikationen bei der Front-End Entwicklung, ist die Interaktion mit der/dem NutzerIn. (Soni, 2017)

2.2.1 Allgemein

Speziell clientseitig sind - auf Basis von JavaScript - in den vergangenen Jahren eine Reihe an neuen Technologien auf den Markt gekommen. Applikationen haben seitens der NutzerInnen den Anspruch neben einer vorzeigbaren GUI auch dynamisch und performant zu sein. Durch diese gestiegenen Anforderungen und die Erwartung einer Echtzeitanzeige - der/die UserIn sieht Änderungen in der GUI direkt nach der Durchführung von Änderungen als Benachrichtigungen - werden Frameworks wie React, Angular oder Vue.js bei der clientseitigen Programmierung immer beliebter.
Zusätzlich wurden die beiden Standards in der Webentwicklung, HTML und CSS, kontinuierlich weiterentwickelt und mit HTML5 sowie CSS3 sind diese beiden Technologien mit dem letzten Stand der Technik ausgestattet und unterstützen JavaScript bei der Erstellung von Applikationen im Web wunderbar.

Ein Augenmerk bei der Entwicklung des Front-End ist darauf zu legen, dass der/die UserIn sich darin intuitiv bewegen kann. Die Benutzeroberfläche soll, neben einem ästhetischen Aussehen, so entwickelt sein, dass jegliche Komponenten und Features dort platziert sind, wo die NutzerInnen diese auch vermuten.(Northwood, 2018)

Im Gegensatz zu den klassischen Websites in welchen einzelne Seiten miteinander verlinkt bzw. ineinander integriert sind, ist es innerhalb von dynamischen Websites nicht mehr nötig die Seite für neuen Inhalt zu wechseln bzw. die komplette Seite neu zu laden, um etwaige Inhalte sehen zu können.

Für die Umsetzung solcher dynamischer Änderungen nutzen Scriptsprachen wie Javascript bzw. darauf aufbauende Frameworks wie Vue.js, Angular oder React, das sogenannte Document Object Model oder kurz DOM. DOM Elemente sind nichts

anderes als HTML Tags, wie beispielsweise `<div>`, auf welche mittels Javascript zugegriffen werden kann.
Ein Punkt, welcher gerade bei der Entwicklung des Front-Ends nicht vernachlässigt werden darf, ist die verwendeten Funktionalitäten darauf zu überprüfen, ob alle gängigen Browser diese auch unterstützen. (Fernandes, Costa & Carrico, 2012)

Um eine dynamische Website zu erzeugen und das ständige Neuladen von Seiten zu umgehen, wird Asynchronous JavaScript and XML (AJAX) verwendet. Mithilfe von AJAX ist es möglich, sowohl Daten zum Server zu senden, als auch Daten vom Server wieder zu empfangen. (Shahzad, 2017)

2.2.2 DOM (Document Object Model)

Sobald eine Interaktion mit Elementen innerhalb der Web-Applikation erfolgt, kommt man bei der Verwendung von HTTP mit dem DOM in Berührung. Aufgebaut ist dieses Objekt wie eine Baumstruktur, sprich Tags sind ineinander verschachtelt und hierarchisch aufgebaut.

Für die Interaktion bzw. die Veränderung von HTML-Komponenten wird mittels JavaScript (oder auf JavaScript aufgebaute Frameworks wie Vue.js) auf einzelne DOM Objekte zugegriffen und folglich werden diese manipuliert.

Ein weiteres wichtiges Feature, welches das DOM mit sich bringt, ist die Verfügbarkeit von Events, welche auf zuvor vordefinierte Ereignisse reagieren. Mithilfe dieser Events können DOM Elemente für die Interaktionsfähigkeit mit Funktionen versehen werden, welche ausgeführt werden, wenn das Element beispielsweise geklickt wird und/oder mittels sogenannten Event Listeners, welche auftreten, sobald ein Event des definierten Typs auftritt, aktiviert wird. (Northwood, 2018)

```
1 // Wird ausgeloest wenn Seite fertig geladen wurde
2 window.addEventListener('load', alert("Fertig geladen"));
3
4 // Wird ausgeloest wenn Element mit der ID "masterthesis" ↩
    geklickt wurde
5 document.getElementById("masterthesis").addEventListener("↩
    click",alert("Foo Bar");
```

Listing 2.1: JavaScript DOM Events

2.3 Back-End

Interagiert der/die UserIn mit der Web-Applikation, wird eine Reaktion bzw. Antwort seitens der App erwartet und exakt diese Bearbeitung und Rückmeldung an die/den EndnutzerIn erfolgt am Server.
Der Back-End Code einer Applikation kann bildlich gesprochen als das Gehirn der Anwendung angesehen werden. Jegliche Interaktionen, welche eine Reaktion der Anwendung erwarten, werden auf dem Server mithilfe von Serversprachen wie Java oder PHP ver- und bearbeitet und userfreundlich aufbereitet. (Soni, 2017)

2.3.1 Allgemein

Grundsätzlich kann man diese Schicht als Middleware innerhalb eines Applikationssystems sehen, denn genau hier gehen die Anfragen von Userseite ein und werden dementsprechend weiterverarbeitet. Übertragen werden die Informationen vom Client hin zum Server und zurück über HTTP(s).

Auf Serverebene können die Schnittstellen zu etwaigen Datenbanken laufen, die es möglich machen, dass die UserInnen Informationen abrufen sowie speichern können. Kommunikation unter den AnwenderInnen wird über den Server geleitet.
Wie die Client-Server-Kommunikation aussehen soll, ist von Web-Applikation zu Web-Applikation unterschiedlich. Die Kommunikation kann unter der Verwendung von WebSockets in Echtzeit stattfinden oder traditionell durch eine bewusste Interaktion der UserInnen mit dem Server, wie beispielsweise einem Klick auf einen Button.

2.3.2 Websockets

Um eine Echtzeitkommunikation zu ermöglichen, bieten Websockets neben der Client-Push-Funktion zusätzlich eine Push-Funktion für den Server an. Veranschaulicht bedeutet das, dass der Server empfangene Informationen direkt an einen weiteren Client senden kann. Das heißt, dass während bei HTTP eine vorangegangen Anfrage des Clients stattgefunden haben muss damit der Server eine Aktion startet, reicht es bei TCP aus, dass der Client die Verbindung einmalig öffnet und dies dem Server zur Verfügung stellt.

WebSockets sind eine Zweiwegkommunikation zwischen einem Web-Browser und einem Server. Da es sich um eine persistente Verbindung handelt, muss der Server darauf ausgelegt sein, eine große Anzahl an Verbindungen gleichzeitig verarbeiten zu können. Auch wenn dies möglich ist, muss der Server eine hohe Anzahl an Verbindungen abarbeiten können. Zusätzlich muss darauf geachtet werden, dass weitere Ressourcen, wie zum Beispiel die Datenbank, ebenfalls eine Vielzahl an Verbindungen zulässt.(Soni, 2017)

Ein Vorteil des Konzeptes der WebSockets ist, dass die Verbindung welche geöffnet wird, mit dem TCP Protokoll läuft, die ein Upgrade zu HTTP darstellt, da eine Verbindung die Fähigkeit besitzt, sowohl Daten an den Server zu senden, als auch dieselbe Informationen vom Server zu empfangen.(Lombardi, 2015)

2.4 Datenbank

Der zentrale Baustein bei der Entwicklung einer Web-Applikation zum Speichern der Eingaben, welche die/der UserIn tätigt, ist die Datenbank. Wurde früher nur textueller Inhalt gespeichert, so werden heute zusätzlich Multimedia-Daten wie Bilder, Videos oder Musik gespeichert. Datenbanken können von der Speicherung der Bestellungen innerhalb eines Online Shops, bis hin zur Auswertung für beispielsweise individuelle Werbung für die NutzerInnen verwendet werden.

2.4.1 Allgemein

Grundsätzlich ist eine Datenbank eine Sammlung von Daten, die einen Auszug aus der realen Welt repräsentieren. Weiters besitzt eine Datenbank die folgenden Eigenschaften:

- spiegelt Aspekte der realen Welt wider
- Daten besitzen einen logischen Zusammenhang mit einer bestimmten inhärenten Bedeutung
- wird für einen bestimmten Zweck entwickelt und von spezifischen Anwendern verwendet und mit Daten befüllt

(Elmasri & Navathe, 2009)

Datenbanken im Kontext des Webs werden für die Speicherung sowie für das Abrufen der Daten verwendet. Zusätzlich ist die Datenbankebene für die Integrität der Daten verantwortlich und es muss ein schneller und flexibler Zugriff auf den Inhalt gewährleistet sein. (Williams & Lane, 2004)

Die Wahl der Technologie für die Datenbank ist einerseits abhängig von der Anzahl der Daten, andererseits spielt auch die Verarbeitung der Daten eine Rolle.
Bewegt sich die Web-Applikation im Big Data Bereich, bietet sich als Datenbank vielmehr eine NoSQL Datenbank wie MongoDB an, als eine relationale Datenbank. Während die Bedeutsamkeit von NoSQL Datenbanken erst seit ein paar Jahren ansteigt erfahren relationale Datenbanken wie MySQL seit geraumer Zeit große Beliebtheit.

2.4.2 Relationale vs NoSQL Datenbanken

Zur besseren Einordnung von NoSQL respektive relationalen Datenbanken werden die beiden Technologien kurz beschrieben und ihre Unterschiede beleuchtet.

Grundlegend kann die Aussage getroffen werden, dass ein relationales Datenbankmodell aus einer Reihe von Tabellen, welche sogenannte Tupel beinhalten, besteht,

welche in gewisser Weise in Relation zueinander stehen. Diese Verbindung wird mithilfe von Schlüsseln hergestellt. Schlüssel bzw. IDs müssen innerhalb einer Tabelle eindeutig sein, denn durch diesen Key kann mittels Abfragen, wie in Abbildung 2.2 zu sehen ist, auf den gewünschten Datensatz zugegriffen werden.

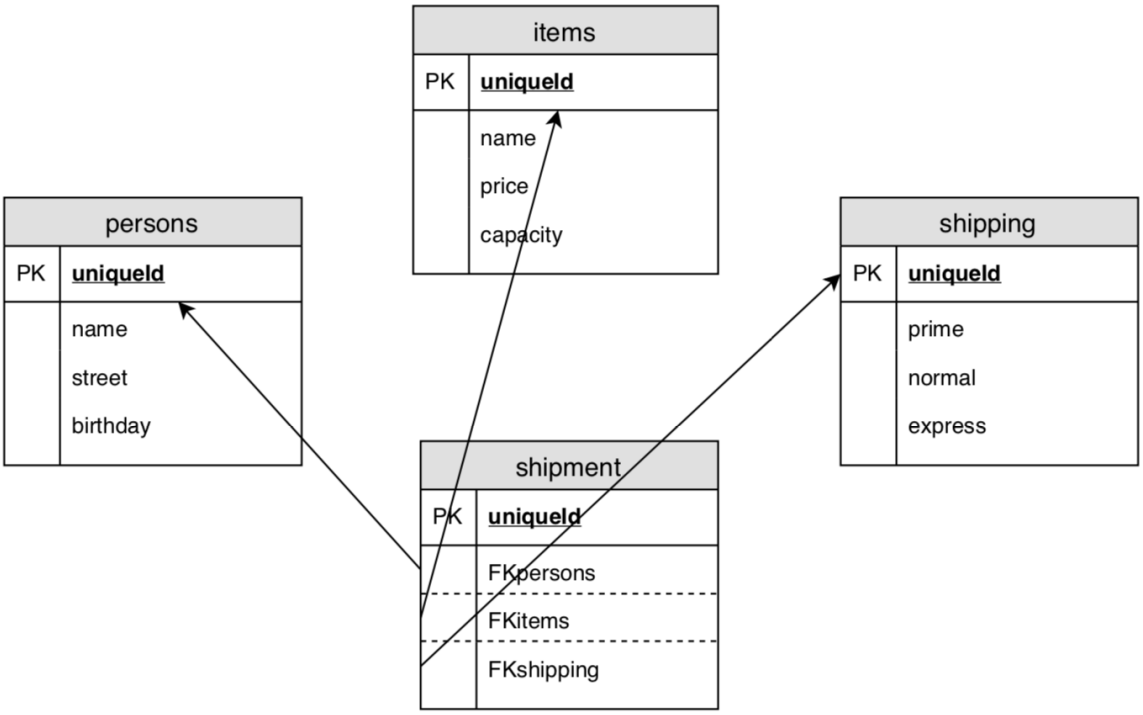


Abbildung 2.2: Abfrage innerhalb einer relationalen Datenbank (Quelle: eigene Darstellung)

Charakteristisch für relationale Datenbanksysteme sind ein Datenbankmodell, welches Datenbeziehungen in Form von Tabellen ausdrückt, Merkmale sowie Definitionen der Tabellen werden in einem übergeordneten Schema beschrieben, Daten und die Anwendung selbst sind weitestgehend getrennt, die Skriptsprache innerhalb des Systems ist SQL. Es können mehrere Benutzer gleichzeitig Daten eingeben beziehungsweise abfragen, Daten müssen konsistent sein, sprich die Datenspeicherung erfolgt fehlerfrei und korrekt und zuletzt müssen die Informationen mittels Sicherung vor Verlust geschützt werden.

Für die meisten Unternehmen reichen relationale Datenbanken aus. Werden jedoch Anwendungen im Web verteilt oder wird die Verarbeitung von großen Datenmengen, also Big Data, erforderlich, müssen die Datenbanksysteme um NoSQL Systeme erweitert werden.
Im Gegensatz zu relationalen Datenbanken erfolgt das Speichern der Daten nicht in Tabellen und die Sprache, welche für die Datenbank verwendet wird, ist nicht SQL.

Die Datenbankarchitektur, welche für NoSQL Datenbanksysteme verwendet wird, wird meist als verteiltes System konzipiert. Die Informationen selbst werden je nach Art als Schlüssel-Wert-Paare, in Spalten, Dokumentspeichern oder Graphen gespeichert. (Meier & Kaufmann, 2016)

Durch das Lockern der strikten Regeln, welche man innerhalb von relationalen Datenbanken vorfindet, erhöht sich die Performance innerhalb von NoSQL Datenbanken erheblich. Um die Skalierbarkeit sowie die Verfügbarkeit zu erhöhen, wird innerhalb von NoSQL-Systemen nicht mehr so streng auf die Konsistenz der Daten geachtet. Nutzt die Datenbank die Dokumentspeicher-Technologie, werden Textdaten unstrukturiert gespeichert und der Zugriff auf diese Daten erfolgt über den Dokumenteninhalt.

Bei der Technologie der Schlüssel-Wert-Paare weisen definierte Schlüssel auf bestimmte Werte hin; beispielhafte Vertreter sind Cache-Speichersysteme. Wird spaltenorientiert gearbeitet, werden die Daten als Schlüssel-Wert-Relation abgelegt. Ziel dieser Technologie ist es, die Performance der Berechnung der Datensätze bei der Ein- sowie Ausgabe zu verbessern. Um Beziehungen zwischen Daten optimal dazustellen, eigenen sich graphenorientierte Systeme, welche die Daten in einer Art Baumstruktur abspeichern. (Gull, 2012)

3 Anforderungen an Web-Applikationen

Für die Validierung der im Praxisteil erfolgenden Kombinationen der einzelnen Frameworks der jeweiligen Schichten, werden in dieser Arbeit die objektiven Faktoren, GUI Funktionalität (interaktiv, dynamisch), Security, Performance, Time-To-Market, Stabilität und Verfügbarkeit, Austauschbarkeit sowie die Zukunftssicherheit der gewählten Frameworks als Evaluierungskriterien herangezogen.
Diese Messwerte können innerhalb einer Web-Applikation in den seltensten Fällen individuell betrachtet werden, da die einzelnen Aspekte in gewisser Weise in Relation zueinanderstehen.

Werden Performanceprobleme festgestellt und dahingehend etwas in der Software optimiert, darf aufgrund dieser Optimierungsarbeiten die Stabilität und Verfügbarkeit des Systems nicht leiden.
Rückt das angepeilte Releasedatum rapide näher, darf aufgrund des Zeitengpasses nicht die Security darunter leiden.
Speziell bei einem Komponentenwechsel muss das große Ganze im Blick behalten werden. Auch wenn die neue Technologie noch so viele Vorteile für die GUI bietet, dürfen die anderen Faktoren nicht negativ beeinflusst werden.

Anhand dieser trivialen Beispiele erkennt man bereits relativ gut, weshalb bei der Evaluierung die einzelnen Kriterien nicht nur in sich geschlossen bewertet bzw. validiert werden dürfen.

3.1 GUI

Neben der Ladezeit der Anwendung ist die GUI das Erste mit dem der/die NutzerIn in Berührung kommt. Eine gut designte und durchdachte Benutzeroberfläche hat erheblichen Einfluss darauf, ob die/der KundIn sich weiter mit der Web-Applikation befasst oder nicht. Ebenfalls nicht zu vernachlässigen ist, dass die GUI intuitiv von der/dem EndnutzerIn genutzt werden soll und ebenfalls ist es von Vorteil, wenn ein Corporate Design ganzheitlich erkennbar ist.

Farben und Kontraste spielen innerhalb der GUI eine ebenso wichtige Rolle wie die Interaktionsfähigkeit der Elemente. Werden Schrift- und Hintergrundfarbe unglücklich bzw. nicht aufeinander abgestimmt ausgewählt, werden KundInnen die Anwendung relativ schnell wieder verlassen, da die Benutzbarkeit sehr darunter leidet, wenn die Lesbarkeit von textuellen Inhalt nicht gegeben ist.(Northwood, 2018)

In der Entwicklung einer Web-Anwendung soll die GUI meist interaktiv und dynamisch sein. Um die Interaktion zu gewährleisten ist eine Aktion seitens der Benutzerin/des Benutzers sowie eine Reaktion seitens der Anwendung erforderlich. Führt ein/e UserIn eine Aktion aus, beispielsweise durch einen Klick auf einen Button, wird eine Reaktion, zum Beispiel das Öffnen eines Pop-Ups, von der Anwendung erwartet.

Von dynamischen Inhalt spricht man, wenn durch Aktionen der Userin/des Users, Daten beispielsweise innerhalb der Datenbank verändert werden und direkt an die Benutzerin/den Benutzer verarbeitet zurückgesendet werden und den Inhalt bzw. die Struktur der Oberfläche somit verändert. Als Beispiel kann ein Online-Shop herangezogen werden; wird der Warenkorb befüllt, wird in der Datenbank ein neuer Eintrag angelegt und die/der KundIn sieht, dass ein Produkt mehr im Warenkorb liegt. (Bauer, 2009)

Wie anhand von Abbildung 3.1 gezeigt wird, bearbeitet der Web-Server mithilfe des Application Servers die HTML-Datei und sendet diese verändert an den Browser zurück. Diese Kommunikation erfolgt je nach gewählter Technologie entweder mittels HTTP(s) oder TCP wenn WebSockets im Einsatz sind.

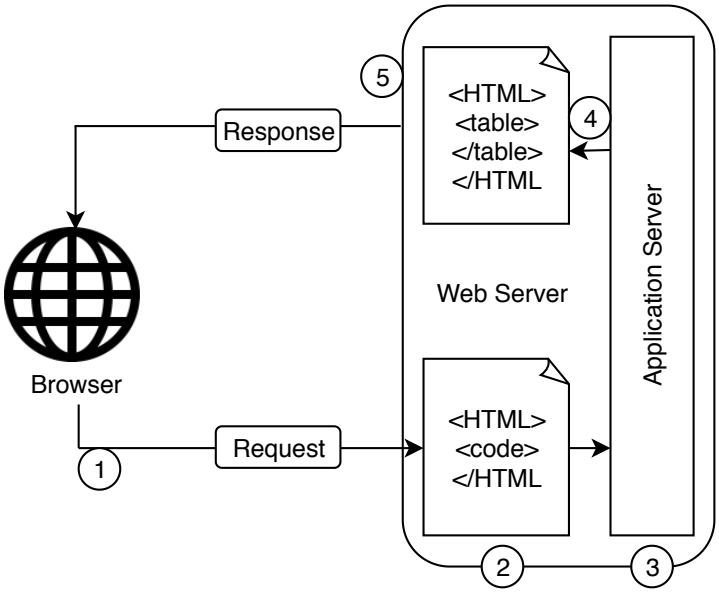


Abbildung 3.1: Verarbeitung dynamische Seite
(vgl. *Grundlegendes zu Webanwendungen*, 2017)

3.2 Security

Ein unausweichlicher Aspekt während eines Entwicklungsprozesses ist die Security, da in Zeiten, in welchen sensible Daten innerhalb des Internet versendet werden, darauf geachtet werden muss, wie die Applikation mit diesen Informationen umgeht. Wird von Entwicklungsbeginn weg dem Securityaspekt eine gewisse Beachtung entgegengebracht und bei Weiterentwicklungen berücksichtig, ist es ohne große Komplexität möglich, einen gewissen Sicherheitsstandard innerhalb der Applikation zu erzeugen. Mit der Hilfe von diversen Libraries und Patterns ist es darüber hinaus - bis zu einer gewissen Größe der Anwendung - auch ohne Sicherheitsexperten möglich, Web-Applikationen sicher zu gestalten.

Grundsätzlich sollte der Sicherheitsaspekt bei der Entwicklung immer beachtet werden, trotzdem sollte der/die EntwicklerIn zu jedem Zeitpunkt wissen, wie sicherheitskritisch die Komponenten, an denen gerade gearbeitet wird, sind. Werden beispielsweise personenbezogene Daten oder Kreditkarteninformationen verarbeitet, sollte gleich zu Beginn die komplette Komponente mittels Libraries sicher gestaltet werden.
So sollten API Keys, Passwörter oder SSH Keys immer separat vom Sourcecode gespeichert werden. Im Bereich der Web-Anwendungen sollte stets eine goldenen Regel

beachtet werden: Eingabe validieren, Ausgabe bereinigen.

Dies heißt, wenn Eingaben von UserInnen oder externen Systemen empfangen werden, sollten diese immer dahingehend validiert werden, ob dieser auch den erwartenden Werten ähnlich ist. Um diese Validierungen konstant zu halten, sollte diese immer zentralisiert, beispielsweise in einem Controller, erfolgen. Ausgaben bereinigen sagt nichts weiter aus, als dass jegliche Eingaben, welche von der Anwendung versendet werden, die Struktur der Nachricht nicht manipulieren. (Northwood, 2018)

Werden Daten innerhalb des Webs übertragen, kommt man an HTTP bzw. HTTPS nicht vorbei. Während unter HTTP alle Daten unverschlüsselt übertragen werden, verschlüsselt HTTPS all diese Informationen. Unterstützt wird HTTPS von TLS, welches die standardisierte Sicherheitstechnologie ist, wenn eine verschlüsselte Verbindung zwischen Webserver und Browser hergestellt werden soll. (Vukadinovic, 2018)

Eine weitere eventuell kritische Sicherheitslücke von Web-Applikationen ist die Möglichkeit seitens der UserInnen, schädliche Scripts auf dem Web-Server der Anwendung auszuführen; dies nennt sich Cross-Site Scripting. Zusammengefasst manipuliert hier der/die NutzerIn den Code bzw. das DOM dahingehend, dass Schadsoftware auf dem Web-Server ausgeführt wird. Verhindert werden können solche Attacken beispielsweise indem man kritische Zeichenfolgen maskiert. So bietet beispielsweise PHP den Befehl `htmlspecialchars()` an, um solche Zeichenfolgen zu verschleiern. (Deepa & Thilagam, 2016)

Um bekannte und häufig auftretende Sicherheitslücken im Umfang des Webs zu identifizieren, bietet sich die Open Web Application Security Project (OWASP)[1] an. Die nachfolgenden Punkte sind mithilfe dieser Website gefunden worden.

3.2.1 Injection

Als einen Injection-Angriff bezeichnet man ein Vorgehen, bei dem manipulierte Befehle bzw. Daten den normalen Informationen angehängt werden. Dies können beispielsweise SQL-Befehle sein, um die Datenbank zu manipulieren und eigentlich unzugängliche Informationen auszulesen. (Burkhart, 2019)

[1] https://www.owasp.org/

3.2.2 Broken Authentication

Mithilfe einer Broken Authentication Attacke gewähren sich Außenstehende unbefugten Zutritt zum System und erhalten so Zugriff auf jegliche Informationen, welche sich innerhalb der Anwendung befinden. Möglich gemacht werden solche Attacken durch ein schlechtes Session Management, wenn z.B. die/der UserIn auch nach einer längeren Inaktivität nicht abgemeldet wird und die Session ewig weiterbesteht. Ebenfalls können mithilfe einer Wörterbuchattacke unendlich oft diverse Kombinationen von Benutzernamen und Passwörtern ausprobiert werden, sofern das System die Anmeldeversuche nach einer gewissen Anzahl an Versuchen nicht blockiert. (Vizcarra, 2019)

3.2.3 Sensitive Data Exposure

Bei diesen Angriffstypen zielen es AngreiferInnen auf sensible Daten wie Passwörter, Kreditkartennummern, Anmeldedaten oder Sozialversicherungsnummern ab. Speziell wenn es um kritische Daten von UserInnen der Applikation geht, muss eine/ein EntwicklerIn sicherstellen, dass diese Informationen für Externe unzugänglich sind. Wird innerhalb der Anwendung für die Kommunikation nicht gänzlich auf TLS gesetzt, ist es für AngreiferInnen relativ simpel, die Cookies der UserInnen zu bekommen und dadurch Zugriff auf die Anmeldedaten zu bekommen. (Vizcarra, 2019)

3.2.4 Cross-Site Scripting

Cross-Site Scripting innerhalb einer Web-Applikation ist dann möglich, wenn die EntwicklerInnen es zulassen, dass Fremdcode ausgeführt werden kann. Bei dieser Attacke wird bösartiger Code in die URL eingefügt, welcher sodann bei dem Aufruf dieses Links ausgeführt wird. (Wetter, 2018)

3.3 Performance

Startet ein/e UserIn eine Applikation, ist das Erste mit dem diese/r in Berührung kommt die Ladezeit. Hier entscheidet sich schon, ob sich AnwenderInnen weiter mit der Anwendung beschäftigen oder diese wieder verlassen. Ein Overflow an Daten, welche zu Beginn geladen werden müssen, hemmen die Ladezeit. Aufgrund dessen sollte gerade während des Startvorganges darauf geachtet werden, dass wirklich nur die Informationen und Komponenten geladen werden, welche der/die BenutzerIn benötigt.

Jegliche Interaktionen zwischen AnwenderIn und Applikation sollten ebenso flüssig vonstatten gehen, wie die Kommunikation mit dem Server. Pakete, welche die KundInnen an den Server senden respektive erhalten, sollten nicht mehr Informationen enthalten als nötig; so ist auch bei einer schwachen Netzwerkverbindung eine flüssige Kommunikation gegeben.

Für die Messung von Performance spielen einige Faktoren eine Rolle, so auch das subjektive Empfinden der Nutzerin/des Nutzers, welche/r gerade die Anwendung bedient. Um die Messung objektiver zu gestalten, konzentriert sich diese Arbeit auf die folgenden Messwerte für die Performance:

- Antwortdauer, Dauer bis eine Antwort auf eine Anfrage geschieht
- Anzahl an Anfragen innerhalb einer Zeit, welche die Applikation abarbeiten kann
- Verarbeitungszeit einer Anfrage, Dauer wie lange die Anwendung benötigt, um die Anfrage an den Server weiterzugeben
- Anzahl der gesendeten Anfragen, je mehr Anfragen gesendet werden desto langsamer wird das System werden
- Servergeschwindigkeit, wie lange dauert es bis der Server die nötigen HTML Komponenten lädt

(Deabes, 2017)

Im Bereich der Web-Anwendungen ist das Hauptaugenmerk auf die Antwortzeit zu legen. Reagiert eine Applikation bzw. ein Server relativ schnell auf UserInnen oder Aktionen, wird die Anwendung meist als sehr performant angesehen.

Um die oben genannten Werte innerhalb einer Anwendung zu messen bzw. zu testen, können Load-Tests, Stress-Tests und Reaktions-Tests durchgeführt werden.
Bei Load-Tests wird die Applikation dahingehend getestet, wie sich die Datenbank, der Application Server oder der Web-Server unter bestimmten Umständen, wie beispielsweise einer großen Useranzahl, verhält.

Mittels Stress-Tests kann das obere Limit der App herausgefunden werden, sprich ab welcher Anzahl an Zugriffen sich die Performance des Systems reduziert; meist wird hier weit über der zu erwartenden Anzahl an NutzerInnen getestet.
Bei Reaktions-Tests wird geprüft, ob die Anwendung in angemessener Zeit auf Nutzungsaktionen reagiert. Grundsätzlich erwarten UserInnen ein Feedback seitens der Applikation in unter einer Sekunde. Steigt jedoch die Komplexität der Interaktion, kann diese, für die/den NutzerIn, akzeptable Dauer auch steigen.(Matam & Jain, 2017)

3.4 Time-To-Market

Unter dem Begriff Time-To-Market versteht man den kompletten Prozess bevor das Produkt bzw. die Anwendung auf den Markt kommt. Dies beinhaltet Punkte wie Ideengenerierung, das Gestalten des Designs, das Entwickeln sowie das Testen des Produkts. (Gonçalves, 2019)

Der Entwicklungsaufwand bzw. die Entwicklungsdauer steht in starker Abhängigkeit zu der Komplexität der gewählten Technologie. Wendet man sich diesem Thema zu, sollte genau analysiert werden, welchen Funktionsumfang die zu entwickelnde Applikation besitzen soll. Werden unnötig komplexe, wenig dokumentierte oder neuartige Technologien verwendet, wird sich der Markteintritt verzögern. Viele Technologiestacks verwenden innerhalb des Front-Ends und des Back-Ends zwar unterschiedliche Programmiersprachen, doch existieren immer Teile welche client- sowie serverseitig verwendet werden können. Frameworks, welche bereits einen fertigen Technologiestack bereitstellen (beispielsweise JHipster wie in Kapitel 4.5 beschrie-

ben) verringern die Zeit bis zum Markteintritt rapide. (Nowak, 2019)

Auch wenn die genaue Abschätzung der Dauer bis zur Markteinführung nahezu unmöglich ist, kann man durch gezielte Analyse und Recherche eine grobe Abschätzung abgeben. Die Schätzung des Aufwandes für die Implementierung der jeweiligen Technologie erfolgt mittels Durchsicht der Dokumentation, Analyse der Größe der Community für etwaige Hilfestellungen und die Einschätzung der tatsächlichen Komplexität der Verwendung und Implementierung des Frameworks. Eine aktive und große Community bietet auch den Vorteil, dass die Wahrscheinlichkeit für bereits fertige Programmteile eine höhere ist, als bei unbekannten, nicht weit verbreiteten Frameworks.
(Bulatovych, 2019)

Zu Beginn steht die individuelle Betrachtung der Frameworks im Vordergrund, doch sollte immer das System als Ganzes beleuchtet werden. Hiermit ist gemeint, ob ein vorerst einfach zu implementierendes Framework plötzlich komplex zu implementieren wird, da es mit den Technologien aus den anderen Schichten nur begrenzt kompatibel ist. Solche Stolpersteine würden den Markteintritt drastisch verzögern, vor allem wenn solche Erkenntnisse erst gegen Ende der Entwicklungsphase gewonnen werden.
Subjektive Faktoren wie Knowhow oder Vorlieben der EntwicklerInnen werden nicht berücksichtigt.

3.5 Stabilität und Verfügbarkeit

Die Verfügbarkeit entscheidet darüber, ob UserInnen überhaupt mit der Anwendung arbeiten können oder nicht. Deshalb ist bei der Entwicklung darauf zu achten, dass die Applikation stabil läuft und wenig bis gar keine Ausfälle zu verzeichnen hat.
Bei der Wahl der Technologien ist in Bezug auf Verfügbarkeit das Augenmerk darauf zu legen, dass die Zeit vom Eintritt des Ausfalls bis hin zur Lösung des Problems so minimal wie möglich gehalten wird. Eine Möglichkeit der Vermeidung von Ausfällen ist, Fallback-Komponenten einzurichten, auf welche das System gegebenenfalls zurückgreifen kann, sollten die Hauptkomponenten ausfallen. (Franz, 2007)

Auch wenn EntwicklerInnen bei der Programmierung der Applikation noch so sen-

sibel auf die Stabilität des Systems geachtet haben, wird es gerade bei steigender Nutzerzahl einen Overload an Anfragen geben, welche die Anwendung bzw. die Server im Hintergrund im schlimmsten Fall zum Absturz bringen.
Um solche Fälle zu testen, empfehlen sich die Tests, welche im Kapitel 3.3 genauer beschrieben worden sind. Bringen die Tests die Anwendung tatsächlich zum Absturz, muss herausgefunden werden, wo sich das Bottleneck befindet. Diese Bottleneck können in vier Kategorien unterteilt werden.

- Disk I/O, z.B. aufwändige Datenbankabfragen oder lokaler Festplattenzugriff
- Netzwerk I/O, tritt bei Web-Anwendungen auf wenn beispielsweise externe API Aufrufe nötig sind
- CPU, ineffektiver Code welcher eine hohe Rechenleistung benötigt
- RAM, tritt auf wenn das System nicht ausreichend Arbeitsspeicher hat

(Jin, Sahni & Shevat, 2018)

3.6 Austauschbarkeit

Wechselende Anforderungen seitens der KundInnen, Neuerungen innerhalb der Applikation, modernere Technologien oder Sicherheitslücken bei den verwendeten Frameworks können Erklärungen dafür sein, dass einzelne Komponenten bzw. Technologien ausgetauscht werden.

Ein Technologiewechsel kommt innerhalb einer Anwendung in den meisten Fällen früher oder später vor, daher sollte schon bei der Implementierung der einzelnen Frameworks daran gedacht werden, ob und wie einfach man diese ausbauen und mit einer anderen Technologie ersetzen kann. (Jablonski, Petrov, Meiler & Mayer, 2004)

Um einen reibungslose Technologiewechsel gewähren zu können, sollten die einzelnen Frameworks in optimaler Weise nicht global innerhalb der ganzen Software eingeklinkt werden, sprich Konfigurationsdateien sollten zentralisiert implementiert werden, um nötige Konfigurationseinstellungen schnell auffindbar zu machen und

gegebenenfalls umzuschreiben. Mit diesem Wissen im Hintergrund ist es empfehlenswert die Applikation granular aufzubauen, das bedeutet, dass alles in so kleinen Bereichen wie möglich entwickelt werden soll. Dies betrifft nicht nur die Trennung von Front-End und Back-End, sondern auch die Services die innerhalb der einzelnen Schichten Anwendung finden. (Neuhaus, 2017)

Unter der Berücksichtigung eines potentiellen zukünftigen Wechsels eines Frameworks, wird es empfehlenswert sein, schon bei der Erstentwicklung der Applikation und der Implementierung der einzelnen Komponenten (Front-End, Back-End, Datenbank) darauf zu achten, dass nicht die komplette Logik der anderen Layer komplett umgeschrieben werden muss.

3.7 Zukunftssicherheit

Speziell in der Web-Entwicklung werden früher oder später neue, modernere und vielleicht für die entwickelte Web-Applikation besser geeignete Technologien auf den Markt kommen. Trotz dieses Umstandes sollte diese vor der Entscheidung bestimmte Frameworks zu verwenden, auf deren Zukunftssicherheit bzw. Weiterentwicklung geprüft werden. Kontrolliert man die Website des Frameworks oder das Repository und sieht, dass die letzten Updates weit in der Vergangenheit liegen, kann darauf geschlossen werden, dass dieses Framework eher eine geringe Chance hat, in Zukunft noch up to date zu sein.

Um gewählte Frameworks auch zukünftig für Featureerweiterungen in der Applikation zu verwenden, sollte auch Rücksicht darauf genommen werden, dass die EntwicklerInnen an die gewählte Technologie glauben. Relativ neuen Technologien, welche gerade einen Hype erleben und eine unsichere Zukunft vor sich haben, sollten mit Vorsicht innerhalb der Anwendung implementiert werden. Bestenfalls ist die Architektur der Applikation modular, was heißt, dass einzelne Teile leicht ersetzt werden können. (Council, 2016)

Bestenfalls fällt die Entscheidung auf eine Technologie, welche von den EntwicklerInnen durchgängig mit Updates versorgt wird und mit den neusten Möglichkeiten und Funktionen versehen wird, wobei zu erwähnen ist, dass etablierte Frameworks mithilfe von vorangegangen Weiterentwicklungen ziemlich einfach auf deren Zukunfts-

sicherheit überprüft werden können. Speziell gänzlich neue Technologien sind in der Welt des Webs, in welcher Browser als Client unausweichlich sind, oftmals problematisch, da Browser die Funktionalitäten dieser Technologien meist nicht unterstützen.

4 Technologien

Um eine Webanwendung erfolgreich entwickeln zu können, werden drei große, gänzlich unterschiedliche Bereiche benötigt. Hierbei handelt es sich um Front-End, Back-End sowie um etwaige Datenbanken (siehe 3.1). Front-End beinhaltet jegliche Technologien, welche im Browser laufen und mittels welchen die GUI für die/den UserIn bereitgestellt wird. Neben den bekannten, aber statischen Front-End Technologien wie HTML oder CSS, fällt auch JavaScript in die Begrifflichkeiten der front-end Technologien und ermöglicht es auch direkt im Browser, ohne jegliche Serverkommunikation, eine dynamische Seite zu gestalten. (Mardan, 2015)

Für Datenverarbeitung bzw. die Kommunikation mit der Datenbank, also die Anwendungslogik, ist es nötig, mittels back-end Entwicklung entsprechende Controller zu implementieren. Mittels dieser Technologien können Anfragen abgearbeitet werden, benötigte Informationen aus der Datenbank ausgelesen sowie hinzugefügt werden und am Ende sendet die back-end Technologie die angefragten Daten als Antwort wieder zurück zu der front-end Technologie. (Soni, 2017)

Schon bei den simpelsten Web-Applikationen wird es unumgänglich sein, ein Datenbanksystem im Hintergrund laufen zu haben, um Informationen speichern zu können. Bei gespeicherten Informationen kann es sich um einfache Anmeldetaten, aber auch um komplette Produktkataloge handeln. (Northwood, 2018)

4.1 Programmiersprachen

Die Wahl der Technologie geht auch meist Hand in Hand mit der Programmiersprache, auf welche das gewünschte Framework aufgebaut ist. Fällt die Entscheidung pro

Vue.js aus - beispielsweise als das front-end Framework - so wird ein/eine EntwicklerIn mit Vorwissen in JavaScript bei dem Umgang mit dieser Technologie Vorteile haben. Auch Technologien wie Angular oder React gehören zur Familie der auf JavaScript basierenden Frameworks.

Sind die clientseitigen Technologien meist auf JavaScript aufbauend, gibt es bei der serverseitigen Technologiefindung schon eine größere Anzahl an möglichen Programmiersprachen mit unterschiedlichen Ansätzen und Syntaxen. Obwohl in dieser Arbeit speziell auf Java, bzw. PHP als back-end Technologien eingegangen wird, ist zu erwähnen, dass auf dem Markt noch weitere Programmiersprachen - wie beispielsweise .Net - existieren.

Befindet man sich innerhalb der Datenbank und möchte Daten aus Tabellen auslesen, hinzufügen, löschen oder ändern, wird der Einsatz von SQL als Entwicklungssprache notwendig. Aufbauend auf SQL haben sich diverse, leicht veränderte, Datenbanktechnologien entwickelt. Diese Arbeit legt das Hauptaugenmerk auf die weitverbreitete Open Source Variante MySQL, das kommerzielle Oracle Datenbanksystem sowie die auf NoSQL basierte Datenbank MongoDB.

4.1.1 Javascript

JavaScript wurde im Jahr 1995, zu Beginn unter den Namen Mocha bzw. LiveScript, entwickelt und letztendlich unter dem heute bekannten Namen veröffentlicht. Auch wenn der Namensteil Java auf die bekannte Programmiersprache schließen lassen würde, ist JavaScript in keiner Form mit Java verwandt. JavaScript hat Gemeinsamkeiten mit prototypbasierenden Programmiersprachen wie beispielsweise Self. (Brown, 2016)

Interagiert eine Website mit der/dem NutzerIn, egal ob in Form eines Pop-Ups, eines neuen Textes oder einer Änderung der Farbe, bedient sich diese Website der Technologie von JavaScript. Der Grund weshalb mittels JavaScript dynamische Webinhalte kreiert werden können ist, dass diese Technologie innerhalb des Browsers läuft und somit direkten Zugriff auf das DOM hat. Durch die Einführung des DOM innerhalb von HTML wurde es für JavaScript-EntwicklerInnen viel einfacher, einzelne Teile einer Website dynamisch zu gestalten, da die HTML durch das DOM eine for-

male Struktur erhielten.

Bei JavaScript handelt es sich um eine clientseitige Programmiersprache, was bedeutet, dass das komplette JavaScript Script innerhalb des Browsers läuft. Um JavaScript in ein HTML Dokument einzubinden, reicht es, mittels `<script> </script>` einen JavaScript Block zu definieren. (Nixon, 2018)

```
1 <html>
2   <head><title>JavaScript Example</title></head>
3   <body>
4   <script type="text/Javascript">
5     //Simples schreiben von "Hello World" auf die Website
6     document.write("Hello World");
7   </script>
8   </body>
9 </html>
```

Listing 4.1: JavaScript Example

JavaScript gehört zu der Kategorie der objektorientierten Programmiersprachen, wobei sich JavaScript von anderen dieser Sprachen unterscheidet. Klassen in JavaScript basieren auf dem JavaScript internen Prototyp Objekt; alle Objekte erben Methoden bzw. Attribute von diesem Objekt.
Beispielsweise kann ein Datumsobjekt, welches mittels `new Date()` aufgerufen wird, jegliche Eigenschaften von dem `Date.prototype` Objekt erben. (Flanagan, 2011)

4.1.2 Java

Bei Java handelt es sich um eine objektorientierte und plattformunabhängige Programmiersprache, welche von Sun Microsystems entwickelt wurde und 2010 von Oracle gekauft wurde.
Der Slogan, welcher im Bezug mit Java oft fällt, ist:

Write once – run anywhere

Diese Aussage bezieht sich auf eine der größten Stärken von Java und zwar auf die Plattformunabhängigkeit. Grundsätzlich sind kompilierte Programme meist eng mit spezifischer Hardware oder Software verknüpft. Grund für die Plattformunabhängigkeit von Java ist, dass der Java-Compiler den geschriebenen Code in sogenannten Bytecode umwandelt. Dieser kompilierte Code kann dann auf jeder Hardware bzw. auf jedem Betriebssystem, auf welchem ein Java-Interpreter läuft, ausgeführt werden. (Fischer, Saddik & Steinacker, 2013)

Um ein lauffähiges Java-Programm zu entwickeln, müssen folgende Punkte berücksichtig werden:

- Java-Anwendungen müssen immer aus Klassen bestehen.
- Klassenname und Dateinamen müssen immer übereinstimmen, auch ist die Namensgebung case-sensitive.
- Startpunkt eines jedes Java-Programmes ist die Main Methode, welche öffentlich und statisch definiert sein muss.

Erzeugte Java-Dateien müssen für die Ausführung an den Java-Compiler zum Kompilieren übergeben werden. Der Compiler oder auch javac erzeugt aus der *.java Datei eine *.class Datei, welche einen kompilierten Java-Bytecode enthält. Diese vom Compiler erzeugte Datei kann sodann mittels Java-Interpreter plattformunabhängig ausgeführt werden (Kofler, 2013)

```
1 public class HelloWorld {
2   public static void main(String[] args) {
3     System.out.println("Hello World");
4   }
5 }
```

Listing 4.2: Java Example

Um die Kommunikation zwischen Präsentations-Layer und der Datenbank zu gewährleisten, wird serverseitig ein sogenannter Controller zwischengeschalten. Um dies mit Java handhaben zu können, bietet Java sogenannte Servlets an.

4.1.2.1 Servlets

Servlets sind Java-Klassen, die auf eingehende HTTP Anfragen reagieren, welche der Browser durch Benutzeraktionen an den Server sendet. Eine solche Anfrage kann eine simple Datenbankabfrage sein, welche mittels POST an den Server gesendet wird. Innerhalb des Servlet wird dann das entsprechende Event ausgelöst, um mit der Datenbank zu kommunizieren und wenn nötig eine Antwort an den Browser zurückzusenden. (Wolf, Henley & Henley, 2017)

4.1.3 SQL

Structured Query Language (SQL) macht es - wie der Name bereits erahnen lässt - möglich, strukturierte Datenbankabfragen auszuführen. Wird SQL häufig mit relationalen Datenbanken (siehe 2.4.2) in Verbindung gebracht, hat SQL nichts damit zu tun wie die Datenbank technisch realisiert ist; hier handelt es sich rein um die Sprache mit welcher innerhalb der Datenbank gearbeitet wird.

Hauptaufgabe von SQL ist es, innerhalb der Datenbank bereits existierende Daten zu verändern, zu entfernen oder gänzlich neue Daten in die Datenbank zu schreiben. Unterschieden wird bei den Befehlen, welche innerhalb von SQL existieren, zwischen Kommandos zur Datenbearbeitung (Data Manipulation Language (DML)) und Befehlen, mit denen man das Design ändern oder definieren kann (Data Definition Language (DDL)).

In die Kategorie von DML fallen Ausdrücke wie `SELECT` für die Abfrage, `INSERT` für das Hinzufügen von neuen Datensätzen, `Update` für das Verändern von bestehenden Datensätzen und `DELTE` für das Löschen von Datensätzen.
Unter DDL finden sich Ausdrücke wie `CREATE` zum Erstellen von Datenbanken oder Tabellen, `ALTER` wird benötigt um Struktur und/oder Eigenschaften von Tabellen und Datenbanken zu verändern und mittels `DROP` können Datenbanken sowie Tabellen entfernt werden. (Throll & Bartosch, 2011)

```
1   --Select all
2   SELECT * FROM table_name;
3   --Delete DB entry with id 1
4   DELETE * FROM table_name WHERE id=1;
5   --Add new column
6   ALTER TABLE table_name ADD column_name type;
7 }
```

Listing 4.3: SQL Example

4.2 Clientframeworks

Im Zuge der Entwicklung des Front-End muss seitens der EntwicklerInnen beachtet werden, dass als Ausführungsplattform der Browser herangezogen wird. Damit ist gemeint, dass der entwickelte Code auf jeglichen Browsern, egal ob Desktop-Computer, Smartphone oder Tablet, lauffähig und bedienbar sein muss. Zusätzlich wird im Rahmen der Front-End Entwicklung ein breites Spektrum an Verantwortlichkeiten abgedeckt. Auf der einen Seite muss das Design der GUI stimmig sowie benutzerfreundlich sein und auf der anderen Seite muss auch die Logik für jeglichen dynamischen Inhalt geschrieben werden. (Aquino & Gandee, 2016)

4.2.1 Vue.js

Für eine moderne Webanwendung ist es unumgänglich, eine interaktive Seite zu gestalten. Für kleine dynamische Textänderungen reicht meist eine JavaScript Bibliothek, wie beispielsweise JQuery aus. Wird jedoch die front-end Entwicklung komplexer, das heißt, ein größerer Bereich der Seite soll dynamisch werden, soll client-side routing stattfinden oder einfach größere Mengen ans JavaScript Code verwaltet werden, bieten sich JavaScript Frameworks an. Eines jener Frameworks, welches zur Zeit relativ neu und populär auf dem Markt ist, ist Vue.js.

Anfang 2014 wurde Vue.js von Evan You, einem Entwickler von Google, veröffentlicht und erfreut sich seither enormer Popularität. Einer der Gründe für die Beliebtheit dieses Frameworks ist, dass eine performante Templatestruktur gegeben ist, wel-

che es ermöglicht, reaktiv in das DOM zu schreiben und auf Events zu hören. Das heißt, dass das Template nach Änderungen nicht neu geladen werden muss um die neuen Daten dazustellen. Trotz des breiten Spektrums an Funktionalitäten, welche in Vue.js bereits integriert sind, existieren Bibliotheken, welche kompatibel mit Vue sind und somit lässt sich der Funktionsumfang optional erweitern. (Macrae, 2018)

Weiters handelt es sich bei Vue.js um ein progressives Framework, was den Vorteil hat, dass man, wenn man mit einer relativ einfachen Applikation startet, diese auch zu einer weitaus komplexeren und umfangreicheren Anwendung weiterentwickeln kann.
Vue lässt sich relativ simpel mittels `npm install vue` installieren und mithilfe von `npm install vue-clie` wird das Command Line Interface nachinstalliert, mit welchem ohne großen Aufwand über die Command Line ein Vue.js Projekt aufgesetzt werden kann. Mittels `vue init webpack projektname` wird ein lauffähiges Vue.js Projekt initialisiert.

Features können ebenfalls direkt bei der erstmaligen Initialisierung mitinstalliert werden; diese werden aufeinanderfolgend während des Installationsprozesses abgefragt. Bei diesen Erweiterungen handelt es sich exemplarisch um den Vue-Router, welcher bei Single Page Applications zur Anwendung kommt. Wird diese Option gewählt, werden alle nötigen Anfragen gleichzeitig während des Ladens der Seite an den Server gesendet. ESLint ist ein Linter Tool von JavaScript und kann zur Codekontrolle verwendet werden, die Option Setup test erstellt eine Testumgebung inklusive der benötigten Konfiguration und Struktur. (Ye, 2018)

Anhand dieser optionalen Erweiterungen bei der Installation ist zu erkennen, wie simpel ein Vue.js Projekt aufgesetzt werden kann, welches schon eine lauffähige Testumgebung und coding standards beinhaltet.

```
<div id="hello-world"> wird ersetzt </div>

<scrip>
  new Vue({
   el: '#hello-world',
    data: {
     message: 'Hello Template!'
    },
  template: '<div>{{ propertyName }}</div>'
   })
</script>
```

Listing 4.4: Vue.js Example

Wie in Listing 4.4 zu sehen ist, wird mittels `new Vue()` eine neue Instanz von Vue.js erstellt, wobei mit der Eigenschaft `el` angegeben wird, auf welches Objekt das Objekt gebunden werden soll. Durch die `data` Eigenschaft werden, wie sich aus dem Name schließen lässt, die Daten an das Template übergeben; dieser Name des Vue-Objekt muss dem im Template identisch sein. Wird ein Vue.js Objekt auf ein HTML-Element gebunden, wird das DOM mit dem Vue DOM ausgetauscht, welches den Inhalt der in der zuvor definierten Funktion beinhaltet.

Eine der Optionen, welche die Vue.js Instanzen besitzen, ist die Template Option. Diese wird von Vue verwendet, um das Vue DOM zu generieren, welches das originale DOM Element, das mittels `el` ausgewählt wurde, ersetzt. Hierfür muss ein root Element existieren, dessen Inhalt gänzlich durch den in der Vue-Instanz generierten Inhalt ersetzt wird. In Listing 4.4 handelt es sich bei `<div id="hello-world»` um das root Element. (Nelson, 2018)

4.2.2 React

Um die Herausforderungen zu meistern, welche mit großen datengesteuert Webseiten auftreten können, hat Facebook im Jahr 2013 React entwickelt. Bei React handelt es sich um eine leichtgewichtige JavaScript Library, die gerade zu Beginn durch ihren einzigartigen Programmierstil eher kritisch gesehen wurde.
Der Programmcode, welcher in React geschrieben wird, ist HTML sehr ähnlich, wird

aber direkt in JavaScript geschrieben und benötigt ein build tool, wie beispielsweise Webpack, um den Code für den Browser lauffähig zu machen. Anders als beispielsweise Vue.js, bietet React bei weitem kein so breites Spektrum an Funktionalitäten an, um diverse Features mit React umzusetzen.
Ein weiterer Unterschied von React zum klassischen JavaScript ist, dass sich React eher auf funktionales Programmieren fokussiert und weniger auf objektorientiertes Programmieren; dies bringt Vorteile im Bereich von Performance und Testing. (Banks, 2017)

Hauptbestandteil von React ist der virtuelle DOM, welcher verwendet wird, um client- sowie serverseitige Kommunikation durchzuführen. Um die Performance auch bei komplexen Systemen hochzuhalten, wird bei React darauf geachtet, dass nur die nötigsten DOM Manipulationen durchgeführt werden.

Der virtuelle DOM von React ist ident mit dem realen DOM. Dieser ist ebenfalls als Baumstruktur aufgebaut und enthält Elemente inklusive Attribute und Inhalte als Objekte bzw. Eigenschaften. Wird mittles `render()` eine Änderung innerhalb eines solchen Knotens durchgeführt, wird ein neuer Knoten mit den Neuerungen erstellt und der/dem UserIn präsentiert. (Kumar & Singh, 2015)

React lässt sich über die Kommandozeile mittels `npm install -g create-react-app` initialisieren und mit dem Befehl `npm install -g create-react-app project-name` kann ein neues React Projekt aufgesetzt werden.

```
1 import React from 'react';
2
3 class Hello_world extends React.Component {
4   render() {
5     return <div>Hello World</div>;
6   }
7 }
8 export default Hello_world;
```

Listing 4.5: React Example

Wie in Listing 4.5 zu sehen ist, wird mittels `render()` eine visuelle Ausgabe an ein HTML zurückgegeben. Ebenfalls wird anhand des Codebeispieles recht deut-

lich die Ähnlichkeit zu nativen HTML sichtbar. Diese Vereinfachung der Syntax für das Schreiben von HTML innerhalb von JavaScript ist dank Javascript XML oder Javascript Syntax Extension (JSX) möglich. JSX ermöglicht es der/dem EntwicklerIn HTML-Code innerhalb von JavaScript nicht mehr mit `document.write()` schreiben zu müssen, sondern wie im Codebeispiel 4.5 zu sehen ist, ist nun nativer HTML Code ausführbar. Weiters wird mittels Präprozessors dieser JSX in simplen JavaScript Code umgewandelt, um diesen auch im Browser darstellen zu können. Bei der Verwendung von JSX ist allerdings darauf zu achten, dass der Returnwert tatsächlich auch nur ein Hauptelement beinhaltet, das heißt, es ist nicht möglich zwei `<div>` Tags, welche nicht ineinander verschachtelt sind, innerhalb der gleichen Renderfunktion zurückzugeben. (Chiarelli, 2018)

4.2.3 Angular

Bei Angular handelt es sich um ein Open-Source Framework, welches von Google auf Basis von JavaScript entwickelt wurde und in allen gängigen Browsern läuft. Durch die Verwendung von Typescript laufen Angular-Applikationen auch auf veralteten Browsern, wie Internet Explorer. (Uluca, 2018)

Mit der steigenden Erwartungshaltung an Web-Anwendungen - im Webbrowser sowie an mobilen Endgeräten - hat sich auch die Komplexität für die Entwicklung solcher Anwendungen gesteigert. Reichte es zu den Anfängen des Internets einen statischen Webauftritt zu haben, sollten heute die Websites dynamisch aufgebaut sein und nativen Desktop-Applikationen ähneln. Innerhalb von modernen Webseiten erfreuen sich Single Page Applications an immer größer werdender Beliebtheit.

Solch eine Architektur hat den Vorteil, dass bei Änderungen nicht die ganze Seite, sondern nur der sich veränderte Teil der Seite neu geladen werden muss. Um eben solche Single Page Applications optimal bearbeiten zu können, wurde Angular entwickelt; hat es zu den Anfangszeiten tatsächlich das Hauptaugenmerk auf rein auf solche Single Page Applications gelegt, kann Angular heutzutage auch für aktuelle Trends wie Web Komponente, Dependency Injection oder Templating genutzt werden.
Angular wird mehrheitlich im Kontext von komplexen und umfangreichen Projekten genutzt. Um solche Großprojekte trotzdem wartbar zu halten, bietet Angular den

EntwicklerInnen einige Features um dies zu gewährleisten.

Zu diesen Hilfsfunktionen zählen Benutzerkomponenten, welche mit Funktionalitäten versehen und relativ simpel mit Web-Komponenten verknüpft werden können. Mittels Data Binding können Informationen aus dem JavaScript Code einfach in die View übergeben werden. Testen steht bei Angular hoch im Kurs und wird von Beginn an berücksichtigt, was die Möglichkeit bietet, beinahe jeden Aspekt der Applikation zu testen. Innerhalb des mitgelieferten Funktionspaketes befinden sich auch schon Lösungen für Serverkommunikation oder routing innerhalb der Anwendung. (Seshadri, 2018)

Wie schon Vue.js (4.2.1) und React (4.2.2), lässt sich, nach der Installation von Bower, Angular über die Kommandozeile durch den Befehl `bower install angular` installieren. Danach lassen sich mittels Bower einfach neue Bibliotheken in das Projekt integrieren.

```
<div ng-app="helloWorld">
 <div ng-controller="HelloWorldCtrl">
  {{message}}
 </div>
</div>
<script>
 var app = angular.module("helloWorld", []);
  app.controller("HelloWorldCtrl", function($scope,$http) {
   $scope.message="Hello World"
  })
</script>
```

Listing 4.6: Angular Example

Module, welche wiederverwendbaren Programmcode sowie Konfigurationsinformationen beinhalten, bilden die Basis für Angular. Durch `module` wird ein Objekt zurückgegeben, welches das jeweilige Modul beschreibt, während in Listing 4.6 die Objektfunktion `controller` aufgerufen wird, um einen neuen Controller zu erzeugen. Der in die Funktion übergebene Parameter `$scope` bezieht sich auf ein Objekt, das wiederum Variablen enthält, die innerhalb der Applikation in Verwendung sind. Bei dem Codebeispiel 4.6 wird dem `$scope` Objekt eine neue Eigenschaft `message` an-

gefügt, welche später dynamisch an die View weitergegeben werden kann. (Steyer & Softic, 2015)

4.3 Datenbanken

Für die Speicherung und Verwaltung der Daten innerhalb von Anwendungen läuft im Hintergrund immer ein Datenbanksystem. Aufgebaut sind Daten als Sammlung von strukturierten Datensätzen, deren Anzahl und Größe willkürlich variieren können, während jeder Datensatz wiederum spezielle Informationen über ein Objekt, z.B. einen Artikel in einem Online Shop, beinhaltet. (Preiß, 2014)
In diesem Kapitel werden mit und PostgreSQL zwei relationale Datenbanken sowie mit MongoDB eine NoSQL Datenbank genauer beleuchtet.

4.3.1 MySQL

MySQL ist eine relationale Datenbank (siehe 2.4.2), deren Beliebtheit in den frühen 2000er Jahren immer mehr zunahm und daraus resultierend im Jahr 2009 von Oracle aufgekauft wurde.

Abgesehen davon, dass MySQL größtenteils kostenlos genutzt werden kann, spielen auch die vielen Features, welche MySQL bietet, eine große Rolle, weshalb sich diese Datenbank an einer solchen Beliebtheit erfreut. Eine der größten Stärken seitens MySQL ist die Flexibilität oder auch die Plattformunabhängigkeit. MySQL ist auf allen großen Betriebssystemen lauffähig und bietet zusätzlich, sollte ein Operating System (OS) nicht verfügbar sein, die Binaries an, um es auf dem gewünschten System lauffähig zu machen. Performance stellt schon seit Beginn einen zentralen Punkt bei MySQL dar. Während früher dafür auf gängige Datenbankfunktionalitäten (z.B. Trigger, Stored Procedures) verzichtet wurde, sind in den aktuellen Versionen auch diese Features zu finden.

Einer der Gründe weshalb MySQL eine relativ performante Datenbank ist, ist das Query Caching, was bedeutet, dass MySQL `SELECT` Abfragen inklusive des Ergebnisses zwischengespeichert, diese mit nachfolgenden Abfragen vergleicht und wenn

sich die `SELECT` Abfragen decken, wird diese nicht ausgeführt, sondern das Ergebnis aus dem Cache verwendet.

Weitere nicht zu vernachlässigende Punkte, weshalb MySQL immer eine Option darstellen sollte ist, dass es eine riesige und sehr aktive Community gibt, jegliche Probleme meist gelöst werden können und die Weiterentwicklung wird in naher Zukunft auch nicht eingestellt werden.

Für die Installation stellt MySQL fertige Installationspakte zur Verfügung, welche die/den UserIn durch das Setup führen. (Gilmore, 2010)

Die Syntax von MySQL ist der von SQL sehr ähnlich, deshalb kann als Codebeispiel 4.3 herangezogen werden.
Reicht bei wachsenden Datenmengen die Geschwindigkeit von Datenbankabfragen nicht mehr aus, bietet MySQL Indexes an. Indexes werden auf Datenbankspalten gebunden, welche im Kontext von möglichen Suchanfragen relevant sind. Unterschieden wird zwischen drei Arten von Indexes, nämlich Primary Key, Index oder Fulltext. Wird ein Index mittels `INDEX` auf eine Spalte gebunden, wird eine Zahl als Parameter mitübergeben (in Codebeispiel 4.7 Integer 20), um zu definieren, wie viele Zeichen der Index maximal beinhalten soll; erst wenn mehrere identische Datensätze gefunden werden, sucht MySQL in der Datenbank nach dem einen angeforderten Datensatz.

Primary Keys werden genutzt, um Tupel innerhalb einer Tabelle einzigartig zu machen, das heißt jeder Wert darf in einer Primary Key Spalte nur einmal vorkommen. Mittels `FULLTEXT` wird ein spezieller Index auf Spalten gebunden, um sehr performante Suchen durchführen zu können. Wie in Listing 4.7 Zeile 2 zu sehen ist, wird ein Fulltext Index auf 2 Spalten gebunden; dies ermöglicht es eine Suche über beide Spalten durchzuführen. (Nixon, 2018)

```
1   ALTER TABLE table_name ADD INDEX(column_name('20'));
2   ALTER TABLE table_name ADD FULLTEXT(column1, column2);
3 }
```

Listing 4.7: MySQL Index Example

4.3.2 PostgreSQL

Bei PostgreSQL handelt es sich um ein erweitertes relationales Datenbanksystem, welches gänzlich unter der Open-Source-Lizenz läuft. Gestartet als Forschungsprojekt an der University of California at Berkeley 1996, wird es neben der PostgreSQL Community nach wie vor von diversen Universitäten weiterentwickelt.

Vorteile, die für einen Einsatz von PostgreSQL sprechen, sind unter anderem die Verwendung der gängigen SQL Standard Syntax. Das heißt, EntwicklerInnen, welche mit PostgreSQL arbeiten können, sind leicht zu finden. (Obe & Hsu, 2012)

Zusätzlich zur Plattformunabhängig bietet PostgreSQL für die meisten modernen Programmiersprachen Drivers an; dadurch kann es in die meisten Technologie-Stacks eingebunden werden. Neben einer guten Skalierbarkeit und Performance ist PostgreSQL relativ ausfallsicher, was gerade bei Systemen, bei welchen eine Hochverfügbarkeit gegeben sein muss, ein großes Plus ist. Aufgrund einer relativ großen Community im Hintergrund, wird PostgreSQL regelmäßig mit Updates versorgt, es existiert eine sehr gute Dokumentation und für Spezialfälle ist der Code frei zugänglich und kann den eigenen Bedürfnissen angepasst werden.
Grob können die Domänen, in welchen PostgreSQL eingesetzt wird, in zwei Gruppen unterteilt werden:

- Online Transactional Processing (OLTP) wird bei Applikationen genutzt, bei denen große Mengen an `insert`, `update` sowie `delete` Operationen in rapider Geschwindigkeit durchgeführt werden müssen.

- Online Analytical Processing (OLAP) wird bei Anwendungen verwendet, welche wenige, aber dafür komplexe Datenbankabfragen benötigen, die große Datenmengen aus verschiedene Datenquellen benötigen; als Beispiel kann hier Data Mining aufgeführt werden.

(Juba & Volkov, 2017)

Wie auch MySQL (siehe Kapitel 4.3.1), verwendet PostgreSQL für die Geschwindigkeitsoptimierung Caching das heißt, wenn nacheinander Abfragen ausgeführt werden, kontrolliert PostgreSQL, ob Änderungen stattgefunden haben. Ist dies nicht der

Fall, wird das im Cache gespeicherte Resultat verwendet, ohne die Abfrage redundant auszuführen.

Zusätzlich zum Caching verwendet PostgreSQL Indexes, um bei Abfragen die Performance zu verbessern. Indextypen von PostgreSQL sind Primary Keys oder Unique Keys, mit deren Hilfe diese einzelnen Datensätze einfach und schnell gefunden werden. Block range index (BRIN) kommt zum Einsatz, wenn große Tabellen genutzt werden, in welcher Primary Keys zu groß werden würden oder das Speicherlimit überschreiten würden; BRIN behandelt einen Block als eine gemeinsame Einheit und versieht alle darin enthaltenen Tupel mit einem Index. Generalized search tree (GiST) wird verwendet, wenn eine Volltextsuche gefordert ist oder wissenschaftliche, unstrukturierte oder hierarchische Daten gegeben sind. (Regina Obe, 2017)

4.3.3 MongoDB

MongoDB (abgeleitet vom engl. hu**mongo**us, gigantisch), stellt eine neue Art von Datenbank dar, die sich von den klassischen Datenbankkonzepten wie Tabellen, Schemas, SQL oder Zeilen löst. Weitere Standards wie Transaktionen, Joins oder Foreign Keys finden innerhalb von MongoDB ebenfalls keine Verwendung. Solche Datenbanksysteme werden nicht als Relationale Datenbanksysteme bezeichnet, sondern als NoSQL Datenbanken oder dokumentenorientierte Datenbank; die Unterschiede werden in Kapitel 2.4.2 genauer beschrieben. (Schildgen, 2016)

Einer der größten Unterschiede in der Philosophie bei der Verwendung von MongoDB im Gegensatz zu klassischen relationalen Datenbanken ist, dass es nicht für jede Art von Applikation geeignet ist. Während Relational Database Management System (RDBMS) als universeller Informationsspeicher verwendet werden kann, legen die EntwicklerInnen von MongoDB den Fokus vermehrt auf die Entwicklung von blitzschnellen, hoch skalierbaren und einfach benutzbaren Datenbanksystemen. Um die gerade erwähnten Vorteile bieten zu können, mussten Einbußen bei anderen Features, wie z.B. bei dem Transaktionshandling, gemacht werden.
Aufgrund dessen kommt MongoDB vermehrt zum Einsatz, wenn komplexe, große Datenmengen verwaltet und analysiert werden müssen; bei Applikationen mit vielen Transaktionen sollte MongoDB außen vor gelassen werden.

Trotz unverkennbarer Ähnlichkeit verwendet MongoDB nicht JSON als Datenspeicher, sondern ein speziell von MongoDB entwickeltes Datenformat Binary JSON (BSON). Dank BSON ermöglicht MongoDB es dem Computer Dokumente noch schneller zu verarbeiten und zu durchsuchen. Weiters bietet BSON zusätzliche Features an, welche im Standard JSON fehlen. (Plugge, Hawkins, Plugge & Hawkins, 2011)

```
1 db.einwohner.find({"personen.name": { $get: "F"} } ))
2 db.COLLECTION_NAME.insert(document)
3 db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA↩
    ) )
4 db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)
5 }
```

Listing 4.8: MongoDB Example

Im Codebeispiel 4.8 kann in der ersten Zeile gesehen werden, dass man mit `find` auf eine Collection (=Datenbank) zugreifen kann und alle Namen mit dem Anfangsbuchstaben “F„ aus dem Subdokument `personen` herausfiltern kann. Anhand des Listings 4.8 kann man relativ gut erkennen, wie MongoDB aufgebaut ist. (Trelle, 2015)

MonogDB Instanzen sind immer nach dem selben Schema aufgebaut. Wie man an der Grafik 4.1 sehen kann, können innerhalb von einer MongoDB mehrere Datenbanken laufen. Diese Datenbanken besitzen im Gegensatz zu relationalen Datenbanken nicht Tabellen, sondern sogenannte `collections`, welche wiederum aus Dokumenten bestehen, die das Äquivalent zu Zeilen darstellen und immer einen speziellen einzigartigen Schlüssel ("_id") besitzen.

Dokumente sind innerhalb einer MongoDB der zentrale Punkt, denn dort sind die Informationen, wie in Listing 4.9 in Form von `keys` und `values` Paaren, gespeichert. Das Schema solcher Dokumente muss innerhalb einer `collections` nicht identisch sein, so können sich, wie in Codebeispiel 4.9, diese beiden Dokumente in der selben `collection` befinden.

```
1 {"foo" : "Hello World"}
2 {"bar" : 42}
```

Listing 4.9: Collection Example

Der Ansatz der dokumentorientierten Datenbanksysteme hat den Vorteil, dass EntwicklerInnen flexibler und einfacher mit der Datenbank arbeiten können. Durch das Ersetzen des Prinzips der Datenzeile durch Dokumente ist es einfacher, komplexe hierarchische Beziehung innerhalb eines Datensatzes abzubilden. Zusätzlich ist diese Art der Datenverwaltung dem modernen Programmieren ähnlicher, auch können ProgrammiererInnen ohne Rücksicht auf Datentypen mit den `keys` und `values` arbeiten. Durch Wegfall eines fixen Schemas im Hintergrund, können Felder seitens der EntwicklerInnen beliebig gelöscht, hinzugefügt und bearbeitet werden. Dies liefert den Vorteil, dass ProgrammiererInnen mit dutzenden Datenmodellen experimentieren können und so die für die Applikation optimalste finden. (Chodorow, 2013)

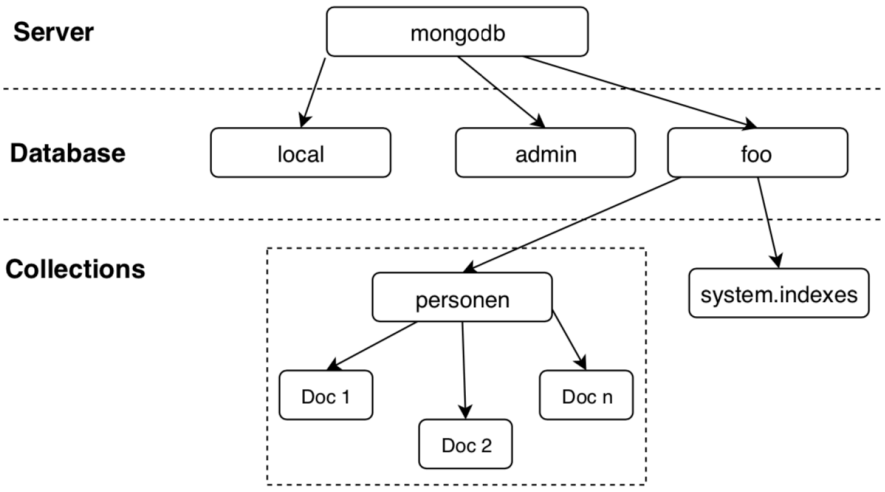


Abbildung 4.1: Aufbau MongoDB
(vgl. Trelle, 2015)

4.4 Serverframeworks

Während im Front-end programmiert wird, was beispielsweise bei dem Klick auf einen Button geschehen soll, wird im Back-end die Logik ausprogrammiert um die

gewünschte Aktion auszuführen bzw. den angeforderten Inhalt der/dem NutzerIn zur Verfügung zu stellen indem mit der Datenbank kommuniziert wird. Die gerade erwähnte Kommunikation mit der Datenbank ist eine der Hauptaufgaben welche im back-end ausgeführt wird, jegliche Anfragen seitens des Front-ends der Anwendung werden über das Back-end an die Datenbank gesendet. Gewisses Grundwissen über die komplette Architektur innerhalb der Applikation muss für das Back-End vorhanden sein, da EntwicklerInnen wissen müssen mit welcher Front-End bzw. Datenbank Technologie kommuniziert werden soll. (Sonmez, 2016)

4.4.1 Hibernate

Bei Hibernate handelt es sich um eine Bibliothek für die Simplifizierung der Verwendung von relationalen Datenbanken innerhalb von Java-Anwendungen, indem Informationen mittels eines `Session-Managers` als ein einfaches Java-Objekt dargestellt werden. Deshalb wird Hibernate auch als Object Relational Mapper (ORM) bezeichnet. Verfügbar sind innerhalb von Hibernate zwei Interfaces, ein `Native-Hibernate` und Java EE Standard Java Persistence API.

Mit Hibernate automatisiert die Abbildung von Datenbanktabellen in Objekten, was auch das Schema der Datenbank beinhalten kann. Es ist nicht nötig pro Tabelle ein Objekt zu kreieren, denn ein Objekt kann zugleich mehrere Tabellen abbilden. Ebenfalls kann Hibernate die Beziehung der einzelnen Datensätze behandeln. Die Zeit in der Hibernate initialisiert wird ist im Gegensatz zu Java Database Connectivity (JDBC) eine höhere, denn Hibernate-Applikationen bieten nach der Initialisierung eine lange Laufzeit sowie eine stärkere Performance. (Ottinger, Linwood, Minter, Linwood & Minter, 2016)

Innerhalb von Hibernate ist zwischen drei verschiedenen Objekttypen zu unterscheiden: temporäre (transient), persistente (persistent), and freistehende (detached) Objekte. Den Unterschied zwischen diesen Objekten zu kennen ist nötig, um das Verhalten von Hibernate besser zu verstehen. Man muss nachvollziehen können, wie Hibernate unterstützend eine API zur Verfügung stellt um Objekte von einer relationalen Datenbank zu löschen, einzufügen, zu empfangen oder zu aktualisieren.

Temporäre Objekte existieren nur im Zwischenspeicher, weshalb Hibernate diese nicht verwaltet. Um solche Objekte in Hibernate trotzdem verwenden zu können, muss über die Session das temporäre Objekt in die Datenbank gespeichert werden, damit Hibernate dieses als ein persistentes Objekt erkennt.

Persistente Objekte existieren hingegen in der Datenbank und werden von Hibernate verwaltet. Wie in Abbildung 4.2 zu sehen ist, hat Hibernate die Aufgabe, jegliche Änderungen an persistenten Objekten und der Datenbank ident zu halten.

Detached Objekte repräsentieren eine Datenbank, wobei sich Änderungen an Objekten aber nicht auf die Datenbank auswirken. Solche Objekte können erstellt werden, indem die Session, mit der das Objekt in Assoziation steht, geschlossen wird. Ein Anwendungsfall für ein solches Objekt ist, dass man Informationen aus der Datenbank ausliest, modifiziert und dieses Ergebnis dann an einer anderen Stelle der Applikation und nicht in der Datenbank gespeichert werden soll. (Minter, Linwood & Ottinger, 2014)

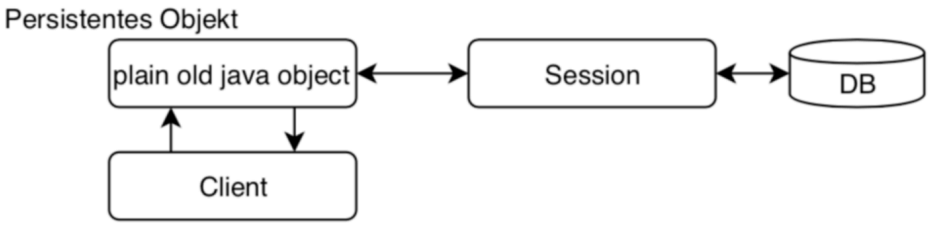


Abbildung 4.2: Persistentes Objekt in Hibernate
(vgl. Minter et al., 2014)

4.4.2 Spring

Hinter dem Namen Spring versteckt sich ein Open Source Java Framework, weshalb der offizielle Name auch Spring Framework lautet. Dieses ermöglicht auf Basis von Java 2 Enterprise Edition (J2EE) relativ simpel Applikationen zu schreiben und findet meist in der Webentwicklung Anwendung. Folgende Prinzipien spielen bei diesem Framework eine wichtige Rolle:

- Dependency Injection; selbstgeschriebener Code bleibt dadurch frei von frameworkspezifischem Code, was bedeutet, dass sich die Anwendung nicht um die

Erstellung von Beziehungen kümmern muss. Dieses Attribut mach den Programmcode sehr gut testbar.

- `ApplicationContext`; wird meist in einer Extensible Markup Language (XML) Datei konfiguriert und verwaltet die Objekterstellung sowie die Dependency Injection; hier wird die Anwendungslogik von der springspezifischen Logik getrennt.

- Plain old Java objects (POJO); hierbei handelt es sich um ein sehr mächtiges Tool innerhalb von Java. Ist die Anwendung stark objektorientiert geschrieben, bleibt ein mittels POJOs geschriebenes Programm gut testbar und flexibel. (Ladd, 2006)

- Aspektorientierte Programmierung (AOP); dient dazu, die Applikation modularer aufbauen zu können. Dies ermöglicht es, Strukturen von Zusammenhängen komponentenübergreifend zu machen (diese Zusammenhänge werden Aspekte genannt), wodurch der Ablauf des Programmes von anderen Aspekten, wie beispielsweise dem Testen sauber getrennt wird.

- Templates; diese werden im Kontext von Spring Klassen bezeichnet, welche für Application Programming Interface (API)s, also Programmschnittstellen, genutzt werden. Diese Klassen bieten Features wie eine einheitliche Fehlerbehandlung oder ein automatisches Ressourcenmanagement.

Durch die bereits erwähnte Modularität von Spring kann das gesamte Framework in sechs große Sparten aufgeteilt werden. Im `Core Container` befinden sich sämtliche zentrale Module wie externe Bibliotheken oder die Dependency Injection. In `AOP` befinden sich die Funktionen für die aspektorientierte Programmierung, während `Messaging` bei Applikationen verwendet wird, die nachrichtenbasiert sind. Mit `Data Access` wird das Datenhandling gesteuert (wie z.B. Datenbankzugriffe) und in `Web` findet man grundlegende Techniken für die Webentwicklung wie beispielsweise das Spring MVC Framework (Kapitel 4.4.2.2) und mittels dem `Test` Modul können alle Komponenten zum Testen der Anwendung integriert werden. (Augsten, 2019)

4.4.2.1 Spring Boot

Spring Boot wird verwendet, um die Erstellung eines Projektes zu vereinfachen. Mittels des Spring Initalizers können Abhängigkeiten untereinander ausgewählt werden, was zur Folge hat, dass eine manuelle Konfiguration innerhalb des Projektes wegfällt. Spring Boot beinhaltet vier Funktionalitäten, um eine solche Initialisierung zu ermöglichen. Durch die `automatische Konfiguration` erkennt Boot eingebundene Libraries und konfiguriert alles soweit automatisch, damit die Bibliothek, ohne XML Konfigurationsdateien oder ähnliches verwendet werden kann.

Mit Hilfe von `Starter Dependencies` bindet Boot zu Beginn grundlegende, für das Projekt nötige, Abhängigkeiten ein; beispielsweise kann für eine Webanwendung das `web` Starterkit verwendet werden (siehe Listing 4.10).
`Command Line Interface` erkennt bei der Initialisierung, welche Pakete innerhalb der Anwendung verwendet werden und weiß dadurch welche `starter Dependencies` verwendet werden müssen, wodurch man sich diverse Importbefehle ersparen kann. Mit dem `Acuator` ermöglicht Boot der/dem EntwicklerIn die Applikation während der Laufzeit zu untersuchen. (Walls, 2012)

```
1 <dependency>
2  <groupID>org.springframework.boot</groudID>
3  <artifactID>spring-boot-starter-web</artifactID>
4 </dependency>
```

Listing 4.10: Spring Boot Example

4.4.2.2 Spring MVC

Spring MVC, Model View Controller (MVC), ist ein Spring Modul, welches verwendet wird, um Web-Applikationen zu entwickeln. MVC ist ein Design Pattern, das gerade in der Verbindung mit GUI Entwicklung weit verbreitet ist und die einzelnen Anwendungsschichten gut voneinander trennt. (Deck, 2016)

Kurz zusammengefasst, versteht man unter dem Model den Part der Anwendung, der die Daten verwaltet. Innerhalb der View befindet sich die Logik für die GUI des Programmes und der Controller ist für die Kommunikation/Interaktion zwischen

Usereingaben und dem Datenhandling verantwortlich. Innerhalb des MVC Patterns läuft jede Interaktion zwischen View und Model rein über den Controller; die Interaktion findet nie direkt zwischen View und Model statt.

Einstiegspunkt bei einer Spring MVC Anwendung ist immer das Dispatcher Servlet, welcher quasi den Hauptcontroller darstellt und die Anfragen an den zuständigen Controller weiterleitet. (Ganeshan, 2016)

Sobald Spring MVC als Ansatz für die Website Entwicklung genutzt wird, werden HTTP Anfragen über den Controller bearbeitet. Diese benötigen, wie im Codebeispiel 4.11 zu sehen ist, die `@Controller` Annotation, um als solcher definiert zu werden. (Santana, 2019)

```
1 @Controller
2  public class HelloWorldController {
3   @RequestMapping(value="/helloWorld", method=RequestMethod↩
       .POST)
4    public String helloWorld(Model model) {
5     model.addAttribute("message", "Hello World");
6    return "helloWorld";
7  }
```

Listing 4.11: Spring MVC Example

4.4.2.3 Spring Security

Bei Spring Security handelt es sich um ein Framework, das verwendet wird, um Sicherheitsbestimmungen mittels Java Platform, Enterprise Edition (JEE) in Spring Applikationen einzubinden. Spring Security lässt sich gut mit dem Spring MVC (Kapitel 4.4.2.2) verknüpfen.

Hauptsächlich genutzt wird es, um die Authentifikation sowie Autorisierung von Anfragen innerhalb der Anwendung zu behandeln. Bei einer Authentifizierung wird kontrolliert, ob die/der UserIn die nötigen Berechtigungen hat, um die gewünschte Ressource zu verwenden. Autorisierung wiederum bezeichnet die Erweiterung der

Rechte für UserInnen für bestimmte Teile eines Bereiches, welche eigentlich gesperrte Ressourcen darstellen. Eine große Stärke von Spring Security ist, dass das Framework weitestgehend den Bedürfnissen des jeweiligen angepasst werden kann. (Debnath, 2018)

Neben Features wie OpenID und O-Auth 2.0, welche durch Verwendung von Spring Security im Projekt verfügbar sind, können auch Usergruppen definiert werden, welche unterschiedliche Zugriffsrechte auf die verschiedenen Ressourcen innerhalb der Anwendung haben.

Die wichtigsten Objekte innerhalb von Spring Security sind `Security-Context-Holder`. Dort sind die Sicherheitseinstellungen für den Bereich der Anwendung gespeichert, in dem sich die/der UserIn befindet. Dieser `SecurityContextHolder` verwendet `the-Thread-Local` um diese Informationen zu speichern. Das `User-Details-Service` wird verwendet, um benutzerspezifische Informationen bezüglich Authentifikation sowie Autorisation zu laden. (Wodehouse, 2017)

4.4.3 Node.js

Node.js wurde mit dem Hintergrundgedanken entwickelt, JavaScript, welches rein im Browser lauffähig ist, mittels Stand-Alone Technologie auch außerhalb von Browsern verwendbar zu machen. Node bietet ein breit gefächertes Featureset, bestehend aus der Möglichkeit die typische JavaScript Syntax zu verwenden, dem Erstellen von direkten Datenbankabfragen und der Möglichkeit Webservers zu erstellen. (Mead, 2018)

Die Node.js Architektur geht von den typischen Entscheidungsgründen, ob es für Anwendungen geeignet ist, weg und bringt neue Aspekte mit, welche die Entscheidungsfindung beeinflussen. Werden Threads großteils genutzt, um die Central Processing Unit (CPU) zu verwenden, damit eine Applikation skalierbar bleibt, meidet Node.js die Verwendung von Threads gänzlich aufgrund deren Komplexität. Stattdessen findet innerhalb von Node die Single-Thread Event-Driven Architektur Anwendung; dadurch wird der Speicher weniger belastet, die Datenrate ist höher und die Latenz unter Verwendung ist eine bessere.

Während die meisten Applikationsservermodelle die I/O nutzen, um Daten zu empfangen, besitzt Node.js nur einen einzigen Thread und blockiert die I/O nicht, was den Vorteil hat, dass im Gegensatz zu der herkömmlichen Technik der Speicher nicht mit wartenden I/O gefüllt wird.

```
1 var http = require('http');
2 http.createServer(function (req, res) {
3 Res.writeHead(200, {'Content-Type': 'text/plain'});
4 res.end('Hello World\n');}).listen(1414, "127.0.0.1");
```

Listing 4.12: Node.js Example

Dieser kleine Codeblock, welcher in Listing 4.12 zu sehen ist, reicht aus, um einen simplen Node Web-Server zu erstellen. Durch das `http` Objekt wird das HTTP Protokoll gekapselt und mittels `http.createServer` wird ein vollwertiger Web-Server initialisiert, welcher auf den in der `listen` Methode definierten Port hört. Jede Anfrage via HTTP seitens des Clients ruft diese Funktion auf; in diesem vereinfachten Beispiel würde nur ein simples "Hello World" als Antwort zurückkommen. (Herron, 2018)

Aufgebaut ist Node.js modular, was bedeutet, dass das Grundgerüst sehr leichtgewichtig ist, kann aber durch externe Module problemlos erweitert werden und dank der sehr aktiven Community werden auch regelmäßig neue Module bereitgestellt, sodass beinahe jeder Anwendungsfall gelöst werden kann. (Dayley, 2014)

Node verwendet die High-Performance V8 JavaScript Engine; V8 verwendet die neuesten Compiler Technologien um die maximale Geschwindigkeit zu gewährleisten. (Hughes-Croucher & Wilson, 2012)
Unter einer JavaScript Engine versteht man ein Programm, welches JavaScript in sogenannten Low-Level bzw. Maschinen Code umwandelt, damit Prozessoren damit arbeiten können. (Tripathi, 2017)

4.5 JHipster

JHipster ist ein Open-Source Programm zur Generierung von modernen Web-Anwendungen auf Basis von Spring (Kapitel 4.4.2) für das Back-End und Angular (Kap-

tiel 4.2.3) oder React (Kapitel 4.2.2) für das Front-End. JHipster verwendet für die Generierung im Hintergrund Yeoman. Dieser ist ein Codegenerator, der mit dem Command-Line Befehl `yo` aufgerufen wird und ein Grundgerüst für eine komplette Anwendung oder einzelne Teile einer Anwendung generiert. Der Codegenerator besteht aus drei zentralen Punkten, um eine Web-Anwendung so optimal wie möglich zu generieren:

- `yo`, das Tool für die Generierung eines Grundgerüstes
- einem Build-Tool (z.B. npm)
- einem Packagemanager (z.B. npm)

(Raible, 2016)

Das Setup von JHipster für die Erstellung einer Web-Anwendung ist in zwei Phasen aufgeteilt: Konfiguration und Generierung. Die erste Phase ist die Konfiguration, dort wird mittels Kommandozeilenbefehle entschieden, welche Technologien eingesetzt werden. Das Ergebnis der Konfiguration ist ein `.json` File, welches im nächsten Punkt, der Generierung, verwendet wird. In der Projektgenerierung erstellt JHipster das Grundgerüst für die Anwendung, wie beispielsweise Java Klassen. Für den Kompilier- und Buildvorgang bietet JHipster ebenfalls bekannte Build-Management-Tools in Form von Maven oder Gradle an. (Halin, 2017)

Wird ein Projekt mittels JHipster initialisiert, werden zu Beginn der/dem UserIn einige Fragen zu dem Projekt gestellt. Darunter befinden sich Fragen zu verschiedenen Aspekten. Beispielsweise wird gefragt, ob es sich um eine um eine Microservice oder Monolith-Architektur handelt, welchen Authentifizierungstypen man verwenden will, welcher Datenbanktyp (relationale Datenbank oder NoSQL) im Hintergrund laufen wird und eben auch welche Frameworks verwendet werden sollen. (Dayley, 2014)

Um zu entscheiden, welcher Architekturtyp für die geplante Anwendung passend ist, gibt es einige Anhaltspunkte, welche zur Entscheidungsfindung beitragen. Monolith-Architektur bietet sich an, wenn die Applikation in Zukunft keinen großen Featurezuwachs erwartet, ein kleines eher unerfahrenes Team an der Anwendung arbeitet und Time-To-Market eine hohe Priorität hat. Wenn die NutzerInnenanzahl

gering ist und diese auch in Zukunft nicht wachsen wird, trifft das Gegenteil der oben genannten Punkte bei der Applikation zu und es empfiehlt sich, auf die Microservice-Architektur zurückzugreifen.

JHipster macht Aktualisierungen der Anwendung verhältnismäßig einfach. Während ein Update traditionell viel Zeit in Anspruch nimmt, weil der Java-Code kompiliert werden muss, Datenbankänderungen ausgeführt werden, clientseitiger Code neu kompiliert werden muss und der Applikationsserver neu gestartet werden muss, ist dieser Vorgang in JHipster weitestgehend automatisiert. Unter der Verwendung von Spring Boot, Webpack und BrowserSync ermöglicht JHipster eine live Aktualisierung des Codes. (Sasidharan, 2018)

JHipster bringt eine Reihe von Vorteilen bei der Erstellung einer Web-Applikation mit, so ist es beispielsweise aktuell eine der schnellsten Möglichkeiten, eine komplette Web-Applikation zu erstellen. Da JHipster bereits einen Großteil von nichtfunktionalen Anforderungen abdeckt, können sich EntwicklerInnen vermehrt auf die Entwicklung von funktionalen Anforderungen konzentrieren. Das Testing innerhalb von JHipster ist relativ einfach, da eine neu generierte Anwendung bereits eine weiträumige Testabdeckung bietet. Die Community hinter JHipster ist eine sehr aktive und entwickelt das Projekt ständig weiter. Dank des Open-Source Ansatzes können auch eigene Änderung relativ leicht realisiert werden. (Altmann, 2018)

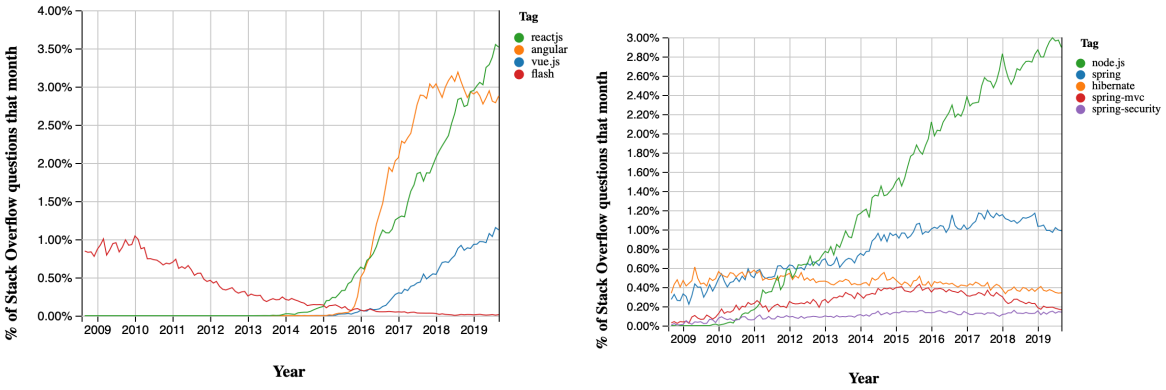
5 Technologie-Stacks

Wie in der Einleitung in Kapitel „Allgemeines zu Web-Applikationen“ ausführlich beschrieben, reicht es heutzutage nicht mehr aus, die Applikation für eine Art von Endgerät, beispielsweise Laptop, nutzbar zu machen. Diverse immer populärer werdende Geräte werden von UserInnen verwendet, um Anwendungen im Web zu bedienen. Durch diese Dezentralisierung der Verwendung ist eine serverseitige, einheitliche Bearbeitung der Daten unausweichlich. Kann das Front-End mithilfe von responsive Design von Endgerät zu Endgerät optimiert werden, muss die Serverlogik, genauer die Datenbank, unabhängig von dem gerade verwendeten Gerät die Daten bzw. die Anfragen des Clients verarbeiten. Durch diese Anforderung ist es unausweichlich, innerhalb der Applikations-Architektur die einzelnen Technologien voneinander zu trennen, das heißt einen Technologie-Stack aufzubauen, welcher aus dem Front-End (Kapitel 2.2), dem Back-End (Kapitel 2.3) und der Datenbank (Kapitel 2.4) besteht.

Bereits in Kapitel “Anforderungen an Web-Applikationen“ und den dort enthaltenen Unterkapiteln wird beschrieben, dass eine aktive bzw. große Community ein Richtungsweiser dahingehend ist, ob eine Technologie innerhalb des Technologie-Stacks verwendet werden soll, daher bietet sich die Trendanalyse von Stack Overflow[1] an, um die Lebensdauer von Frameworks innerhalb der einzelnen Schichten zu betrachten. Wie anhand der Grafik 5.1 zu sehen ist, weisen speziell clientseitige Frameworks große Sprünge bezüglich deren Nachfrage auf, während die serverseitigen Technologien, abgesehen von Node.js, relativ konstant in Verwendung sind.

Aufgrund dieses ständigen Wandels und der wechselnden Popularität von Front-End Technologien werden in den nachfolgenden Kapiteln die einzelnen Technologie-Stacks auch immer auf die Möglichkeit des Austausches des Client-Frameworks beleuchtet.

[1] https://insights.stackoverflow.com/trends



(a) Front-End Frameworks

(b) Back-End Frameworks

Abbildung 5.1: Trendanalyse
Datenquelle: Stack Overflow Trends

5.1 Vue.js / Spring inkl. Hibernate / PostgreSQL

Dieses Kapitel befasst sich mit dem Technologie-Stack, welcher auf Vue.js als Front-End Framework, Spring in Kombination mit Hibernate als Back-End Technologie sowie PostgreSQL als Datenbanksystem besteht. Wie in Kapitel „Spring“ erläutert, werden innerhalb von Spring für die Entwicklung einer Web-Anwendung weiter Module wie Spring MVC und Spring Security Anwendung finden.

Als Schnittstelle zwischen Server und Datenbank wird Hibernate innerhalb dieses Technologie-Stacks verwendet, Spring beinhaltet sogenannte HibernateTemplates, mithilfe dieser ist es ein Einfaches, Hibernate innerhalb des Spring Frameworks zu implementieren und damit einen Datenzugriff zu ermöglichen. (Mumar, 2013)

Durch die Funktionalität von Hibernate, relationale Datenbanken bzw. deren Inhalte als einfache Java-Objekte abzubilden, wird es für EntwicklerInnen einfacher, innerhalb von Spring Daten zu bearbeiten, also die Clientanfragen abzuwickeln.
Ausgewählt wurden die Frameworks aufgrund der Tatsache, dass, wie in Grafik 5.1 abgebildet, Vue.js eines der am stärksten wachsenden Frameworks im Bereich des Front-End ist und Spring in den letzten zehn Jahren konstant im Back-End genutzt wurde. Ausgehend davon, dass das Vue.js-Grundgerüst bereits mittels `npm` initialisiert wurde und das Spring-Framework ebenfalls generiert wurde, wird nachfolgend auf das Zusammenspiel bzw. die Schnittstellen eingegangen.

5.1.1 Technologiefaktoren

Wie im Theorieteil bei „Anforderungen an Web-Applikationen“ beschrieben, sollten Frameworks, welche innerhalb der Softwarearchitektur verwendet werden, bestimmte Kriterien erfüllen, um überhaupt in Betracht gezogen zu werden. Die nachfolgenden Indikatoren nehmen jeweils Bezug auf die zuvor im Theorieteil ausgearbeiteten Faktoren.

Einzig der Faktor Zukunftssicherheit ist für jede Technologie per se zu betrachten; die weiteren Kriterien sind grundsätzlich im Kontext des kompletten Technologie-Stacks zu bewerten. Wird beispielsweise Vue.js durch ein anderes Front-End Framework getauscht, kann die Ersetzbarkeit erst dann bewertet werden, wenn das komplette System wieder läuft, was bedeutet, dass auch darauf zu achten ist, wie aufwändig es ist, jegliche Schnittstellen am Server umzuschreiben.

Gerade was den Time-To-Market Aspekt betrifft, eignet sich die Verwendung von Vue und Spring relativ gut. Ein Grundgerüst, mit welchem EntwicklerInnen beginnen können zu programmieren, lässt sich bei Vue.js schnell und unkompliziert mithilfe von `npm` initialisieren. Seitens des Frameworks am Server, in diesem konkreten Fall Spring, sind die Zusatzfeatures eine große Hilfe um einen lauffähigen Back-End Code zu erstellen. Umgeht man mit Spring Boot die Konfigurationseinstellung für jegliche, innerhalb von Spring verwendete Module, werden Schnittstellenfeatures, welche für die Kommunikation mit dem Client nötig sind mit Spring MVC mitgeliefert.

Bezugnehmend auf das Kriterium Performance hat Vue dank des reaktiven Schreibens in den DOM und des Umgehens eines kompletten Neuladens der Seite, auch hier gewisse Stärken. An seine Grenzen stößt dieses Framework allerdings, wenn die Anwendung relativ komplex wird, bzw. sehr leichtgewichtiges Framework ist; eine der bekanntesten Seiten, welche Vue im Einsatz hat, ist 9GAG[2]. Anhand dieser Seite sieht man, dass dort zwar große Datenmengen existieren und verarbeitet werden müssen, aber der Aufbau der Seite relativ simpel ist.

Innerhalb dieses Technologie-Stacks wird die Sicherheit serverseitig mittels Spring Security gewährleistet, clientseitig kann mit Vue durch Eingabeüberprüfungen Angriffsversuche in erster Instanz überwacht bzw. verhindert werden; ob und in wie

[2]https://9gag.com/

weit die Frameworks die in Kapitel 4.4.2.3 beschriebenen Gefahren eindämmen können, wird in Kapitel 5.1.2 genauer beleuchtet.

5.1.2 Technologie-Stack

Die benötigte Schnittstelle beeinflusst die Austauschbarkeit des Frameworks respektive die Fähigkeit der gesamten Applikation, auf Technologieänderungen zu reagieren. Muss unter großem Aufwand eine Clientschnittstelle im Back-End programmiert werden, damit der Client überhaupt mit dem Server kommunizieren kann, wird die Wahl wohl nicht auf dieses Front-End Framework fallen.

Vue.js lässt sich simpel und ohne viel Aufwand innerhalb des Technologie-Stacks mit Spring kombinieren; die Codeteile 5.1 und 5.2 reichen gänzlich aus, um Vue.js mit Spring kommunizieren zu lassen. Dank Spring MVC ist es im Spring Framework möglich, Routen zu erstellen, auf welche der Clientanfragen senden kann.

```
1 @GetMapping("/helloWorld")
2 public String helloworld() {
3   return "helloworld!"
4 }
```

Listing 5.1: Vue Spring Example Server

Unter Routen sind bestimmte Endpunkte innerhalb der Software zu verstehen. Im Beispielcode 5.1, wird der Endpunkt `\helloWorld` erstellt; wird dieser aufgerufen, wird der Programmcode, welcher nach der `@` Annotation folgt, ausgeführt. Routing ist in der modernen Webentwicklung einer der gängigsten Wege, um Daten zwischen den unterschiedlichen Layern zu versenden. (Long, Lui, Gray & Chan, 2011)

Serverseitig reicht dieser kleine Codeblock völlig aus, um einen String an den Client zurückzusenden, wenn dieser diesen Endpunkt aufruft. Genau dies geschieht bei Listing 5.2. In Zeile 1 wird eine neue Funktion, `callHelloWorld`, definiert und diese kann dann beispielsweise mittels `onClick()` auf einen Button in der GUI gebunden werden. Wird diese Funktion nun aufgerufen, wird mit der Hilfe des Axios Vue Moduls, welches ein HTTP Client für Vue ist, eine `get` Anfrage an den Server gesendet

und explizit wieder der Endpunkt `\helloWorld` aufgerufen. Bei diesem einfachen Beispiel würde der Server nun den String `helloworld!` zurücksenden, auf welchen in `response.data` clientseitig zugegriffen werden kann.

```
callHelloWorld ();{
 axios.get('http://localhost:8080/helloWorld')
  .then(response => {
   this.response = response.data
  }).catch(e => {
    this.errors.push(e)
   })
}
```

Listing 5.2: Vue Spring Example Client

Ist eine Client-Server Kommunikation gegeben, wird es in den meisten Anwendungen unausweichlich sein, ein Datenbanksystem im Hintergrund zu initialisieren bzw. zu integrieren. Innerhalb dieses Technologie-Stacks ist die Wahl des Datenbanksystems im Hintergrund auf PostgreSQL gefallen.

Ein Kriterium, weshalb die Entscheidung auf diese Datenbank gefallen ist, ist, dass PostgreSQL mittels OLTP bei Systemen mit vielen Datenmodifizierungen (insert, update, delete), als auch bei Applikationen mit großen Datenmengen dank OLAP verwendbar ist. Zusätzlich handelt es sich hier um ein relationales Datenbanksystem, was wiederum in Kombination mit Hibernate einfach in ein Java Back-End nutzbar ist; Hibernate - oft auch als ein ORM bezeichnet - macht aus diversen Datenbankinformationen (Tabellen, Datensätze) Java-Objekte, welche wiederum von Spring gut verarbeitet werden können.

Die Einbindung von Hibernate in Spring ist mittels Spring Boot relativ unkompliziert; der erste Teil von Listing 5.3 zeigt den Ausschnitt der pom.xml der nötig ist, damit Hibernate durch Spring Boot in die Software integriert wird. Der zweite `<dependency>` Block ist nötig, um Hibernate mitzuteilen, dass als Datenbanksystem PostgreSQL läuft. Somit reichen diese zehn Konfigurationszeilen aus, um Hibernate-Funktionalitäten in Spring zu ermöglichen.

```
<dependencies>
 <dependency>
  <groupId>hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
 </dependency>
 <dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
 </dependency>
</dependencies>
```

Listing 5.3: Hibernate und PostgreSQL in pom.xml

Kommuniziert die Datenbank mit dem Server, der Server mit dem Client und vice versa, ist nun der in Abbildung 2.1 aufgezeigte Kreislauf einer Web-Anwendung gegeben.

Um bei Markteintritt eine voll funktionsfähige Applikation auszuliefern, muss verständlicherweise jegliches Zusammenspiel von Server, Client und Datenbank funktionieren. Neben der Realisierung der einzelnen Layer, welche, wie in Kapitel 5.1.1 erläutert, recht simpel vonstatten geht, muss auch die Zeit für die Programmierung der Schnittstellen in die Time-To-Market eingerechnet werden. Eben diese Schnittstellenentwicklung ist in einer Architektur mit diesen drei gewählten Frameworks relativ einfach und rasch durchzuführen. Aufgrund der Möglichkeit von Spring Boot, Abhängigkeiten automatisch zu generieren und somit das aufwendige Konfigurieren zu umgehen und die in Spring MVC, mitgelieferten Funktionalitäten des Routings, vermindern die Time-To-Market um ein Vielfaches.

Wie innerhalb dieser Arbeit schon des Öfteren thematisiert, ist eine dynamische GUI nahezu eine Grundvoraussetzung von modernen Web-Anwendungen. Inhalte dynamisch, direkt und live auf der Benutzeroberfläche zu ändern, kann mit Vue.js problemlos realisiert werden. Vue greift, wie in Kapitel 4.2.1 erläutert, reaktiv auf den DOM zu, um dort, ohne ein gänzliches Neuladen der Seite, von der Nutzerin/vom Nutzer geforderte Änderung durchzuführen. Aufgrund der reibungslosen Kommunikationsmöglichkeit zwischen Vue und Spring ist es für EntwicklerInnen ein Leichtes, Daten aus der Datenbank auszulesen oder zu bearbeiten und gegebenenfalls der/dem UserIn zur Verfügung zu stellen.

Bezugnehmend auf Stabilität und Verfügbarkeit der Anwendung bringen die gewählten Frameworks viele Features mit, um dies zu gewährleisten. Dass aufwändige Datenbankabfragen oder große Datenmengen das System zum Absturz bringen, ist dank OLAP und OLTP bei PostgreSQL in Kombination mit einer effektiv programmierten JDBC auf Softwareebene eher unwahrscheinlich. Code, welcher eine hohe Rechenleistung benötigt, weil dieser nicht optimal entwickelt wurde, kann innerhalb dieses Technologie-Stacks großteils vermieden werden, da sowohl Vue, als auch Spring ein Grundgerüst für die EntwicklerInnen anbieten.

Sowohl in Abbildung 5.1 zu sehen und in Kapitel 4.2.1 zu lesen, ist Vue.js ein noch relativ neues, aber sehr beliebtes Framework, was eben dazu führt, dass die Zukunftssicherheit dieser Technologie nicht vollends gegeben ist. Grundsätzlich kombiniert Vue einige Features aus React und Angular, was dieses Framework sehr attraktiv für zukünftige Applikationen macht und dank des großen Zuspruches der Open-Source Community hat Vue großes Potential eine lange Lebensdauer zu haben.(Martin, 2019)

Trotz dieser vorwiegend positiven Eigenschaft von Vue ist die Zukunftssicherheit, gerade was Front-End Technologien betrifft, nie vollends gegeben.
Im Kontrast zu den neuen, modernen Front-End Frameworks, läuft mit Spring im Back-End ein etabliertes und auf Jahre in Verwendung befindliches Framework. Abbildung 5.1 zeigt auch die Beständigkeit, mit welcher Spring seit 2009 bei Stack Overflow[3] thematisiert wird, was unter anderem auf eine aktive Community sowie eine ständige Weiterentwicklung der Technologie schließen lässt.

Zusätzlich laufen sowohl Spring, als auch Vue.js auf sehr populären Programmiersprachen, nämlich Java sowie Javascript. Beide Sprachen befinden sich seit geraumer Zeit unter den Favoriten, wenn es um die meist verwendeten Programmiersprachen geht. Diese haben sich durch deren großen Umfang an Features, welche in Kapitel 4.1.2 für Java und in Kapitel 4.1.1 genauer beschrieben werden, an der Marktspitze etablieren können. (Okoi, 2019)

Im Kontext von Web-Anwendungen ist der Sicherheitsaspekt einer der wichtigsten und darf bei der Entwicklung nicht außer Acht gelassen werden. Bietet Spring mit Spring Security schon ein eigenes Modul für gewisse Absicherungen, bieten Frameworks wie Hibernate ebenfalls einen gewissen Schutz für die Applikation.

[3] https://stackoverflow.com/

Die in Java verfügbaren Methoden zur Ausführung von SQL Code `execute()`, `execute-Query()` und `execute-Update()`, sind gleichermaßen anfällig für Injections, da diese Methoden jegliche, mit Semikolon getrennte, Befehle ausführen, ohne diese zu überprüfen. Der bekannteste und effektivste Weg, eine SQL Injection zu verhindern, sind Prepared Statements. Hierbei wird der Query- und Datenkontext gänzlich voneinander getrennt; dies bedeutet, dass in die gewünschte Query nur noch die entsprechenden Parameter, wie in Listing 5.4 mithilfe von `?` als Platzhalter eingefügt. (Schadow, 2014)

```
1  SELECT * FROM products WHERE id = ?;
```

Listing 5.4: Prepared Statement

Um sich gegen Broken Authentication zu schützen, ist ein optimiertes Session Management innerhalb der Software nötig, welches mit Spring Security gewährleistet werden kann. Um zu verhindern, dass UserInnen unzählig viele Sessions aufrecht haben und somit im System angemeldet sind, kann in Spring Security die Anzahl der maximal bestehenden Sessions per User konfiguriert werden. (Knutson, Winch & Mularien, 2017)

Zusätzlich zu der Begrenzung von bestehenden Sessions, kann Spring Sessions eine gewisse Lebensdauer zuweisen. Läuft diese ab, ist die Session und die damit einhergehende Session ID ungültig und die/der UserIn muss sich erneut bei der Applikation anmelden. Das Mitsenden der Session ID direkt in der Uniform Resource Locator (URL), kann mit Spring Security deaktiviert werden und somit ist die ID nicht mehr in Plain Text für jeden sichtbar. Weiters können auch Cookies mit einem einfachen Flag so konfiguriert werden, dass der Browser diese nicht öffnen kann. (Paraschiv, 2019)

Die Kommunikation zwischen Server und Client sollte heutzutage standardmäßig über TLS erfolgen und kann innerhalb des Spring Frameworks aktiviert werden.
Um Cross-Site-Scripting, also die Möglichkeit Fremdcode im System auszuführen bzw. zu verhindern, müssen Zeichen escaped werden, damit aus einem ausführbaren Code, ein nicht lauffähiger String an den Browser gesendet wird. Hierzu kann Spring MVC eingesetzt werden, da das Framework gewisse Konfigurationen bietet, um genau dies zu gewährleisten. Spring MVC bietet diese Konfiguration auf den folgenden Ebenen an: global, innerhalb einer Seite oder gebunden auf bestimmte Ele-

mente. (Patil, 2016)

Abschließend lässt sich über den Sicherheitsaspekt sagen, dass dieser Technologie-Stack, gerade serverseitig, sehr gut aufgestellt ist. Somit lohnt sich die Einbindung der Spring Frameworks Spring MVC für die Verhinderung von Cross-Site Scripting sowie die Implementierung von Spring Security um jegliche mögliche Sicherheitslücken innerhalb einer Web-Anwendung zu schließen.

Zusammenfassend zu dieser Architektur und den darin enthaltenen Frameworks lässt sich sagen, dass der Großteil der zuvor definierten Auswahlkriterien zumindest erfüllt werden. Ausgehend von der Annahme, dass die Applikation eine, auch in Zukunft, leichtgewichtige bleibt, spricht als Front-End Framework relativ wenig gegen Vue. Hauptkritikpunkt ist - neben der fehlenden Komplexität - die noch kurze Zeit, in der Vue.js am Markt ist. Bleibt die Popularität auf jenem Level, auf dem sie zurzeit ist, hat Vue großes Potential, zukünftig - mit Hilfe von Zusatzmodulen, welche von der Community entwickelt werden - auch für komplexe Web-Anwendung in Betracht gezogen zu werden.

Spring auf dem Back-End zu nutzen scheint die nahezu perfekte Lösung zu sein, da gerade die Zusatzmodule Spring MVC, Spring Boot und Spring Security aus dem so schon mächtigen Framework einen Allrounder machen, der jegliche Kriterien, die man an ein Serverframework stellt, erfüllt. Wird, wie in diesem Fall, nicht das springeigene Spring Data für die Datenverwaltung verwendet, lassen sich ohne großen Konfigurationsaufwand diverse Hibernate Funktionalitäten für den Datenverarbeitungsprozess integrieren. Hibernate kann - wenn nötig - durch Spring Boot ohne Zeitverlust in Spring integriert und im vollen Umfang genutzt werden.

Zwar sollte zu Beginn Klarheit darüber bestehen, ob die Anwendung eher transaktionslastig sein wird oder eher große Datenmengen verarbeiten muss, jedoch können mit PostgreSQL beide Anwendungsfälle behandelt werden. Unter Berücksichtigung des Umstandes, dass Hibernate als ORM in die Architektur integriert wird, hat PostgreSQL als relationale Datenbank zusätzliche Vorteile, was die Kommunikation mit dem Serverframework betrifft.

Wurden die alleinstehenden Komponenten der jeweiligen Schichten, mit relativ wenig Programmieraufwand, mittels Schnittstellen zusammengefügt, bietet dieser Technologi-Stack eine nahezu optimale Basis um eine moderne, dynamische und zeitge-

mäße Web-Anwendung zu entwickeln. Jegliche mögliche Sicherheitslücken, welche von OWASP herausgefunden wurden, können durch das Featureset der Frameworks und deren Zusatzmodulen geschlossen werden. Wurde innerhalb dieses Abschnittes nur der Grundstock der einzelnen Technologien beleuchtet, so können sowohl Vue.js, als auch Spring durch eine Vielzahl an Modulen und Bibliotheken für nahezu jeden Anwendungsfall erweitert werden.

5.1.3 Austausch Vue.js gegen Angular

Ausgehend davon, dass nicht alle Web-Applikationen bezüglich der Komplexität mit dem Vue-Framework ausreichend Funktionalitäten zur Verfügung gestellt bekommen, wird innerhalb des Technologie-Stacks in diesem Kapitel Vue gegen Angular ausgewechselt.

Können sowohl Spring als auch PostgreSQL trotz geänderten Anforderungen an die Anwendung bestehen bleiben, da beide mit unterschiedlichen Komplexitätsleveln keine Probleme haben, wird nur analysiert, wie hoch der Aufwand ist, wenn das Front-End verändert wird. Wie in Kapitel 4.2.3 beschrieben, eignet sich Angular, im Gegensatz zu Vue, für die Realisierung von großen und komplexen Systemen, da es hauptsächlich für solche Anwendungsfälle konzipiert und entwickelt wurde.

Einer der größten Unterschiede, welcher auch zu einer besseren Skalierung von Angular führt ist die Verwendung von TypeScript anstelle von JavaScript. TypeScript ist eine Erweiterung zu JavaScript und liefert standardmäßig Werkzeuge wie Autocompletion, Typsicherheit und Refactoring. Aufgrund dieser Unterstützungen seitens TypeScript bietet Angular eine strukturiertere Programmierumgebung an und liefert damit auch bei komplexen Systemen skalierbaren und besser lesbaren Code. (Halliday, 2018)

Anforderungen seitens der BenutzerInnen an eine moderne GUI deckt Angular gänzlich ab, da es sowohl das Laden von speziellen Teilen einer Seite unterstützt, als auch das dynamische Laden von ganzen Inhalten. Gerade mit der Möglichkeit des Neuladens von speziellen Komponenten anstelle der gesamten Seite, ähnlich wie es Vue anbietet, wird die Performance von der GUI verbessert.

Durch die Berücksichtigung der Testbarkeit des Systems von Beginn an, macht es Angular den EntwicklerInnen auch bei großen Systemen relativ einfach einzelne Komponenten zu testen und somit die Stabilität und Verfügbarkeit gravierend zu erhöhen. Bezugnehmend auf Kapitel 4.2.3, bietet Angular standardmäßig Routing-Funktionalitäten um den Client mit dem Server zu verbinden; wie eine solche Schnittstelle programmiertechnisch aussehen kann ist bei 5.5 zu sehen. Der Code bzw. die Schnittstelle am Server bleibt die gleiche, das heißt, es wird nach wie vor der Endpunkt `/HelloWorld` verwendet, da sowohl Angular, als auch Vue dieselbe Schnittstelle ansprechen können. Dies bringt den Vorteil mit sich, dass bei dem Front-End Tausch der Servercode weitestgehend unberührt bleiben kann und damit der Programmieraufwand - abgesehen vom Front-End - ein minimaler ist.

```
1  @Injectable()
2  export class HelloWorldService {
3   private myUrl: string;
4   constructor(private http: HttpClient) {
5    this.myUrl = 'http://localhost:8080/helloWorld';
6   }
7   public getHelloWorld(): Observable<> {
8    return this.http.get<>(this.myUrl);
9   }
10 }
```

Listing 5.5: Angular Spring Example Client

Um sich vom Angular-Framework zum Server verbinden zu können, kann ein Service geschrieben werden, welches sich stets auf einen Endpunkt verbindet.
Die `@Injectable` Annotation von Angular wird dazu benötigt, um aus einem einfachen Service, wie in Listing 5.5 `HelloWorldService`, eine Komponente zu machen, welche in anderen Codeteilen mittels Dependency Injection geladen und genutzt werden kann. (Anil, 2018)

Wird die `getHelloWorld()` Funktion aufgerufen, bekommt der Client als Antwort vom Server eine Observable Objekt zurück. Diese sind Teile der Reactive Bibliothek und werden verwendet, um innerhalb einer Applikation einzelne Teile miteinander zu verknüpfen. Grundsätzlich arbeitet man in Angular nicht direkt mit den Observable Objekten, da diese weitestgehend im Hintergrund zum Einsatz kommen. Eine

Ausnahme, wenn direkt mit Observable Objekten gearbeitet wird, ist, wenn der Client Daten vom Server bzw. der Datenbank benötigt. (Freeman, 2019)

Größer wird der Programmieraufwand nicht nur bei dem Austausch des Codes, weg von Vue.js und hin zu Angular, sondern auch die Templates müssen umgeschrieben werden, damit Angular damit arbeiten kann. Codebeispiel 5.6 zeigt einerseits eine `foreach` Schleife in Vue und andererseits in Angular, daran ist schon erkennbar, dass sich die Syntax nicht gravierend unterscheidet, aber doch eine andere ist, während statischer Output, also beispielsweise reiner Text, in beiden Technologien innerhalb von zwei geschwungenen Klammern übergeben werden kann. Das Einlesen in eine neue Programmiersprache umgehen EntwicklerInnen aufgrund der Tatsache, dass sowohl Vue, als auch Angular auf JavaScript basieren. Einzig die Syntax der Frameworks müssen ProgrammiererInnen erlernen.

```
1 <!-- Foreach Loop in Vue.js. -->
2 <ul><li v-for="item in list" :key="item"></li></ul>
3 <!-- Foreach Loop in React. -->
4 <ul><*ngFor="let item of list"></li></ul>
```

Listing 5.6: Vue.js vs Angular

EntwicklerInnen haben bezüglich Technologietausch auf Front-End Seite und den damit einhergehenden Time-To-Market Indikator, kaum Probleme. In Bezug auf die anderen Schichten, bleibt die Datenbank gänzlich unberührt von dem Wechsel und auch die Schnittstellen am Server bleiben ident.

Ein anderer Vorteil von Angular zu Vue.js ist die längere Marktpräsenz. Hier bietet Angular schon eine längere Lebensdauer, mit ziemlich großer Resonanz seitens der Community, wie man anhand von Abbildung 5.1 sehen kann. Zusätzlich zu der Open-Source Community kann Angular auf Google[4] im Hintergrund als Erfinder bauen; ausgehend von der Website madewithangular[5], ist Angular bei einigen großen Firmen, wie beispielsweise Forbes, Samsung und diversen Google Diensten, als Front-End Framework im Einsatz.

[4]https://www.google.at
[5]https://www.madewithangular.com/

5.1.4 Conclusio

Abschließend lässt sich für diesen Technologie-Stacks sagen, dass, wie in diesem Kapitel festgestellt wurde, diese Architektur viele Vorteile und Stärken mit sich bringt, um eine moderne, sichere und zukunftsorientierte Web-Anwendung zu entwickeln. Spring aufseiten des Servers liefert durch diverse zusätzliche Frameworks und in Kombination mit Hibernate, verschiedenste Features, welche EntwicklerInnen im Back-End benötigen. Relationale Datenbanken, in diesem Stack PostgreSQL, arbeiten mit Hibernate performant zusammen und sind leicht mit Spring zu kombinieren.

Vue.js bietet auf der Clientseite ein breites Featureset. Um jegliche Anforderungen an die GUI zu realisieren, wurde die Applikation im Laufe der Zeit zu komplex für die leichtgewichtige Technologie. Aus diesem Grund lässt sich Vue, abgesehen von dem Front-End, ohne großen Aufwand durch Angular ersetzen; der Großteil des Back-End Codes bleibt bei einem solchen Wechsel gänzlich unberührt.

Vorteile	Nachteile
dynamische GUI	fehlende Zusatzpakete für Vue.js
großes Toolset im Back-End	Angular schwergewichtig, speziell bei Kleinprojekten
Schutz vor Angriffen	
einfaches Arbeiten mit der Datenbank	
einfaches Routing mit Spring MVC	

Tabelle 5.1: Gegenüberstellung Technologie-Stack 1

5.2 React / Node.js / MongoDB

Diese Programmarchitektur konzentriert sich sowohl clientseitig, als auch serverseitig auf JavaScript als Entwicklungssprache und ist daher für EntwicklerInnen mit reiner JavaScript Erfahrung besser geeignet als beispielsweise das auf Java basierende Serverframework Spring. Ebenfalls wird innerhalb dieses Stacks keine relationale Datenbank als Informationsspeicher herangezogen, sondern mit MongoDB eine NoSQL

Datenbank. Wie bereits in Kapitel 4.3.3 beschrieben, sollte MongoDB eher bei analytischen und nicht bei transaktionsorientierten Systemen zum Einsatz kommen.

Neben der reinen Verwendung von JavaScript macht diesen Technologie-Stack die Tatsache, dass sowohl Node.js, als auch React laut Grafik 5.1 die meistverwendeten Technologien unter den verglichenen sind, interessant. Ausgehend von der Tatsache, dass immer mehr Datenanalyse bzw. große Daten verarbeitet werden müssen, macht die Integration einer NoSQL Datenbank Sinn und bringt Aufschlüsse bei der Analyse eines Technologie-Stacks.

5.2.1 Technologie Faktoren

Bezugnehmend auf die im Theorieteil 3 ausgearbeiteten Faktoren, wird diese Architektur hinsichtlich ihrer Nutzbarkeit analysiert und bewertet.
In Anbetracht dessen, dass es sich hier um eine NoSQL und nicht um eine relationale Datenbank handelt, wird die Designentscheidung bewusst auf diese Technologie gefallen sein. Ein Wechsel von NoSQL zu einem relationalen Datenbanksystem in einem späteren Entwicklungsschritt, ist hinsichtlich des Aufwandes, den ein solcher Wechsel mit sich bringen würde, eher unwahrscheinlich.

Wie in der Einleitung schon erwähnt, bringt dieser Stack den Vorteil von einer einheitlichen Programmiersprache mit sich, was EntwicklerInnen das Erlernen neuer Sprachen erspart und somit die Zeit zur Markteinführung rapide sinken lässt. Einzig die Verwendung von MongoDB könnte den ganzen Entwicklungsprozess, aufgrund der Tatsache, dass MongoDB, wie in Grafik 5.2, bei weitem nicht so weiterverbreitet ist wie MySQL, verzögern.

Ähnlich wie Vue.js arbeitet React ebenfalls mit dem virtuellen DOM um auch bei großen Systemen die Performance hochzuhalten, auch werden bei React nur die nötigsten Teile einer Website neu geladen, was wiederum die Performance positiv beeinflusst.

Referenzierend auf Kapitel 4.2.2, wurde React dahingehend konzipiert, um auch bei großen Datenmengen performant zu arbeiten; gerade dieser Punkt lässt auf eine gute Kompatibilität mit MongoDB schließen - da, wie Kapitel 4.3.3 zu lesen ist - MongoDB

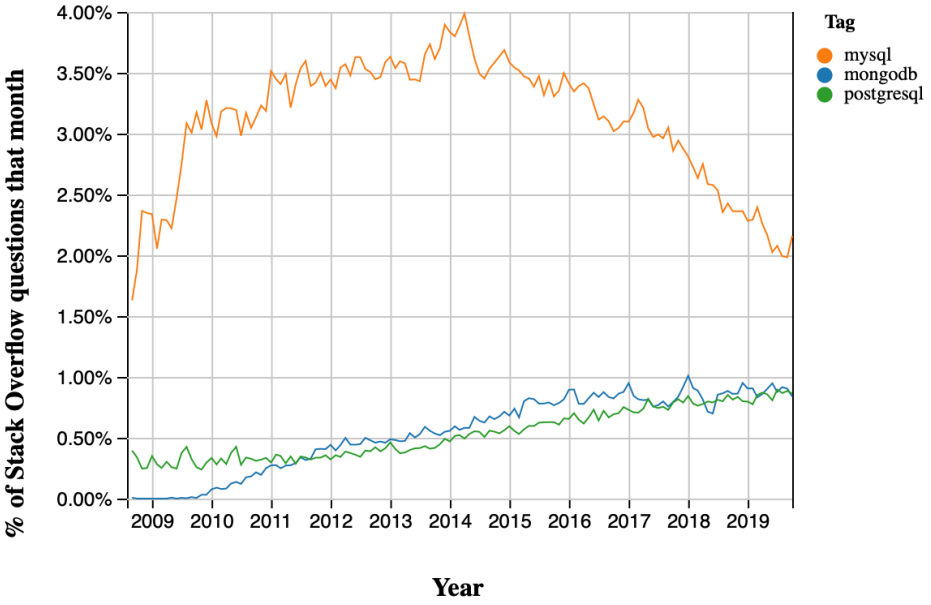


Abbildung 5.2: Trendanalyse Datebanken
Datenquelle: Stack Overflow Trends

für die Verarbeitung von große Datenmengen entwickelt wurde.

Auf der Serverseite läuft ein Node.js Webserver, welcher die Fähigkeit der Asynchronität besitzt. Das bedeutet, dass die Anwendung nicht auf Antworten des Servers wartet, sondern den Programmablauf weiter ausführt. Probleme wie fehlende, noch nicht vom Server zurückgesendete Daten, können auftreten, können aber mittels `await` Funktionen abgefangen werden. Grundsätzlich bietet diese Fähigkeit hinsichtlich auf Performance Vorteile, da die Applikation nicht zwingend auf die Verarbeitung einzelner Anfragen warten muss. (Liang, 2017)

Bezüglich der Sicherheit und den von OWASP beschriebenen Sicherheitslücken innerhalb einer Web-Anwendung, wird innerhalb dieses Technologie-Stacks, im Gegensatz zu Spring Security, kein Framework mitgeliefert, welches sich rein auf die Sicherheit fokussiert. Wie und ob auch diese Architektur jegliche Lücken schließen kann, wird im nachfolgenden Kapitel 5.2.2 untersucht.

5.2.2 Technologie-Stack

Um eine lauffähige Software bieten zu können, ist es unumgänglich, die einzelnen Schichten miteinander zu verknüpfen. Ausgehend davon wird zu Beginn der Aufwand beleuchtet, welcher für die Schnittstellenprogrammierung aufgebracht werden muss, damit React, Node.js und MongoDB miteinander kommunizieren können.

Anhand von 5.7 ist der Codeblock zu sehen, welcher nötig ist, um innerhalb des Node.js Frameworks einen Einstiegspunkt zu erzeugen. Wird eine Clientanfrage an `/api/helloWorld` gesendet, wird diese asynchron vom Server abgearbeitet. Node selbst lässt sich mittels dem Terminalbefehl `make` schnell und problemlos initialisieren und verwenden.

```
1 const Countries = mongoose.model('countries');
2 app.get('/api/helloWorld', async (req, res) => {
3  let countries = await Countries.find();
4  return res.status(200).send(countries);
5 });
```

Listing 5.7: React Node.js Example Server

Wurde MongoDB heruntergeladen und installiert, muss nur noch ein Datenordner angelegt werden, auf welchen die Mongoinstanz später zugreifen kann. Starten lässt sich MongoDB danach einfach mit dem Kommandozeilenbefehl `mongod -dbpath /datenordner`, mittels –dbpath wird angegeben, wo sich der zuvor angelegte Datenordner innerhalb des Dateisystems befindet. (Trelle, 2014)

Um Node.js Zugriff auf MongoDB zu ermöglichen, existieren Zusatzmodule, wie beispielsweise `mongoose`. Mongoose ist ein Object Document Mapping (ODM), das NoSQL Äquivalent zu einem ORM innerhalb von relationalen Datenbanken. (Mardan, 2018)

Listing 5.8 zeigt, dass der Aufwand für die Erstellung eines neuen Schemas recht unkompliziert ist. Bei diesem Beispielcode wird mittels `mongoose` ein Schema mit dem Namen `countriesSchema` mit den Attributen name und capitalCity erstellt und in die MongoDB Instanz eingefügt.

```
const mongoose = require('mongoose');
const {Schema} = mongoose;
const countriesSchema = new Schema({
 name: String,
 capitalCity: String,
})
mongoose.model('countries', countriesSchema);
```

Listing 5.8: React Node.js Example MongoDB

Aufgrund der Tatsache, wie in Kapitel 4.2.2 beschrieben, dass React kein großes Standard-Featureset mitliefert, wird für die Kommunikation, gleich wie bei Vue, der HTTP Client `axios` verwendet. Wie Codebeispiel 5.9 zeigt, ist die Initialisierung einer Schnittstelle zum Server mit wenigen Zeilen Code realisiert. Ruft der Client die Funktion `getTheWorld` auf, wird am Server die Route `/api/helloWorld` angesteuert und die darin ausprogrammierte Logik ausgeführt.

In diesem konkreten Fall würde nun der Server alle Datensätze, welche im `countries-Schema` vorhanden sind, aus der MongoDB Instanz auslesen und dieses Array wieder an den Client zurücksenden.
Dieses Beispiel zeigt, dass diese drei Frameworks untereinander kompatibel sind und sich der Aufwand für die Schnittstellenentwicklung mehr oder weniger in Grenzen hält.

```
export default {
 getTheWorld: async () => {
  let res = await axios.get('/api/helloWorld');
  return res.data;
 }
}
```

Listing 5.9: React Node.js Example Client

Bezugnehmend auf die Anforderung einer dynamischen Benutzeroberfläche werden EntwicklerInnen mit React bei dieser Umsetzung keine Probleme haben. Durch die Fähigkeit der DOM-Manipulation, kann React jegliche Bereiche der Anwendung dynamisch abändern. Zusätzlich bietet Node.js die Möglichkeit, WebSockets zu ver-

wenden, um der/dem BenutzerIn in Echtzeit Benachrichtigungen zu senden. Web-Sockets funktionieren anders als die bekannte Pull-Technologie bei der Client-Server Kommunikation. Das heißt, der Client fragt beim Server an und dieser sendet eine Antwort zurück. Bei WebSockets ist immer eine offene Verbindung zwischen Client und Server gegeben. Sollte nun am Server etwas geschehen, das für den Client relevant ist, bekommt dieser ohne ein Zutun eine Mitteilung vom Server. (Chopra, 2015)

Obwohl die Zukunftssicherheit bei Front-End Technologien schwer vorauszusagen ist, bietet React eine stabile Basis, was in Zukunft wiederum eine Weiterentwicklung gewährleistet. Steht mit Facebook ein riesen Unternehmen als ursprünglicher Entwickler hinter der Technologie, so ist auch die Popularität des Frameworks, wie Grafik 5.1 zeigt, eine große und ein guter Indikator dafür, dass diese Technologie auch in Zukunft bestehen wird.

Node.js hat seit ungefähr zehn Jahren ein ständiges Wachstum bezüglich der Zahlen der NutzerInnen. Aufgrund dessen, dass JavaScript im Hintergrund läuft, erfreut sich diese Technologie einer großen Beliebtheit und bietet mit der Asynchronität und Web-Sockets Features an, welche auch in Zukunft gefragt sein werden. Ist es zu Beginn ein relativ leichtgewichtiges Framework, gibt es, dank der umfangreichen Community, genügend Zusatzmodule für jegliche Anwendungsfälle.

Bietet Spring mit Spring Security ein eigenes Modul für die Gewährleistung der Sicherheit für die/den UserIn innerhalb der Web-Anwendung, ist dies bei Node.js nicht gegeben. Der Vorteil, wenn eine NoSQL Datenbank verwendet wird, ist, dass damit SQL Injections nicht mehr möglich sind, da die Daten in Objekten gespeichert sind, welche nicht mit SQL Statements ausgelesen werden können.
Um potentiellen AngreiferInnen innerhalb der Node Anwendung die Session ID nicht auslesbar zu machen, empfiehlt es sich, diese ID nicht bei der URL anzufügen, sondern beispielsweise wie in Listing 5.10 zu sehen ist, in einem Cookie.

Kommunikation innerhalb des Webs sollte immer sicher vonstatten gehen, also anstelle von HTTP ist immer HTTPs zu verwenden. Mit dem Setzen des `secure` Flags aktiviert man bei Node.js, dass dieses Cookie nur über eine sichere Verbindung gesendet werden soll. Das dritte Flag, `httpOnly`, blockiert den Clientzugriff mittels JavaScript auf dieses Cookie, womit dieses Cookie ausschließlich über HTTPs an den Client gesendet werden kann. (Smith, 2015)

Um Cross-Site-Scripting zu verhindern, kann einerseits wieder das `httpOnly` Flag gesetzt werden, damit kein Clientcode Zugriff auf diverse Cookies bekommt, zusätzlich kann in der Node.js Anwendung dem Header das `Content Security Policy` Flag gesetzt werden. Damit ist es möglich anzugeben, von wo aus ausführbare Scripts stammen dürfen; man erstellt so eine Art Whitelist. (Duuna, 2015)
Zusätzlich zu dieser Vorkehrung gegen Cross-Site-Scripting am Server, bietet React auf der Front-End Seite schon einen Schutzmechanismus dahingehend an und zwar maskiert React alle Strings die in einer Eingabemaske eingegeben werden und verhindert so die Möglichkeit bösartige Scripts auszuführen. (Stefanov & Demmig, 2017)

```
app.use(session({
 secret: 'secrectString',
 cookie: {
  maxAge: 3600000,
  secure: true,
  httpOnly: true
 }
}));
```

Listing 5.10: Node.js Security Konfiguration

5.2.3 Austausch Node.js gegen Spring

Zwar bringt Node.js - wie im vorangegangenen Kapitel erläutert - einige Features und Vorteile als Back-End Framework mit, jedoch fehlen innerhalb dieser Technologie einige Hilfsmodule, welche die Konfiguration, Sicherheit und Performance betreffen. Eben solche Frameworks für die Automatisierung von Abläufen bietet Spring, weshalb im folgenden Kapitel analysiert wird, ob ein Technologiewechsel auf dem Server ohne weiteres möglich ist und welcher Aufwand damit einhergehen würde.

Die größte Änderung, welche auf die EntwicklerInnen bei diesem Tausch zukommen würde, ist die Programmiersprache, denn hier würde ein Wechsel von TypeScript auf Java vollzogen werden müssen. Ist seitens der ProgrammiererInnen keine Erfahrung mit dem Umgang von Java vorhanden, würde das die Time-To-Market signifikant erhöhen, da sich die Programmiersprachen, wie in den Kapiteln 4.1.2 für Java und 4.1.1

für JavaScript beschrieben wird, stark unterscheiden.

Wurde die Entwicklung der Applikation, wie in Kapitel 3.6 zu lesen ist, modular aufgebaut, sollte der Front-End Code, trotz eines Technologiewechsels am Server, weitestgehend unberührt bleiben. Wie das Routing innerhalb von Spring aussehen kann ist im Codebeispiel 5.1 zu sehen; dieser Endpunkt sollte auch nach dem Tausch derselbe sein, was zur Folge hat, dass der Zugriffspunkt am Client unverändert bleiben kann.

Wie die einzelnen Sicherheitsrisiken innerhalb dieser Architektur vermieden werden können, wurde bereits eingängig im vorherigen Kapitel beschrieben. Zusätzlich zu den diversen Unterstützungen seitens Spring Security bietet hier das Escaping von Eingabestrings von React ein weiteres Werkzeug, um die Applikation abzusichern.

Spannend wird bei dieser Architektur das Zusammenspiel zwischen einer NoSQL Datenbank und Spring mit Hibernate, welches wieder als Datenschnittstelle verwendet wird. In den vorangegangenen Kapiteln wurde Hibernate durchgängig als ORM bezeichnet, was heißt, dass bisher der Fokus auf relationalen Datenbanksystemen lag. Um nun Hibernate und MongoDB miteinander zu verknüpfen und arbeiten zu lassen, wurde Hibernate Object Grid Mapper (OGM) entwickelt.

Hibernate OGM basiert auf der Basis der JPA Engine und wurde entwickelt, um innerhalb von Java mit NoSQL Datenbanken arbeiten zu können. Es bietet den EntwicklerInnen als Modell die bereits erwähnte JPA an und speichert im Hintergrund NoSQL Informationen anstelle von relationalen Datenbanken.

Aufgrund der Tatsache, dass es sich bei Hibernate OGM um ein relativ neues Projekt handelt, werden, im Gegensatz zu MongoDB, nicht alle NoSQL Systeme unterstützt. Aufgrund dessen, dass Hibernate OGM auf den Hibernate ORM Modulen aufbaut, wird auch innerhalb dieser Technologie JDBC Layer verwendet, um Datenbankverbindungen aufzubauen. Der sogenannte Provider ist zuständig für die spezifische Verbindung zur Datenbank; hierzu muss für MongoDB `MongoDBDatastore Provider` verwendet werden, der Dialect spezifiziert die Kommunikation zwischen Hibernate und der Datenbank, für MongoDB wird `MongoDBDialect` verwendet. (Leonard, 2013)
Konfiguriert wird Hibernate OGM wieder in einer XML Datei, was auch im Codebei-

spiel 5.11 zu sehen ist.

```
<dependency>
  <groupId>org.hibernate.ogm</groupId>
  <artifactId>hibernate-ogm-mongodb</artifactId>
</dependency>
```

Listing 5.11: Hibernate OGM

Der größte Benefit, den der Wechsel von Node.js zu Spring liefert, ist die Performanceerhöhung. Asynchronität hat diverse Vorteile, doch sind einzelne Aktionen von anderen abhängig, kann es schnell zu einem Problem hinsichtlich der Performance kommen. Durch diverse Callbacks wird auch die Wartbarkeit des Codes erschwert, da dieser sehr unleserlich wird. Ein weiteres Problem bezüglich des Single Thread Ansatzes (beschrieben in Kapitel 4.4.3) ist, dass Clientanfragen warten müssen, bis die vorherigen Anfragen abgearbeitet wurden.
Aufgrund der losen Typisierung in JavaScript und des nicht vorhandenen Kompiliervorganges sind Node-Anwendungen relativ anfällig, was Programmierfehler und die damit einhergehende fehlende Stabilität der Anwendung betrifft.(Shelat, 2016)

5.2.4 Conclusio

Zusammenfassend bieten diese Technologie-Stacks speziell für JavaScript erfahrene Entwickler den Vorteil, dass sowohl serverseitig, als auch clientseitig JavaScript bzw. TypeScript zum Einsatz kommt und dass mit MongoDB die Datenbank als BSON Datenformat, JSON sehr ähnlich ist. Reicht zu Beginn Node.js als Back-End und leichtgewichtiges Framework aus, so kann speziell diese Server-Technologie bei großen Nutzerzahlen an ihre Grenzen stoßen.

Der Tausch von Node.js hin zu Spring am Server ist bezüglich der damit einhergehenden Änderung der Programmiersprache doch eine relativ aufwändige, da nichts vom vorhandenen Code beibehalten werden kann. Die Verwendung einer NoSQL Datenbank konnte früher noch als Grund für die Vermeidung von Spring als Argument herangezogen werden, doch gibt es mittlerweile mit Hibernate OGM eine Technologie, die ähnlich arbeitet wie ein ORM bei relationalen Datenbanken.

Vorteile	Nachteile
dynamische GUI	Sicherheitseinstellungen mit Node.js aufwendiger
React kann auch mit wachsender Komplexität umgehen	Single Thread by Node kann zu Performanceprobleme führen
Einheitliche Programmiersprache	NoSQL als Datenbanksystem nicht so weit verbreitet
Hibernate auch mit NoSQL kompatibel	
einfaches Routing Node.js	

Tabelle 5.2: Gegenüberstellung Technologie-Stack 2

5.3 Full Stack mit JHipster - React / Spring / MySQL

Nachfolgend wird in diesem Kapitel eine Software-Architektur genauer analysiert, welche als Front-End React und als Back-End Spring mit jeglichen Zusatzmodulen und als Datenbank MySQL verwendet. Wie schon in den Kapiteln davor wird als ORM Hibernate und für die Absicherung der Anwendung wird Spring Security verwendet.

Als Datenbanksystem fungiert in diesem Stack MySQL, das laut der Trendanalyse welche in Bild 5.2 zu sehen ist, das beliebteste Datenbanksystem von denen in dieser Arbeit beschriebenen ist. Das Einbinden von MySQL in diese Umgebung wird dank Hibernate, ähnlich wie PostgreSQL in Kapitel 5.1, relativ unkompliziert vonstattengehen.

Anders als in der Architekturanalyse davor, wird in diesem Teil der Praxisarbeit keine Technologie durch eine andere ersetzt, sondern es wird gezeigt, wie dieser Stack mittels JHipster automatisch initialisiert werden kann und welche Vorteile damit einhergehen.

Auf die Bewertung des Client-Frameworks sowie des Server-Frameworks wird bewusst verzichtet, da diese Technologien bereits ausreichend analysiert wurden und die Stärken und Vorzüge von MySQL im Kapitel 4.3.1 beleuchtet wurden, wird diesmal die Einzelanalyse hinsichtlich auf die Indikatoren übersprungen.

5.3.1 Technologie-Stack

Um ausgehend von React eine Serververbindung herzustellen, muss diese wie in Listing 5.9 entwickelt werden, während der dort angegebene Endpunkt am Server erstellt werden muss. Spring allein unterstützt standardmäßig kein Routing. Aktiviert wird dieses Feature, wenn Spring MVC innerhalb des Back-Ends angewendet wird; ein solcher Routingpunkt wird in Listing 5.1 beispielhaft gezeigt.

Aufgrund dessen, dass der Großteil der Clientanfragen eine Datenbankoperation beinhalten, wird serverseitig auch eine Datenbankverbindung nötig sein. Um eine solche Verbindung zu konfigurieren und mit dieser innerhalb von Spring arbeiten zu können, wird sowohl Spring Boot, als auch Hibernate verwendet. Während Spring Boot um die Konfiguration der Module kümmert, wird mit Hibernate eine Kommunikation mit MySQL ermöglicht. Wie eine solche Konfiguration aussehen kann, zeigt der Codeblock 5.12.

```
<dependencies>
 <dependency>
  <groupId>hibernate</groupId>
   <artifactId>hibernate-entitymanager</artifactId>
  </dependency>
  <dependency>
  <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
  </dependency>
</dependencies>
```

Listing 5.12: Hibernate und MySQL

Eine weitere Möglichkeit, um Clientanfragen abzuarbeiten, sind, wie Kapitel 4.1.2.1 beschrieben, Servlets, mithilfe derer es auch möglich ist, direkt aus dem HTML, ohne Client-Framework, mit dem Server zu kommunizieren.
Mitgeliefert wird diese Funktion innerhalb von Spring mit Spring MVC; als Hauptcontroller dient das sogenannte `DispatcherServlet`, jegliche weiteren Servlet Endpunkte werden innerhalb der `web.xml` Datei definiert. (Mak, Rubio & Long, 2010)

Wie der Prozess bei Servlets abläuft, wird im Folgenden beschrieben. Das oben ge-

nannte `DispatcherServlet` fängt Anfragen vom Client ab und weist diese den passenden Endpunkten zu, welche die Logik beinhalten und die gewünschte Aktion durchführen. In Codebeispiel 5.13 wurde ein Servlet mit dem Namen *masterarbeit* definiert, welche auf den Pfad */helloWorld* verweist und im Hintergrund die Java Klasse `at.campus02.masterarbeit` aufruft.(Bretet, 2016)

```
<servlet>
  <servlet-name>masterarbeit</servlet-name>
  <servlet-class>at.campus02.masterarbeit</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>masterarbeit</servlet-name>
  <url-pattern>/helloWorld</url-pattern>
 </servlet-mapping>
</servlet>
```

Listing 5.13: Servlets XML

Aufgerufen werden kann dieses Servlet nun direkt innerhalb von HTML, beispielsweise mittels eines from Tags. Dazu muss dem `action` Attribut der Name, welcher zuvor in der `web.xml` definiert wurde, mitgegeben werden.

```
<form name="helloWorldForm" action="masterarbeit" method="↩
   POST">
```

Listing 5.14: Servlets Aufruf

Um ein Bild davon zu bekommen, wie Hibernate mit relationalen Datenbanken arbeitet, zeigen Listing 5.15 und 5.16 exemplarisch den Ablauf innerhalb von Spring. Im Codebeispiel 5.15 wird ein Abbild der Tabelle *country* als Java-Objekt erstellt. Während das Feld *name* später manuell eingefügt werden muss, wird die *Id* mittels der

`@GeneratedValue` automatisch erstellt.

```
@Entity
@Table
public class Country{
 @Id
 @GeneratedValue
 private Integer id;
 private String name;
 public Country (Integer id, String name) {
  this.id = id;
  this.name = name;
}
```

Listing 5.15: MySQL Hibernate

Wie ein `insert` Statement im Rahmen von Hibernate aussehen kann, wird im unten angeführten Beispiel 5.16 aufgezeigt. Es wird - wie in Java üblich - ein neues Objekt, `country`, definiert und diesem werden Werte zugewiesen und die beiden Befehle `save` und `getTransaction().commit()` fügen diesen Datensatz nun in die Datenbank ein.

In den letzten beiden Zeilen in Listing 5.16 ist zu sehen, dass Spring auf ein `session` Objekt zugreift, welches ein Hibernate `Session-Factory` Objekt ist. Bei der `Session -Factory` handelt es sich um schwergewichtiges Objekt, welches, bestenfalls einmalig, zu Beginn initialisiert werden sollte, da es jegliche Konfigurationseinstellungen, welche für die Datenbank benötigt werden, beinhaltet. (Stark, 2005)

Diese Beispiele zeigen, dass somit Java EntwicklerInnen Datenbankoperationen ohne SQL-Kenntnisse durchführen können, was wiederum der Time-To-Market zugute kommt, da abgesehen von einem Grundverständnis für Datenbanken bzw. SQL seitens der ProgammiererInnen kein detailliertes Wissen vorhanden sein muss.

,

```
1 Country country = new Country();
2 country.name("Austria");
3 session.save(country);
4 session.getTransaction().commit();
```

Listing 5.16: MySQL Hibernate Insert

Um die Applikation vor Angriffen von außen zu schützen, wird bei dieser Architektur Spring Security im Back-End initialisiert. Wie man sich mithilfe von Spring Security respektive Spring MVC und React vor diversen möglichen Attacken schützen kann, wurde in den Kapiteln 5.1.2 sowie 5.2.2 bereits beleuchtet.

Stabilität und Ausfallssicherheit ist soweit gegeben, dass Spring sehr darauf bedacht ist, den Programmcode testbar, lesbar und wartbar zu halten. Durch den Modularen Aufbau von Spring sind die einzelnen Programmteile schon von Beginn an gut voneinander getrennt, was wiederum das Tauschen von Programmteilen innerhalb des Back-Ends einfacher macht. Durch die Trennung von Konfigurationscode und tatsächlichem Programmcode, sinkt auch die Gefahr, dass ProgrammiererInnen versehentlich Konfigurationen umschreiben bzw. diese beiden Teile zu vermischen beginnen.
Zusätzlich dazu ist mit React auf der Front-End Seite ein stabiles Framework vorhanden, welches speziell für große Datenmengen konzipiert wurde.

Dieser Technologie-Stack beinhaltet, laut Grafik 5.1 und 5.2, clientseitig das derzeit beliebteste Community Framework und mit MySQL das gefragteste Datenbanksystem. Zusätzlich kommt mit Spring ein mächtiges Tool im Back-End zum Einsatz, welches über ein enormes Featureset verfügt.
Hinsichtlich auf Zukunftssicherheit, Stabilität und Performance wird diese Architektur nur in den geringsten Fällen an ihre Grenzen stoßen.

5.3.2 Aufbau mit JHipster

Mussten sich in den vorangegangen Architekturen EntwicklerInnen sowohl um die Installation des Front-End, Back-End und des Datenbanksystems kümmern, wird in diesem Kapitel mittels JHipster dieser Vorgang von einem Tool automatisiert. Ne-

ben der automatischen Installation übernimmt JHipster zusätzlich jegliche Konfigurationsarbeiten, welche nötig sind, damit die einzelnen Komponenten auch lauffähig sind.

Gestartet wird der gesamte Initialisierungsprozess mittels JHipster und dem Kommandozeilenbefehl `yo jhipster`. Wurde dieser ausgeführt, wird man Schritt für Schritt durch die Installation geführt. Innerhalb von wenigen Minuten ist der Installationsprozess abgeschlossen und JHipster hat ein Grundgerüst für Front-End und Back-End aufgebaut; zusätzlich dazu wurden auch diverse Konfigurationseinstellungen in die `pom.xml` geschrieben und Spring Boot zusätzlich aufgesetzt.

Zu Beginn muss angegeben werden, um welche Art von Applikationsarchitektur es sich handelt - um eine Monolithen oder eine Microservice-Architektur - aufbauend auf diese Entscheidung gibt es individuelle, für diese Arbeit aber nicht relevante, Konfigurationsoptionen, welche von JHipster abgefragt werden.
Abbildung 5.3 zeigt, dass JHipster gleich zu Beginn Werkzeuge für die Skalierung und Überwachung der Applikation einbauen möchte. Diese Option ist gerade für wachsende Anwendungen interessant, da somit gleich ein Tool gegeben ist, welches das Wachstum kontrolliert. Ein solches Monitoring beeinflusst wiederum die Indikatoren Stabilität und Ausfallsicherheit, da das System relativ gut überwacht wird.

```
? Do you want to use the JHipster Registry to configure, monitor and scale your
application? Yes
```

Abbildung 5.3: JHipster Scaling
Quelle: eigene Darstellung

Für die Vorkonfiguration einer Datenbankverbindung bzw. das Schreiben der Konfigurationseinstellungen in die `pom.xml` fragt JHipster, wie in Grafik 5.5 abgebildet, welches Datenbanksystem im Hintergrund laufen soll. Möglich wären sowohl NoSQL Systeme wie MongoDB, als auch - wie in diesem Technologie-Stack verwendet - relationale Datenbanken wie MySQL. Wurde MySQL als Datenbank gewählt, werden jegliche Vorkehrungen getroffen damit das Back-End mit dieser Datenbank arbeiten kann.

Bezüglich Sicherheit arbeitet JHipster laut deren Website[6] mit Spring Security; somit können EntwicklerInnen damit die Applikation vor möglichen Attacken schützen.

[6] https://www.jhipster.tech/

```
? Which *type* of database would you like to use? SQL (H2, MySQL, MariaDB, Postg
reSQL, Oracle, MSSQL)
? Which *production* database would you like to use? MySQL
? Which *development* database would you like to use? MySQL
```

Abbildung 5.4: JHipster Datebank
Quelle: eigene Darstellung

Zusätzlich dazu kann bei der Installation auch schon der Authentication Typ gewählt werden.
Wurde eine Datenbank gewählt, wird als nächstes das Front-End bestimmt. Hier würden alle drei Front-End Frameworks dieser Arbeit zur Verfügung stehen. Da in dieser Architektur die Wahl auf React gefallen ist, wird in Abbildung 5.6 React gewählt. Wird React ausgewählt, legt JHipster im Hintergrund alles dafür Nötige an. Darunter auch `webpack`, um das React-Programm lauffähig zu machen.

```
? Which *type* of authentication would you like to use?
```

Abbildung 5.5: JHipster Authentication
Quelle: eigene Darstellung

Ein weiteres sehr praktisches Feature, welches mit JHipster mitgeliefert wird, ist, dass eine vollumfängliche Testumgebung mitaufgesetzt wird. Zusätzlich zu der standardmäßigen Testumgebung, können ProgrammiererInnen noch weitere Testframeworks verwenden. Eine solche Testabdeckung, welche von Beginn an gegeben ist, erhöht die Qualität der Anwendung enorm, da von Start weg jegliche Programmteile prüfbar sind.

```
? Which *Framework* would you like to use for the client? React
```

Abbildung 5.6: JHipster Front-End
Quelle: eigene Darstellung

Wurde der Installationsvorgang abgeschlossen, wurde ein lauffähiges Grundgerüst für eine Web-Anwendung angelegt und JHipster gibt auch gleich Hinweise darauf, wie die einzelnen Programmteile ausgeführt werden können (siehe Abbildung 5.8).

JHipster bietet gerade für unerfahrene EntwicklerInnen ein hervorragendes Toolset an, um ohne großen Aufwand ein/e Grundgerüst und –konfiguration mit wenig Vor-

```
? Besides JUnit and Jest, which testing frameworks would you like to use?
```

Abbildung 5.7: JHipster Testing
Quelle: eigene Darstellung

```
Server application generated successfully.

Run your Spring Boot application:
./mvnw

Client application generated successfully.

Start your Webpack development server with:
 npm start
```

Abbildung 5.8: JHipster Fertig
Quelle: eigene Darstellung

wissen zu erstellen, mit welchem man sodann eine moderne Web-Anwendung entwickeln kann. Die komplette Testabdeckung bringt den Vorteil, dass von Beginn an ein stabiler Code geschrieben wird und somit die Qualität der Anwendung erheblich steigt.

Für erfahrene EntwicklerInnen stellt sich die Frage, ob es tatsächlich gewünscht ist, dass die ganze Projektumgebung automatisch angelegt wird und man nicht über jegliche Konfigurationsschritte Bescheid weiß. Gerade in späteren Entwicklungsschritten könnte es eventuell zu Komplikationen kommen, wenn sich JHipster-spezifische Konfigurationen mit denen von UserInnen in die Quere kommen.
Abgesehen von diesen Gegenargumenten bietet JHipster bezüglich Stabilität, Time-To-Market und Security ein sehr nützliches und brauchbares Grundgerüst an.

5.3.3 Conclusio

Das Fazit über diesen Tech-Stack fällt durchwegs positiv aus; sowohl server,- als auch clientseitig werden zwei stabile und zuverlässige Frameworks verwendet. Diese bringen einen Funktionsumfang mit, der voll ausreichend für eine Web-Anwendung ist. Abgesichert mit Spring Security und der problemlosen Datenbanknutzung durch Hibernate sind auch wesentliche Punkte innerhalb der Architektur erfüllt.

Die Möglichkeit, mithilfe einer Software wie JHipster ein komplettes Grundgerüst für eine Web-Anwendung aufzusetzen, ist eine sehr interessante und nützliche. Mit wenigen Angaben während der Installation ist in Kürze ein System konfiguriert, welches sowohl mit einer Datenbank kommunizieren kann, als auch vollumfänglich prüfbar ist. Durch die einheitliche Initialisierung ist ein funktionierendes Zusammenspiel der Komponenten wahrscheinlicher, als es würden EntwicklerInnen jedes Framework individuell aufsetzen und die Konfigurationen manuell durchführen.

Vorteile	**Nachteile**
Vollumfängliche Testbarkeit durch JHipster	JHipster legt einige Zusatzdateien an, die die Projektstruktur umfangreicher machen
zukunftssicher	Durch JHipster Initialisierung kann das spätere Einbinden von Frameworks aufwändiger werden
gute Skalierung	EntwicklerInnen kennen die Konfiguration im Detail nicht
JHipster reduziert Time-To-Market	
vereinfachte Konfiguration durch JHipster	

Tabelle 5.3: Gegenüberstellung Technologie-Stack 3

6 Resümee

Ziel dieser Arbeit war es, herauszufinden, ob es möglich ist, einen Leitfaden zu erstellen, um die Entwicklung einer Web-Applikation zu optimieren. Um fundierte Kenntnisse dahingehend zu gewinnen, wurden für die einzelnen Layer ausgewählte Frameworks bezüglich zuvor definierten Indikatoren geprüft. Im ersten Schritt wurden die Technologien individuell betrachtet und analysiert, während daraufhin das Zusammenspiel der Komponenten getestet und überprüft wurde. Zusätzlich wurde im Praxisteil in den ersten zwei Technologie-Stacks jeweils ein Framework gegen ein anderes ersetzt und schließlich wurde evaluiert, ob und für wen ein solcher Tausch Sinn machen würde. Im Rahmen des dritten Experiments wurde kein Technologietausch vorgenommen, sondern es wurde mittels JHipster die ganze Architektur automatisch aufgebaut.

Fazit ist, dass die Entwicklung eines generellen Leitfadens aufgrund der zu unterschiedlichen Anforderungen an die jeweils zu entwickelnden Applikationen nicht möglich zu sein scheint. Da gerade die individuellen Stärken der EntwicklerInnen extrem unterschiedlich sind und auch die Anforderungen an die Anwendung selbst variieren. Clientseitig existieren am Markt fast ausschließlich Technologien, welche auf JavaScript basieren, weshalb gerade Programmiererfahrung in dieser Sprache im Kontext der Webentwicklung unausweichlich ist.

Was sich im Rahmen dieser Arbeit noch gezeigt hat ist, dass mit Spring im Back-End ein universell einsetzbares Werkzeug existiert, welches viele essentielle Kriterien erfüllt und durch Zusatzmodule wie Spring MVC, Spring Security und Spring Boot erweitert werden kann. Mit Hibernate ist eine weitere Technologie am Markt, welche das Arbeiten mit Datenbanken in Kombination mit Spring simplifiziert. Spring MVC ermöglicht es zusätzlich ohne viel Aufwand Routingpunkte zu erzeugen, welche von unterschiedlichen Client-Frameworks angesprochen werden können.
Anders als im Back-End kann sich im Front-End keine der drei Technologien wirk-

lich von den anderen abheben. In diesem Layer kann keine eindeutige Empfehlung für ein Framework gegeben werden, da Vue, React und Angular vom Prinzip her relativ ähnlich aufgebaut sind. Für Single-Page Anwendungen würde sich Vue oder React aufgrund der Leichtgewichtigkeit eher anbieten als Angular, da diese Technologie eher bei komplexen und umfangreichen Projekten zum Einsatz kommt.
Bezüglich des Datenbanksystems zeigt sich, dass die grundlegende Entscheidung dahingehend fallen muss, ob eine relationale oder eine NoSQL Datenbank verwendet werden soll. Aufgrund der Verbreitung und den verfügbaren Schnittstellen bieten sich nach wie vor relationale Datenbanken an, da NoSQL Systeme erst bei Big Data Systemen wirklich Vorteile bringen.

Abschließend lässt sich sagen, dass es gerade im Front-End nicht möglich ist, eine einheitliche Handlungsempfehlung abzugeben. Speziell innerhalb dieses Layers sind die Unterschiede der einzelnen Frameworks relativ gering, sodass hier kein Framework signifikante Vorteile gegenüber den anderen bietet.
Dafür existiert mit Spring schon seit geraumer Zeit eine Technologie, welche ohne Bedenken empfohlen werden kann. Aufgrund dessen, kann - im Gegensatz zum Front-End - im Back-End gesagt werden, dass Spring als Technologie präferiert werden könnte.
Hinsichtlich der Faktoren, bezüglich welcher die Frameworks in dieser Arbeit getestet wurden, kann gesagt werden, dass die Zukunftssicherheit im Front-End schwieriger zu bewerten ist als im Back-End. Austauschbarkeit, Sicherheit und Stabilität sind Kriterien, welche nur innerhalb der Gesamtarchitektur bewertet werden können.

Bezugnehmend auf die Forschungsfrage kann gesagt werden, dass eine einheitliche Handlungsempfehlung für die Entwicklung einer Web-Anwendung nicht möglich ist. Speziell im Front-End ist es schwierig, eines der drei analysierten Frameworks hervorzuheben. Gibt es im Back-End mit Node.js für reine JavaScript-EntwicklerInnen eine Alternative, kann innerhalb dieses Layers aber Spring inklusive den mitgelieferten Zusatzmodulen empfohlen werden. Auf Datenbankebene empfiehlt es sich - sollte es sich nicht um eine Big Data Anwendung handeln - auf eine relationale Datenbank zu setzen, da hier beispielsweise mit Hibernate ein hervorragendes Tool existiert, welches das Arbeiten mit Daten erheblich erleichtert.

Eine Handlungsempfehlung zur Hilfe zu nehmen, wird die Effizienz des Entwicklungsprozesses positiv beeinflussen und auch die Übersichtlichkeit über die einzelnen Technologien wird erhöht, da ohne aufwendige Recherchearbeiten ein Überblick

über die einzelnen Technologien sowie deren Schnittstellen geschaffen wird. Kristallisierte sich im Back-End mit Spring eine Technologie heraus, ist es im Front-End nicht mögliche eine Technologie den anderen den anderen vorzuziehen. Hier obliegt die Entscheidung, welches Framework eingesetzt werden soll, der/dem EntwicklerIn .

Node.js	**Spring**
Basiert auf JavaScript	zukunftssicher, lange Marktpräsenz
Ansatz der Asynchronität - kann zu Race Condition führen	Spring Boot, MVC und Security einfach erweiterbar
Single Thread Ansatz - kann zu Performance Problemen führen	Hibernate leichte Datenbankanbindung und -verwaltung
Sicherheitseinstellungen müssen selbst erstellt werden	Durch Spring Security leichte Sicherheitskonfiguration
einfaches Routing Node.js	einfaches Routing mit Spring MVC

Tabelle 6.1: Gegenüberstellung Back-End

React	**Angular**	**Vue.js**
Single Page Entwicklung möglich	Single Page Entwicklung möglich	Single Page Entwicklung möglich
Dynamische Änderung der Seite	Dynamische Änderung der Seite	Dynamische Änderung der Seite
DOM Manipulation möglich	DOM Manipulation möglich	DOM Manipulation möglich
Erstellung eigener Komponenten möglich	Erstellung eigener Komponenten möglich	Erstellung eigener Komponenten möglich
Basiert auf JS - keine Typisierung führt eventuell zu schlechtem Code	Basiert auf TypeScript - besitzt Typisierung, Code wird lesbarer und bei komplexen Systemen sinkt damit die Fehlerrate	Basiert auf JS - keine Typisierung führt eventuell zu schlechtem Code

Tabelle 6.2: Gegenüberstellung Front-End

MySQL	PostgreSQL	MongoDB
Relationales Datenbanksystem	Relationales Datenbanksystem	NoSQL Datenbanksystem
GNU Lizenz, darf nicht im Softwarebundle ausgeliefert werden	Open-Source-Lizenz	Open-Source-Lizenz
Darstellung in tabellarischer Form	Darstellung in tabellarischer Form	Darstellung in BSON

Tabelle 6.3: Gegenüberstellung Datenbanken

Glossar

Annotation	Erlauben es Metadaten Informationen in den Sourcecode zu übergeben.
API Key	Ist ein Code der von Anwendungen an Schnittstellen übergeben wird, um sich zu identifizieren.
Bottleneck	Komponent innerhalb einer Software, welcher unter einer bestimmter Last das System verlangsamt.
Bower	Ist ein Paketmanager für clientseitige Webentwicklung.
BrowserSync	Ist ein Werkzeug für die Synchronisation des Browsers in Echtzeit über mehrere Geräte; ebenfalls wird automatisch eine Aktualisierung angestoßen wenn Daten geändert werden.
Cache	Cache wird in der EDV ein schneller Puffer-Speicher bezeichnet.
case-sensitive	Bedeutet, dass eine Unterscheidung von Groß- und Kleinschreibung gemacht wird.
Dependency Injection	Ein Entwurfsmuster in der objektorientierten Programmierung welches die Abhängigkeit einzelner Objekte zur Laufzeit regelt.
GUI	Graphical User Interface (GUI); Versteht man die grafische Benutzeroberfläche, mit welcher die/der UserIn interagiert.

I/O	Input/Output (I/O); Bezeichnet die Interaktion eines System mit anderen Systemen. Input sind Eingaben vom System, Output sind Ausgaben vom externen System.
Java	Java ist eine objektorientierte Programmiersprache von der Firma Oracle.
JavaScript	Eine Skriptsprache, die hauptsächlich in Web-Browsern eingesetzt wird.
JQuery	Ist eine JavaScript Bibliothek, mit welcher man DOM Elemente manipulieren/ändern kann.
JSON	JavaScript Object Notation, ist ein kompaktes Datenformat.
Linter Tool	Eine Software für statische Code-Analyse.
Monolith	Alle Elemente befinden sich in einem einzigen und untrennbaren Gebilde.
MVC	Model View Controller (MVC); Unterteilung einer Software in die drei Komponenten Daten (model), Präsentation (view) und Steuerung (controller).
O-Auth	OAuth bezeichnet zwei offene Protokolle, die eine standardisierte API-Autorisierung erlauben.
OpenID	Ein dezentrales Authentifizierungssystem für webbasierte Dienste, dieses ermöglicht es, dass User-Innen sich nach einer einmaligen Anmeldung im System, bei sich im System befindlichen Websites, nicht nochmals anmelden müssen; läuft also nach dem Single-Sign-on Prinzip.
PHP	Eine Skriptsprache, hauptsächlich zur Erstellung dynamischer Webseiten oder Webanwendungen verwendet wird.
pom.xml	Eine Datei welche Informationen über das Projekt und die Konfigurationsdetails enthält.

responsive Design	Begriff für das Design einer Websites welches flexibel gestaltet ist, dass Funktion, Design und Inhalt sich der jeweiligen Bildschirmauflösung des verwendeten Desktop, Tablet oder Smartphones anpassen.
routing	Das Routing ist ein Vorgang, der den Weg zur nächsten Station eines Datenpakets bestimmt.
Servlet	Sind Java-Klassen, deren Instanzen innerhalb eines Webservers Anfragen von Clients entgegennehmen und beantworten.
Single Page Application	Wird eine Webanwendung bezeichnet, die aus einem einzigen HTML-Dokument besteht und deren Inhalte dynamisch nachgeladen werden.
SSH Key	Mit deren Hilfe kann auf sichere Art und Weise eine verschlüsselte Netzwerkverbindung mit einem entfernten Gerät hergestellt werden.
Templating	Ein Dokument welches eine vordefinierte Struktur beinhaltet, welche als Ausgangspunkt für Anwendungen dient und man einzelnen Elemente dynamisch ändern/befüllen kann.
Thread	Ausführungsreihenfolge während der Verarbeitung einer Anwendung.
Tupel	Datenzeile innerhalb einer Datenbank.
Typescript	Ist eine von Microsoft entwickelte Programmiersprache und erweitert JavaScript um bestimmte Features.
Webbrowser	Ein Programm zur Darstellung von Websites.
Webpack	Ein Opensource-JavaScript-Modul-Packer, mit welchem man JavaScript-Dateien für die Nutzung im Browser zusammenzuführen und zu einer Datei bündeln kann.

Webseite	Sammelbegriff für Dateien die gemeinsam dargestellt werden. Zum Beispiel XHTML-Dokument mit Bildern, CSS und JavaScript-Elementen,
Website	Eine Menge von Dateien im WWW, die organisatorisch einer Person/Organisation/Gruppe zugeordnet sind.
Wörterbuchattacke	Ist eine Methode bei der Mithilfe von Passwortlisten versucht wird ein unbekanntes Passwort herauszufinden.

Akronyme

AJAX	Asynchronous JavaScript and XML
AOP	Aspektorientierte Programmierung
API	Application Programming Interface
BSON	Binary JSON
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DDL	Data Definition Language
DML	Data Manipulation Language
DOM	Document Object Model
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I/O	Input/Output
J2EE	Java 2 Enterprise Edition
JDBC	Java Database Connectivity
JEE	Java Platform, Enterprise Edition
JSX	Javascript XML oder Javascript Syntax Extension
MVC	Model View Controller

ODM	Object Document Mapping
OGM	Object Grid Mapper
OLAP	Online Analytical Processing
OLTP	Online Transactional Processing
ORM	Object Relational Mapper
OS	Operating System
OWASP	Open Web Application Security Project
POJO	Plain old Java objects
RDBMS	Relational Database Management System
SQL	Structured Query Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URL	Uniform Resource Locator
WWW	Word Wide Web
XML	Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

Altmann, T. (2018, März). *7 gründe für jhipster.* Zugriff am 04. Okt. 2019 auf https://www.solvistas.com/blog/7-gruende-fuer-jhipster/

Anil, S. (2018). *Angular interview questions and answers- including angular 6,5 ,4 and 2*. BPB Publications.

Aquino, C. & Gandee, T. (2016). *Front-end web development: The big nerd ranch guide*. Pearson Education.

Augsten, S. (2017, April). *Was ist eine web app?* Zugriff am 30. Mai. 2019 auf https://www.dev-insider.de/was-ist-eine-web-app-a-596814/

Augsten, S. (2019, Juni). *Was ist das spring framework.* Zugriff am 30. Sep. 2019 auf https://www.dev-insider.de/was-ist-das-spring-framework-a-829846/

Banks, P. (2017). *Learning react - functional web development with react and redux*. Sebastopol: O'Reilly Media, Inc.

Bauer, G. (2009). *Architekturen für web-anwendungen: Eine praxisbezogene konstruktionssystematik*. Vieweg+Teubner Verlag.

Bretet, A. (2016). *Spring mvc cookbook*. Packt Publishing.

Brown, E. (2016). *Learning javascript: Javascript essentials for modern application development* (3rd Aufl.). O'Reilly Media, Inc.

Bulatovych, D. (2019). *Choosing a tech stack for the full-cycle web application development.* Zugriff am 17. Okt. 2019 auf https://yalantis.com/blog/tech-stack-for-web-app-development/

Burkhart, M. (2019, September). *Owasp top 10 der api-sicherheit: Airlock bringt whitepaper zur liste.* Zugriff am 22. Okt. 2019 auf https://computerwelt.at/news/kommentar/owasp-top-10-der-api-sicherheit-airlock-bringt-whitepaper-zur-liste/

Chiarelli, A. (2018). *Beginning react - simplify your frontend development workflow and enhance the user experience of your applications with react* (1. Aufl. Aufl.). Birmingham: Packt Publishing Ltd.

Chodorow, K. (2013). *Mongodb: The definitive guide: Powerful and scalable data storage*. O'Reilly Media.

Chopra, V. (2015). *Websocket essentials – building apps with html5 websockets*. Packt Publishing.

Council, F. T. (2016, Februar). *Eight strategies to future-proof your technology.* Zugriff am 17. Okt. 2019 auf `https://www.forbes.com/sites/forbestechcouncil/2016/02/18/eight-strategies-to-future-proof-your-technology/#12f95ffa6ccf`

Dayley, B. (2014). *Node.js, mongodb, and angularjs web development*. Addison-Wesley Professional.

Deabes, S. (2017, September). *Introduction to web application performance.* Zugriff am 04. Juli. 2019 auf `https://dzone.com/articles/introduction-to-web-applications-performance`

Debnath, M. (2018, August). *What is spring security?* Zugriff am 01. Okt. 2019 auf `https://www.developer.com/java/ent/what-is-spring-security.html`

Deck, P. (2016). *Spring mvc: A tutorial* (2. Aufl. Aufl.). Brainy Software Inc.

Deepa, G. & Thilagam, P. S. (2016). Securing web applications from injection and logic vulnerabilities: Approaches and challenges. In *Information and software technology* (S. 160-180). Elsevier B.V. Selection.

Duuna, K. (2015). *Secure your node.js web application: Keep attackers out and users happy*. Pragmatic Bookshelf.

Elmasri, R. & Navathe, S. (2009). *Grundlagen von datenbanksystemen*. Pearson Deutschland.

Fernandes, Costa & Carrico. (2012). Evaluating the accessibility of web applications. In *Proceedings of the 4th international conference on software development for enhancing accessibility and fighting info-exclusion.* Elsevier B.V. Selection.

Fischer, S., Saddik, A. & Steinacker, A. (2013). *Open java: Von den grundlagen zu den anwendungen*. Springer Berlin Heidelberg. Zugriff auf `https://books.google.at/books?id=rc8jBgAAQBAJ`

Flanagan, D. (2011). *Javascript - the definitive guide* (6. Aufl. Aufl.). O'Reilly Media, Inc.

Franz, K. (2007). *Handbuch zum testen von web-applikationen: Testverfahren, werkzeuge, praxistipps*. Springer Berlin Heidelberg.

Freeman, A. (2019). *Essential angular for asp.net core mvc 3: A practical guide to successfully using both in your projects*. Apress.

Ganeshan, A. (2016). *Spring mvc: Beginner's guide* (2. Aufl. Aufl.). Birmingham: Packt

Publishing Ltd.

Gilmore, W. J. (2010). Beginning php and mysql | |. In (Bd. 10.1007/978-1-4302-3115-8). doi: 10.1007/978-1-4302-3115-8

Gonçalves, L. (2019, März). *Time to market: How your company can keep launching new products and features ahead of your competition.* Zugriff am 17. Okt. 2019 auf https://luis-goncalves.com/time-to-market/

Grundlegendes zu webanwendungen. (2017). Zugriff am 07. Juli. 2019 auf https://helpx.adobe.com/de/dreamweaver/using/web-applications.html

Gull, C. (2012). *Web-applikationen entwickeln mit nosql: Das buch für datenbank-einsteiger und profis!* Franzis Verlag.

Halin, A. A. M. D. X. P. G. H. P., Axel; Nuttinck. (2017). [acm press the eleventh international workshop - eindhoven, netherlands (2017.02.01-2017.02.03)] proceedings of the eleventh international workshop on variability modelling of software-intensive systems - vamos '17 - yo variability! jhipster.. doi: 10.1145/3023956.3023963

Halliday, P. (2018). *Vue.js 2 design patterns and best practices: Build enterprise-ready, modular vue.js applications with vuex and nuxt*. Packt Publishing.

Harringer, M. (2018, November). *Die geschichte der website-entwicklung in einer infografik.* Zugriff am 30. Mai. 2019 auf https://page-online.de/tools-technik/die-geschichte-der-website-entwicklung-in-einer-infografik/

Herron, D. (2018). *Node.js web development - server-side development with node 10 made easy, 4th edition* (4. Aufl. Aufl.). Packt Publishing Ltd.

Hughes-Croucher, T. & Wilson, M. (2012). *Node: Up and running: Scalable server-side code with javascript*. O'Reilly Media.

Jablonski, S., Petrov, I., Meiler, C. & Mayer, U. (2004). *Guide to web application and platform architectures*. Springer Berlin Heidelberg.

Jin, B., Sahni, S. & Shevat, A. (2018). *Designing web apis - building apis that developers love*. O'Reilly Media, Inc.

Juba, S. & Volkov, A. (2017). *Learning postgresql 10 - a beginner's guide to building high-performance postgresql database solutions* (2. Aufl. Aufl.). Packt Publishing Ltd.

Knutson, M., Winch, R. & Mularien, P. (2017). *Spring security: Secure your web applications, restful services, and microservice architectures*. Packt Publishing.

Kofler, M. (2013). *Die java-syntax*. ebooks.kofler.

Kumar, A. & Singh, R. K. (2015). Comparative analysis of angularjs and reactjs. In *International journal of latest trends in engineering and technology.*

Ladd, S. (2006). *Expert spring mvc and web flow* -. Apress.

Leonard, A. (2013). *Pro hibernate and mongodb*. Apress.

Liang, L. (2017). Express supervision system based on nodejs and mongodb..

Lombardi, A. (2015). *Websocket*. O'Reilly Media.

Long, J., Lui, M., Gray, M. & Chan, A. (2011). *Pro spring integration*. Apress.

Macrae, C. (2018). *Vue.js: Up and running - building accessible and performant web apps*. O'Reilly Media, Inc.

Mak, G., Rubio, D. & Long, J. (2010). *Spring recipes: A problem-solution approach*. Apress.

Mardan, A. (2015). *Full stack javascript - learn backbone.js, node.js and mongodb* (1st ed. Aufl.). New York: Apress.

Mardan, A. (2018). *Practical node.js: Building real-world scalable web apps*. Apress.

Martin, S. (2019, Juli). *Angular vs react vs vue: Which is the best choice for 2019?* Zugriff am 24. Okt. 2019 auf `https://hackernoon.com/angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847`

Matam, S. & Jain, J. (2017). *Pro apache jmeter: Web application performance testing*. Apress.

Mead, A. (2018). *Advanced node.js development - master node.js by building real-world applications* (1. Aufl. Aufl.). Birmingham: Packt Publishing Ltd.

Meier, A. & Kaufmann, M. (2016). *Sql- & nosql-datenbanken*. Springer Berlin Heidelberg.

Minter, D., Linwood, J. & Ottinger, J. (2014). *Beginning hibernate* - (3. Aufl. Aufl.). Apress.

Mumar, S. (2013). *Spring and hibernate*. Tata McGraw-Hill Education.

Nelson, B. (2018). *Getting to know vue.js - learn to build single page applications in vue from scratch* (1. Aufl. Aufl.). New York: Apress.

Neuhaus, J. (2017, August). *9 steps: Choosing a tech stack for your web application.* Zugriff am 17. Okt. 2019 auf `https://medium.com/unicorn-supplies/9-steps-how-to-choose-a-technology-stack-for-your-web-application-a6e302398e55`

Nixon, R. (2018). *Learning php, mysql & javascript - with jquery, css & html5*. Sebastopol: O'Reilly Media, Inc.

Northwood, C. (2018). *The full stack developer: Your essential guide to the everyday skills expected of a modern full stack web developer*. Apress.

Nowak, M. (2019, Juni). *Choosing a technology stack for your web application development.* Zugriff am 17. Okt. 2019 auf `https://www.monterail.com/blog/web-development-technology-stack`

Obe, R. & Hsu, L. (2012). *Postgresql: Up and running*. O'Reilly.

Okoi, M. D. (2019, April). *The 7 most popular programming languages on github in 2019.* Zugriff am 26. Okt. 2019 auf https://www.fossmint.com/popular-programming-languages-on-github/

Ottinger, J. B., Linwood, J., Minter, D., Linwood, J. & Minter, D. (2016). *Beginning hibernate - for hibernate 5* (4. Aufl. Aufl.). Apress.

Paraschiv, E. (2019, Oktober). *Control the session with spring security.* Zugriff am 25. Okt. 2019 auf https://www.baeldung.com/spring-security-session

Patil, J. (2016, Januar). *Prevent xss security threat in spring application.* Zugriff am 25. Okt. 2019 auf https://beansroasted.wordpress.com/2016/01/21/prevent-xss-security-threat-in-spring-application/

Plugge, E., Hawkins, P., Membrey, Plugge, E. & Hawkins. (2011). *The definitive guide to mongodb - the nosql database for cloud and desktop computing* (1. Aufl. Aufl.). Apress.

Pop, D.-P. & Altar, A. (2013). Designing an mvc model for rapid web application development. In *24th daaam international symposium on intelligent manufacturing and automation.* Elsevier Ltd.

Preiß, N. (2014). *Entwurf und verarbeitung relationaler datenbanken: eine durchgängige und praxisorientierte vorgehensweise*. De Gruyter.

Raible, M. (2016). *The jhipster mini-book*.

Rawindu, K. (2019). *Distributed systems.* Zugriff am 07. Juli. 2019 auf https://medium.com/@kushanrawindu/distributed-systems-9acfea481af

Regina Obe, L. H. (2017). *Postgresql: Up and running - a practical guide to the advanced open source database*. O'Reilly Media.

Santana, O. (2019, Juni). *Spring mvc and mongodb: A match made in platform.sh heaven.* Zugriff am 01. Okt. 2019 auf https://dzone.com/articles/spring-mvc-and-mongodb-a-match-made-in-platformsh

Sasidharan, K. N. (2018). *Full stack development with jhipster: Build modern web applications and microservices with spring and angular*. Packt Publishing.

Schadow, D. (2014). *Java-web-security: Sichere webanwendungen mit java entwickeln*. dpunkt.verlag.

Schildgen, J. (2016). *Mongodb kompakt: Was sie Über die NoSQL-Dokumentendatenbank wissen müssen, isbn=9783833497513, series=Datenbanken kompakt,*.

Seshadri, S. (2018). *Angular: Up and running - learning angular, step by step*. Sebastopol: O'Reilly Media, Inc.

Shahzad, F. (2017). Modern and responsive mobile-enabled web applications. In *The 12th international conference on future networks and communications*. Elsevier B.V. Selection.

Shelat, M. (2016, Mai). *Node.js for enterprise applications! are you kidding?* Zugriff am 06. Nov. 2019 auf https://dzone.com/articles/nodejs-for-enterprise-applications-are-you-kidding

Smith, S. (2015, Juni). *Secure node apps against owasp top 10 - authentication and sessions.* Zugriff am 05. Nov. 2019 auf http://scottksmith.com/blog/2015/06/15/secure-node-apps-against-owasp-top-10-authentication-and-sessions/

Soni, R. K. (2017). *Full stack angularjs for java developers*. Apress.

Sonmez, J. (2016, Dezember). *What is back-end development?* Zugriff am 17. Okt. 2019 auf https://simpleprogrammer.com/what-is-back-end-development/

Stark, T. (2005). *J2ee: Einstieg für anspruchsvolle*. Pearson Deutschland.

Stefanov, S. & Demmig, T. (2017). *Durchstarten mit react: Web-apps einfach und modular entwickeln*. O'Reilly.

Steyer, M. & Softic, V. (2015). *Angular js: Moderne webanwendungen und single page applications mit javascript*. O'Reilly Verlag.

Throll, M. & Bartosch, O. (2011). *Einstieg in sql*. Galileo Press.

Trelle, T. (2014). *Mongodb: Der praktische einstieg*. dpunkt.verlag.

Trelle, T. (2015). *Mongodb für software-entwickler.* Zugriff am 25. Sep. 2019 auf https://www.informatik-aktuell.de/betrieb/datenbanken/mongodb-fuer-software-entwickler.html

Tripathi, M. (2017, Dezember). *Understanding how the chrome v8 engine translates javascript into machine code.* Zugriff am 04. Okt. 2019 auf https://www.freecodecamp.org/news/understanding-the-core-of-nodejs-the-powerful-chrome-v8-engine-79e7eb8af964/

Uluca, D. (2018). *Angular 6 for enterprise-ready web applications - deliver production-ready and cloud-scale angular web apps* (1. Aufl. Aufl.). Birmingham: Packt Publishing Ltd.

Vizcarra, L. D. (2019, Mai). *Top 10 web security vulnerabilities to watch out for in 2019.* Zugriff am 22. Okt. 2019 auf https://cai.tools.sap/blog/top-10-web-security-vulnerabilities-to-watch-out-for-in-2019/

Vukadinovic, D. (2018, Juni). *Was ist der unterschied zwischen http und https?* Zugriff am 02. Juli. 2019 auf https://www.globalsign.com/de-de/blog/unterschied-zwischen-http-und-https/

Walls, C. (2012). *Spring im einsatz*. Carl Hanser Verlag GmbH Co KG.

Wetter, D. (2018, Januar). *Owasp top 10: Kritischer blick auf die charts.* Zugriff am 22. Okt. 2019 auf `https://www.heise.de/developer/artikel/OWASP-Top-10-Kritischer-Blick-auf-die-Charts-3953648.html?seite=all`

Williams, H. E. & Lane, D. (2004). *Web database applications with php & mysql, 2nd edition*. O'Reilly Media.

Wodehouse, C. (2017, September). *Securing apps from the ground up with the spring security framework.* Zugriff am 01. Okt. 2019 auf `https://www.upwork.com/hiring/development/spring-security-framework/`

Wolf, D., Henley, A. & Henley, A. J. (2017). *Java ee web application primer - building bullhorn: A messaging app with jsp, servlets, javascript, bootstrap and oracle* (1st ed. Aufl.). New York: Apress.

Ye, J. J. (2018). *Building applications with spring 5 and vue.js 2 - build a modern, full-stack web application using spring boot and vuex* (1. Aufl. Aufl.). Birmingham: Packt Publishing Ltd.