Analyse der Usability von „Lara“ – ein Online-Berichtsheft

Transferleistung 4

NORDAKADEMIE - Hochschule der Wirtschaft

Matrikelnummer	10001
Abgabedatum	07 Juni 2022
Kooperationsunternehmen	SinnerSchrader Deutschland GmbH
Anschrift	Völckersstr. 38 22765 Hamburg

Sperrvermerk

Die vorliegende Transferleistung basiert auf internen und vertraulichen Daten und Informationen der SinnerSchrader Deutschland GmbH. In diese Arbeit dürfen Dritte, mit Ausnahme der Gutachter und befugten Mitgliedern des Prüfungsausschusses, ohne ausdrückliche Zustimmung des Unternehmens und des Verfassers, keine Einsicht nehmen.

Die Vervielfältigung oder Veröffentlichung dieser Transferleistung ohne ausdrückliche Genehmigung der SinnerSchrader Deutschland GmbH ist, auch auszugsweise, nicht erlaubt.

Inhaltsverzeichnis

Tabellenverzeichnis

Abküzungsverzeichnis

CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
JSX	Javascript XML

1 Einleitung

In Deutschland gibt es, Stand 2020, um und bei 1.3 Millionen Auszubildende, welche ein Berichtsheft führen müssen, um ihren Fortschritt in der Ausbildung festzuhalten. Dieses musste bis Oktober 2017 analog abgegeben werden. Doch die Umsetzung dieses digitalen Prozesses dauert bis heute an. So kann man sein Berichtsheft in der Hamburger Handelskammer erst ab Sommer 2021 digital abgeben. Rechnet man also mit einer Ausbildungsdauer von zweieinhalb Jahren und dem Fakt, dass pro Woche eine Seite eines Blattes in Anspruch genommen wird, muss jeder Auszubildende um und bei 65 Blätter Papier bedrucken. Dazu gehören noch nicht die zuvor an Ausbilder zum korrigieren übergebene Berichte oder Fehldrucke.

1.1 Projektvorstellung

Um diesen Papierverbrauch zu reduzieren, haben sich die Auszubildenden von SinnerSchrader 2017 entschieden ein digitales Berichtsheft namens „Lara“ zu entwickeln. Dieses gibt den Auszubildenden und Ausbildern die Möglichkeit jegliche Prozesse im Zusammenhang mit dem Berichtsheft auf einer Online-Plattform abzuwickeln.
Zwar hat sich das Berichtsheft zu teilen in den letzen Jahren digitalisiert, doch sind die Möglichkeiten immer noch eingeschränkt und von Handelskammer zu Handelskammer unterschiedlich. Auch haben nicht alle Unternehmen die verschiedenen Lösungen der Handelskammern im Einsatz, sondern eigene Lösungen.
Diese Gründe führten zur Motivation „Lara“ Open Source verfügbar zu machen und so eine umfangreiche, einfache und anpassbare Lösung für Alle zu bieten.

Mit „Lara“ können Auszubildende tägliche Berichte verfassen, welche in Wochen zusammengefasst werden. Diese Wochen können direkt an Ausbilder übergeben werden. Ausbilder können diese übergebenen Wochen überprüfen und Kommentare hinterlassen, falls Änderungen vorgenommen werden müssen. Falls der Bericht keine Korrektur benötigt, wird dieser Archiviert. Aus diesem Archiv können die Auszubildenden ihre Berichte im PDF Format exportieren.
Auszubildende können außerdem die Status ihrer Berichte einsehen und Ausbilder haben eine Übersicht über ihre Auszubildenden.

1.2 Zielsetzung

Zur Verfassungszeit dieser Transferleistung ist „Lara“ im Prozess Open Source zu werden. Diese Transferleistung soll diesen Prozess unterstützen in dem die Usability untersucht wird und mögliche Verbesserungsmöglichkeiten aufgedeckt werden. So soll die Attraktivität von „Lara“

gesteigert werden. Außerdem soll ein möglicher schlechter Einfluss auf das Image von SinnerSchrader durch Veröffentlichung eines nicht optimierten Produktes verhindert werden.

1.3 Aufbau der Arbeit

Zur Analyse von „Lara“ werden die im fünften Semester erlernten Inhalte angewandt. Es wird die erste Phases des Vorgehensmodells „Scenario Based Design“, die Analysephase, durchgeführt. Hierbei wird im ersten Schritt eine Umfrage zur Kontextanalyse entworfen und durchgeführt. Die Ergebnisse dieser Umfrage werden in die folgende heuristische Analyse eingebunden. Anhand der Ergebnisse beider Methodiken wird final ein Usertest aufgebaut und durchgeführt. Die erhaltenen Ergebnisse werden zum Schluss so zusammengefasst, dass diese in den anschließenden Phasen des „Scenario Based Design’s“ außerhalb dieser Transferleistung verarbeitet werden können.

1.4 Unternehmensprofil SinnerSchrader

SinnerSchrader ist eine 1996 in Hamburg gegründete Digitalagentur. 1999 ging sie an die Börse und verfolgt mit dem Zusammenschluss der Accenture Digital Holdings GmbH 2017 das Ziel die größte Digitalagentur für die Region Deutschland, Österreich und Schweiz zu sein.
Die Leistungen fokussieren sich auf E-Commerce, Strategie und Kommunikation und umfassen dabei Gestaltung und Konzeption, Entwicklung und den Betrieb von digitalen Plattformen, sowie Kampagnen, Analytics, Mobile Apps, Service Design und Audience Management. An den insgesamt sechs Standorten Hamburg, Berlin, Frankfurt am Main, München, Prag und Hannover arbeiten etwa 600 Mitarbeiter, wovon knapp 200 Entwickler sind. [S2AG2020]

2 Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen ausgeführt. Dies ist für den anschließenden Vergleich notwendig.

2.1 Single Page Application

Single Page Applications sind Webanwendungen, welche nur aus einem Hypertext Markup Language (HTML)-Dokument bestehen. Dabei werden die Inhalte dynamisch nachgeladen. Ein Webserver sendet erst eine HTML-Datei ohne Inhalt. Darauf folgend wird eine JavaScript Datei versendet, welche den Programmcode beinhaltet. Dieser beinhaltet die Instruktionen um die Seite zu generieren, darzustellen und die benötigten Inhalte vom einem Server anzufragen.

[Bew18, S.309f]
Die Popularität von Single Page Applications ist in den vergangenen Jahren gewachsen. Damit einhergehend ist auch die Zahl an Hilfsmitteln zur Erstellung dieser gestiegen. Zu diesen gehören auch die beiden Open Source Projekte React und Vue. Open Source bedeutet, dass der Quellcode öffentlich einsehbar ist und Nutzer auch selbst Änderungswünsche vorschlagen und oder implementieren können. [AA18, S.2]
Diese beiden Projekte setzen daran an, die Entwicklung eines Frontends weitestgehend zu vereinfachen. Dabei werden vor allem drei Faktoren in Angriff genommen. [Ber17, S.157f]
Ein Faktor ist das Verwalten des globalen Verhaltens von HTML, Cascading Style Sheets (CSS) und JavaScript. Dabei müssen zum Beispiel Konflikte in der Benennung von Variablen und CSS-Klassen behandelt werden. Ein weiterer Faktor ist es das DRY-Prinzip zu unterstützen, indem wiederverwendbare Komponenten erstellt werden können. So wird Code-Duplizierung vermieden und auch die Wartbarkeit erhöht. [Ste19, S.62f]
Der dritte Faktor ist die effiziente und schnelle Aktualisierung des Document Object Model (DOM). Das DOM stellt den Inhalt der HTML-Datei als Baumstruktur in einem JavaScript-Objekt dar und durch Zugriff auf dieses können Änderungen am Inhalt getätigt werden. Manuelle Änderungen sind aufwändig und je effizienter die Änderung erfolgt, desto geringer ist die Rechenzeit um neue Inhalte anzuzeigen.

2.2 Framework versus Library

Die Begriffe „Framework“ und „Library“ werden häufig als Synonyme verwendet, wobei sich die Definitionen und auch die Möglichkeiten unterscheiden.
Eine Library muss von einem*er Entwickler*in explizit aufgerufen werden. Frameworks hingegen werden eingesetzt, wobei benutzerdefinierter Programmcode in vorgefertigte Strukturen eingesetzt werden muss. [Woz]

2.3 React

React ist eine Library und bietet, wie in Kapitel 2.1 bereits erläutert, die Möglichkeit wiederverwendbare Komponenten zu erstellen. In React bestehen diese aus einer Datei, welche den Programmcode und die Struktur in Form von Javascript XML (JSX) Syntax enthält. JSX ist hierbei vergleichbar mit der HTML Syntax. Neben den üblichen HTML-Elementen können auch eigene Elemente verwendet werden, wobei diese wiederum aus HTML- oder eigenen Elementen bestehen können. Ein weiterer Unterschied ist, dass JavaScrip Code direkt innerhalb der JSX Syntax verwendet werden kann. Zum Beispiel durch den Einsatz des Ternary Operators kann

einfach zwischen verschiedenen Zuständen unterschieden und die Darstellung des Inhaltes angepasst werden. [BP20, S.81ff]
React bietet keinen direkten Ansatz um die Verkapselung von CSS Klassen zu ermöglichen. Dies kann jedoch durch zusätzliche Module wie „styled-components“ erreicht werden. Hierbei wird für jede Komponente eine eindeutige CSS Klasse zugewiesen um mögliche Konflikte zu vermeiden. [Ber17, S.157f]
React benutzt außerdem einen Virtual DOM um eine effiziente und schnelle Aktualisierung des DOM durchzuführen. Der Virtual DOM ist eine Kopie des echten DOM und wird im Arbeitsspeicher festgehalten. Dieser wird mit dem echten DOM mithilfe eines Algorithmus möglichst effizient synchronisiert.
Der Algorithmus hat vor allem das Bestreben möglichst minimale Änderungen an dem DOM durchzuführen. So wird zum Beispiel nicht ein gesamtes Element ersetzt, wenn sich ein Attribut oder eine Klasse ändert, sondern nur die betroffene Eigenschaft. [BP20, S.59ff]

2.4 Vue

Vue ist ein Framework und bietet auch die Möglichkeit wiederverwendbare Komponenten zu erstellen. Ebenso wie in React bestehen die Komponenten aus einer Datei. Diese beinhaltet die Struktur innerhalb eines „template“-Tags, den Programmcode in einem „script“- und die CSS Definitionen in einem „style“-Tag. Zur Beschreibung der Struktur wird HTML eingesetzt, welches um einige spezielle Direktiven erweitert wurde. Es existiert zum Beispiel eine „v-if“-Direktive welche entscheidet, ob ein Element angezeigt werden soll oder nicht. Wie in React können auch eigene Komponenten in dem „template“-Tag verwendet werden.
Im Gegensatz zu React besitzt Vue selbst die Fähigkeit CSS Definitionen zu kapseln. Ähnlich wie bei „styled-components“ werden CSS Klassen mit eindeutigen Ids generiert.
Genauso wie React verwendet auch Vue einen Virtual DOM um die Aktualisierung des echten DOM's effizient zu gestalten. [Ste19, S.10f]

3 Vergleich

Im folgenden Kapitel werden React und Vue auf folgende Vergleichskriterien untersucht:

Performance

Es wird ein Beispielprojekt in jeweils React und Vue erstellt, welches das gleiche Frontend darstellt und mit GraphQL Abfragen an ein lokales Backend schickt. Es sollen dabei einige Metriken, wie der „First Contentful Paint“ oder die „Time to Interactive“, betrachtet werden.

Ease of Use

Hier soll festgestellt werden, welche der beiden Technologien einfacher zu hand haben ist. Dazu gehört auch die Lernzeit.

Community

Bei diesem Kriterium soll ein kurzer Blick auf die Community geworfen werden.

Potential

Bei dem letzen Kriterium sollen die natürlichen Möglichkeiten und Einschränkungen beider Technologien aufgewiesen werden, welche aus deren Programmierung hervorgehen.

Es muss jedoch vorab erwähnt werden, dass aus Gründen der Umfangsbeschränkungen der Transferleistung nur auf die Performance näher eigegangen werden kann. Die anderen Kriterien werden auch betrachtet, um ein vollständiges Vergleichsbild zu schaffen, werden jedoch nicht ausführlich hergeleitet.

3.1 Performance

Die Performance soll anhand von zwei Bespielapplikationen, eine für React und eine für Vue, verglichen werden. Dafür werden „Lighthouse“ und der Chrome-Profiler verwendet.
Zunächst sollen jedoch die beiden Anwendungen erstellt werden.

3.1.1 Aufbau der Bespielapplikationen

Beide Applikationen wurden mithilfe des von React beziehungsweise Vue bereitgestellten Command Line Interface Tool erstellt. Zusätzlich werden die nötigen Plugins / Module installiert, um GraphQL abfragen schicken zu können. Außerdem wird in der React-Anwendung das Modul „styled-componets“ eingesetz, um wie bei Vue, die Kapselung von CSS-Definitionen zu ermöglichen.
Darstellen tun beide Frontends das Selbe. Hierbei wird darauf geachtet, dass das „JSX“ von React und das „template“ von Vue möglichst gleich aufgebaut sind.
Da die Idee für diese Transferleistung beim Aufsetzen eines neuen internen Projektes entstanden ist, soll ein Teil der Funktionalität abgebildet werden. Das interne Projekt mit dem Namen „s2gether“ entwickelt eine Single Page Application um das Buchen von Tischen im Büro zu ermöglichen. Diese können auf einer Karte gesehen und ausgewählt werden. Die Beispielapplikationen sollen das Rendern der Tische nachstellen, indem mit GraphQL die nötigen Daten zunächst abgefragt und dann in einzelnen „div’s“ dargestellt werden. Außerdem existiert ein Knopf mit der Funktionalität neue Tisch-Daten anzufragen.

Die Daten stellen 58 einzelne Tische dar.

Um die schwankende Leistung des Internetzugangs nicht in der weiteren Analyse betrachten zu müssen, wird das zuständige Backend und ein Webserver lokal gestartet.

3.1.2 Performance Tests

Es werden zwei Tests durchgeführt. Einmal ein Initial Load Test mit Hilfe von Lighthouse und ein Scripting/Rendering Test. Beim Zweitem wird das Neuladen der Tische durch Klicken des Buttons mit dem Profiler von Chrome betrachtet. [Bas]

Initial Load Test

Die Ergebnisse von Lighthouse wurden jeweils zehn mal erfasst, um Ausreißer ausschließen zu können. Dabei ergeben sich folgende Ergebnisse:

	React	Vue
First Contentful Paint	0,6s	0,6s
Time to Interactive	0,6s	0,6s
Speed Index	0.8s	0,6s
Total Blocking Time	0s	0s
Largest Contentful Paint	0,6s	0,6s
Performance Score	100	100

Tabelle 2: Ergebnisse von Lighthouse

Scripting/Rendering Test

Auch die Ergebnisse des Scripting/Rendering Tests wurden zehn mal erfasst. Folgende Ergebnisse waren das Resultat.

	React	Vue
Scripting Time	40ms	75ms
Rendering Time	3ms	1ms
Total Time	43ms	76ms
Layout Shift	11ms	54ms

Tabelle 3: Messwerte mit dem Profiler

3.1.3 Auswertung der Ergebnisse

Im Folgenden werden jeweils die Bedeutungen der einzelnen Metriken erklärt [web] und parallel die Ergebnisse verglichen und analysiert. Zuerst wird der Initial Load Test behandelt.

Performance Score

Lighthouse errechnet den Performance Score aus sechs gemessenen Metriken. Dabei werden diese unterschiedlich gewichtet.

React und Vue erreichen hierbei die volle Punktzahl, was für eine hervorragende Performance steht.

First Contentful Paint

Der First Contentful Paint beschreibt die Zeit, bis das erste Element auf der Seite angezeigt wird.

Bis zu diesem Zustand benötigen React und Vue 0,6 Sekunden. Wobei das erste Element wohlmöglich ein kurzer Text ist, welcher eine Überschrift simulieren soll.

Time to Interactive

Die Time to Interactive beschriebt die Dauer, bis mit einer Seite interagiert werden kann. Sprich alle „Event-Handler“ sind registriert und reagieren innerhalb von 50 Millisekunden auf Interaktion.

Auch hier benötigen beide Technologien 0,6 Sekunden. Dabei ist dieser Wert identisch zum First Contentful Paint. Dies lässt sich damit erklären, dass das einzige interaktive Element ein Button ist, welcher direkt neben dem Text steht und auch direkt gerendert werden kann.

Speed Index

Der Speed Index misst die Zeit, bis die Seite vollständig angezeigt wird.

Hier zeigt sich der einzige Unterschied zwischen React und Vue im Initial Load Test, denn React braucht mit 0,8 Sekunden 0,2 länger als Vue. Da Vue die selbe Zeit wie beim First Contentful Paint benötigt lässt sich folgern, dass der First Contentful Paint erst auf die Daten vom Sever hört. Somit wurde zu diesem Zeitpunkt die Anfrage geschickt, eine Antwort erhalten und auch schon verarbeitet. Dies spiegelt sich auch im Networktab in den Entwicklertools wieder. React sollte in der Theorie genauso reagieren, weist jedoch in den meisten Fällen einen Overhead auf.

Total Blocking Time

Die Total Blocking Time summiert alle Funktionsaufrufe zwischen dem First Contentful Paint und der Time to Interactive stattfinden und länger als 50 Millisekunden brauchen.

Da die Total Blocking Time in beiden Fällen null ist und der First Contentful Paint und die Time to Interactive identisch sind muss untersucht werden, ob keine Funktionsaufrufe getätigt wurden oder diese weniger als 50 Millisekunden benötigt haben. Tatsächlich ist im Profiler zu

erkennen, dass Funktionsaufrufe in beiden Fällen getätigt wurden. Somit kann auf eine hohe Optimierung geschlossen werden.

Largest Contentful Paint
Der Largest Contentful Paint beschreibt die Dauer bis das größte Element dargestellt wurde. Da der Largest Contentful Paint mit beiden Technologien 0,6 Sekunden gedauert hat und somit jeweils identisch zum First Contentful Paint ist, kann die Aussage bestätigt werden, dass der First Contentful Paint erst auf die Daten vom Server reagiert hat.

Cumulative Layout Shift
Hier wird die Zeit gemessen, welche die Elemente brauchen um sich auf der Seite zu verschieben. Da sich jedoch keine Elemente in den Beispielanwendungen verschieben müssen, ist der Wert 0 und auch nicht interessant für diesen Vergleich.

Im Weiteren werden die Ergebnisse des Scripting/Rendering Tests betrachtet. Hierbei wurde die Interaktion mit dem Button genauer untersucht. Beim Klick werden neue Tisch-Daten vom Server abgefragt. Dabei wurden die Daten durch Buchungseinträge, wie in s2gether, erweiter.

Scripting Time
Die Scripting Time beschriebt die Zeit, in der JavaScript Code nach dem Klick auf den Knopf ausgeführt wird. Vue braucht mit 75 Millisekunden im Durchschnitt 35 Millisekunden länger als React. Interessant hierbei ist, dass der Layout Shift deutlich mehr Zeit braucht als React. Da dieser auch in der Scripting Time integriert ist, kann erschlossen werden, dass Vue's Algorithmus zum Aktualisieren des Virtual-DOM's etwas ineffizienter ist, wenn Daten hinzugefügt werden. Würde man den Layout Shift nicht mit betrachten, wäre Vue rund acht Millisekunden schneller als React.

Rendering Time
Wie der Name schon verrät, beträgt die Rendering Time die Zeit, um Änderungen wieder zu stellen und anzuzeigen.
Hier zeigt sich der Vorteil in der Verwendung des Virtual-DOM. React braucht nur drei und Vue nur eine Millisekunde um die Änderungen im Virtual-DOM auf den echten DOM zu übertragen.

Total Time
Die Total Time ist die Summe aus Scripting und Rendering Time. Durch den langen Layout

Shift in Vue liegt React hier vorne mit 43 Millisekunden. Vue braucht hingegen 76 Millisekunden. Würde man den Layout Shift nicht betrachten, wäre Vue mit rund zehn Millisekunden vorne.

3.2 Ease of Use

In verschiedenen Arbeiten wurde festgestellt (Beispielsweise in [Woh18], [Ata]), dass der Umgang mit Vue simpler ist als mit React. Dabei wurden Aspekte wie die Lernzeit und Einfachheit des Gebrauchs betrachtet, wobei Vue jeweils vorne lag.
Dabei sind genau zwei Gründe für die Überlegengenheit von Vue zum Vorschein gekommen:

JSX versus Template

JSX wird von den meisten Benutzern als schwerer empfunden als das sehr HTML ähnliche Template von Vue. Da beide grundlegende Kenntnisse von HTML voraussetzen, ist dieser Zustand auch nachvollziehbar, da Vue die HTML Syntax nur geringfügig erweitert.

Documentation

Auch hier sind sich die Meisten einig. Vue hat eine bessere und strukturiertere Dokumentation. Alle nötigen Plugins werden dabei einzeln beschrieben und werden von Grund auf mit steigender Komplexität ausgearbeitet.
Aus eigenen Erfahrungen muss hier jedoch hinzugefügt werden, dass noch nicht alle Dokumentationen von Vue vollständig sind. Zum Beispiel fehlen einige wichtige Punkte in der GraphQL Dokumentation.

3.3 Community

Sucht man in Google nach „react“ findet man über 328 Millionen Sucheinträge. Bei der Suche nach „vue“ sind dies 471 Millionen.
Ein entgegengesetztes Bild entsteht jedoch, wenn man nach Paketen bei npm sucht. Für React wurden 175.416 Pakete gefunden. Für Vue wurden „nur“ 54.985 aufgelistet.

3.4 Potential

Dadurch, dass React eine Library ist und Vue ein Framework, entstehen nicht nur Unterschiede in der Definition sondern auch in den Möglichkeiten der Technologien.
So ist React zum Beispiel weniger eingeschränkt. Es kann vereinfacht behauptet werden: „Solange der Code React erfolgreich aufrufen kann, ist die Implementation möglich.“.
Vue hingegen ist stärker eingegrenzt. Einfach gesagt: „Vue muss den Code verstehen können, damit dieser funktioniert.“.

4 Fazit

Zusammenfassend lässt sich sagen, dass die Implementierung von Vue nur sehr geringfügige Performance-Vorteile besitzt. Beide Technologien sind in diesem Gebiet so gut wie gleich auf, wobei sich nur der Speed Index unterscheidet. Dies kann darauf zurück geführt werden, dass die Größe des von React generierten Bundels, rund 50 kB mehr Umfang hat als das von Vue.
Es kann außerdem festgehalten werden, dass beide Technologien in diesem Gebiet große Fortschritte gemacht haben. Noch vor einem Jahr waren die Performance Ergebnisse deutlich schlechter und auch nicht so nah beieinander.
Jedoch müssen die hier aufgezeichneten Werte mit Vorsicht betrachtet werden. Nicht das volle Potential beider Technologien wurde getestet. Für eine bessere Representation müssten mehrere und auch größere Projekte nachgebaut werden. Für eine grobe Einstufung sollten die gemessenen Ergebnisse jedoch genügen.

Wann sollte jetzt aber React oder Vue in einem Projekt eingesetzt werden? Da die Performance keine großen Unterschiede aufweist, müssen zur Beantwortung dieser Frage die anderen Vergleichskriterien hinzugezogen werden.
Aus diesen kann erschlossen werden, dass Vue, als Framework, sich vor allem für Projekte ohne viele besondere individuelle Funktionalitäten eignet. Auch Einsteiger in der Webentwicklung sollten es mit Vue, durch die simplere Dokumentation, einfacher haben.
Im Gegensatz richtet sich React, als eine Library, eher an Projekte mit vielen komplexen und individualisierten Funktionalitäten. Zusätzlich ist die Dokumentation komplexer und durch Module von Drittanbietern verstreuter. Somit werden hier eher erfahrenere Entwickler angesprochen.

Abschließend kann für SinnerSchrader festgehalten werden, dass Vue mittlerweile seine Daseinsberechtigung verdient hat und öfter in Projekten zum Einsatz kommen könnte. Nicht alle Projekte brauchen die Vorteile einer Library.
Zukünftig könnte somit durch Beantwortung der Frage „Besteht ein Mehrwert durch den Einsatz von React?“ Zeit durch den einfacheren Entwicklungsprozess von Vue eingespart werden.

Literatur

[AA18] Daniel Graziotin Amantia Pano und Pekka Abrahamsson. *Factors and actors leading to the adoption of a JavaScript framework*. 2018.

[BP20] Alex Banks und Eve Porcello. *Learning React*. Bd. 2. O'Reilly Media, Inc., 2020.

[Ber17] Michele Bertoli. *React Design Patterns and Best Practices*. Bd. 1. Packt, 2017.

[Bew18] Jörg Bewersdorff. *Objektorientierte Programmierung mit JavaScript*. Bd. 2. Springer Verlag, 2018.

[Ste19] Ralph Steyer. *Webanwendungen erstellen mit Vue.js*. Bd. 1. Springer Vieweg, 2019.

[Woh18] Eric Wohlgethan. *Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js*. HAW, 2018.

Internetquellen

[Ata] André Atalaia. *VUE.JS VS REACT: WE BUILT AN APP ON BOTH FRAMEWORKS*. Website. `https://www.imaginarycloud.com/blog/vue-js-vs-react-an-app-on-both-frameworks/`, abgerufen am 25.10.2021.

[Bas] Kayce Basques. *Analyze runtime performance*. Website. `https://developer.chrome.com/docs/devtools/evaluate-performance/`, abgerufen am 27.10.2021.

[web] web.dev. *Performance audits*. Website. `https://web.dev/lighthouse-performance/`, abgerufen am 27.10.2021.

[Woz] Brandon Wozniewicz. *The Difference Between a Framework and a Library*. Website. `https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/`, abgerufen am 23.10.2021.

Firmeninterne Quellen

[S2AG2020] SinnerSchrader AG. *Bericht zum Jahresabschluss 2019/2020*. Website. `https://sinnerschrader.ag/media/filer_public/2b/9f/2b9f3774-108d-4819-803e-67bde328ea63/sinnerschrader_bericht_jahresabschluss_2019_2020_pw_de.pdf`, abgerufen am 20.10.2021.