Bachelorarbeit

Bessere Nachvollziehbarkeit clientseitiger JavaScript-basierter Webapplikationen

Ein Ansatz zur Verbesserung der Nachvollziehbarkeit von Nutzerinteraktion und Anwendungsverhalten

An der Fachhochschule Dortmund
im Fachbereich Informatik
Studiengang Software- und Systemtechnik, Vertiefung Softwaretechnik
erstellte Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Science

von
Marvin Kienitz
geb. am 26.04.1996
Matr.-Nr. 7097533

Betreuer:
Prof. Dr. Sven Jörges
Dipl. Inf. Stephan Müller

Dortmund, 23. November 2020

Kurzfassung

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Open Knowledge GmbH ist als branchenneutraler Softwaredienstleister in einer Vielzahl von Branchen wie Automotive, Logistik, Telekommunikation und Versicherungs- und Finanzwirtschaft aktiv. Zu den zahlreichen Kunden der Open Knowledge GmbH gehört auch ein führender deutscher Direktversicherer.

Ein Direktversicherer bietet Versicherungsprodukte seinen Kunden ausschließlich im Direktvertrieb, d. h. vor allem über das Internet und zusätzlich auch über Telefon, Fax oder B rief an. Im Unterschied zum klassischen Versicherer verfügt ein Direktversicherer jedoch über keinen Außendienst oder Geschäftsstellen, bei denen Kunden eine persönliche Beratung bekommen können. Da das Internet der primäre Vertriebskanal ist, gehört heute ein umfassender Online-Auftritt zum Standard. Dieser besteht typischerweise aus einem Kundenportal mit der Möglichkeit Angebote für Versicherungsprodukte berechnen und abschließen zu können, sowie persönliche Daten und Verträge einzusehen.

Während in der Vergangenheit Online-Auftritte i. d. R. als Webapplikation mit serverseitigen Rendering realisiert wurden, sind heutzutage Javascript-basierte Webapplikation mit clientseitigem Rendering die Norm. Bei einer solchen Webapplikation befindet sich die gesamte Logik mit Ausnahme der Berechnung des Angebots und der Verarbeitung der Antragsdaten im Browser des Nutzers.

Im produktiven Einsatz kommt es auch bei gut getesteten Webapplikationen hin und wieder vor, dass es zu unvorhergesehenen Fehlern in der Berechnung oder Verarbeitung kommen kann. Liegt die Ursache für den Fehler im Browser, z. B. aufgrund einer ungültigen Wertkombination, ist dies eine Herausforderung. Während bei Server-Anwendungen Fehlermeldungen in den Log-Dateien einzusehen sind, gibt es für den Betreiber der Anwendung i. d. R. keine Möglichkeit die notwendigen Informationen über den Nutzer und seine Umgebung abzurufen. Noch wichtiger ist, dass er mitbekommt, wenn ein Nutzer ein Problem bei der Bedienung der Anwendung hat. Ohne eine aktive Benachrichtigung durch den Nutzer, sowie detaillierte Informationen, ist es dem Betreiber nicht möglich, Kenntnis über das Problem zu erlangen, geschweige denn dieses nachzustellen.

Dies stellt ein Kernproblem von Webapplikationen dar [Fil20]. Im Rahmen der Arbeit soll daher ein Proof-of-Concept konzipiert und umgesetzt werden, welcher dieses Kernproblem am Beispiel einer Demoanwendung löst.

1.2 Zielsetzung

Das grundlegende Ziel dieser Arbeit soll es sein, den Betreibern einer JavaScript-basierten Webapplikation die Möglichkeit zu geben das Verhalten ihrer Applikation und die Interaktionen von Nutzern. Diese Nachvollziehbarkeit soll insbesondere bei Fehlerfällen u. Ä. gewährleistet sein, aber auch in sonstigen Fällen soll eine Nachvollziehbarkeit möglich sein. Eine vollständige Überwachung der Applikation und des Nutzers (wie bpsw. bei Werbe-Tracking) sind jedoch nicht vorgesehen. Daraus ergibt sich die Forschungsfrage:

> Wie sieht ein Ansatz aus um bei clientseitigen JavaScript-basierten Webapplikation den Betreibern eine Nachvollziehbarkeit zu gewährleisten?

Vom Leser wird eine Grundkenntnis der Informatik in Theorie oder Praxis erwartet, aber es sollen keine detaillierten Erfahrungen in der Webentwicklung vom Leser erwartet werden. Daher sind das Projektumfeld und seine besonderen Eigenschaften zu erläutern.

Die anzustrebende Lösung soll ein Proof-of-Concept sein, welches anhand einer bestehenden Demo-Anwendung erstellt werden soll. Die Demo-Anwendung soll repräsentativ eine abgespeckte JavaScript-basierte Webapplikation darstellen, bei der die zuvor benannten Hürden zur Nachvollziehbarkeit bestehen.

Vor der eigentlichen Lösungserstellung soll jedoch die theoretische Seite beleuchtet werden, indem die Nachvollziehbarkeit sowie Methoden und Praktiken zur Erreichung dieser beschrieben werden. Es gilt aktuelle Literatur und den Stand der Technik zu erörtern, in Bezug auf die Forschungsfrage. Beim Stand der Technik sollen zwei bis drei Technologien näher betrachtet werden und beschrieben werden.

Weiterhin gilt es zu beleuchten, wie die Auswirkungen für die Nutzer der Webapplikation sind. Wurde die Leistung der Webapplikation beeinträchtigt (erhöhte Ladezeit, erhöhte Datenlast)? Werden mehr Daten von ihm erhoben und zu welchem Zweck?

Am Ende der Ausarbeitung soll überprüft werden, ob und wie die Forschungsfrage beantwortet wurde. Auch die Übertragbarkeit der erstellten Lösung (PoC) und Ergebnisse gilt es hierbei näher zu betrachten.

1.2.1 Abgrenzung

Die Demo-Anwendung wird als Single-Page-Application (SPA) realisiert, denn hier bewegt sich das Projektumfeld von der Open Knowledge GmbH. Bei der Betrachtung der Datenerhebung und -verarbeitung ist eine volle Konformität mit der DSGVO nicht zu prüfen.

Bei der Erörterung zum Stand der Technik sollen detaillierte Produktvorstellungen nicht das Ziel sein, auch ist keine umfassende Gegenüberstellung anzustreben.

1.3 Vorgehensweise

Zur Vorbereitung eines Proof-of-Concepts wird zunächst die Ausgangssituation geschildert. Speziell wird auf die Herausforderungen der Umgebung "Browser" eingegangen, besonders in Hinblick auf die Verständnisgewinnung zu Interaktionen eines Nutzers und des Verhaltens der Applikation. Des Weiteren wird die Nachvollziehbarkeit als solche formal beschrieben und was sie im Projektumfeld genau bedeutet.

Darauf aufbauend werden allgemeine Methoden vorgestellt, mit der die Betreiber und Entwickler eine bessere Nachvollziehbarkeit erreichen können. Dabei werden die Besonderheiten der Umgebung beachtet und es wird erläutert, wie diese Methoden in der Umgebung zum Einsatz kommen können. Hiernach sind Ansätze aus der Literatur und Fachpraxis zu erörtern, welche eine praktische Realisierung der zuvor vorgestellten Methoden darstellen.

Auf Basis des detaillierten Verständnisses der Problemstellung und der Methoden wird nun ein Proof-of-Concept erstellt. Ziel soll dabei sein, die Nachvollziehbarkeit einer Webapplikation zu verbessern. Der Proof-of-Concept erfolgt auf Basis einer Demo-Anwendung, die im Rahmen dieser Arbeit erstellt wird.

Ist ein Proof-of-Concept nun erstellt, wird analysiert, welchen Einfluss es auf die Nachvollziehbarkeit hat und ob die gewünschten Ziele erreicht wurden (vgl. Zielsetzung).

1.4 Open Knowledge GmbH

Die Bachelorarbeit wird im Rahmen einer Werkstudententätigkeit innerhalb der Open Knowledge GmbH erstellt. Der Standortleiter des Standortes Essen, Dipl. Inf. Stephan Müller, übernimmt die Zweitbetreuung.

Die Open Knowledge GmbH ist ein branchenneutrales mittelständisches Dienstleistungsunternehmen mit dem Ziel bei der Analyse, Planung und Durchführung von Softwareprojekten zu unterstützen. Das Unternehmen wurde im Jahr 2000 in Oldenburg, dem Hauptsitz des Unternehmens, gegründet und beschäftigt heute 74 Mitarbeiter. Mitte 2017 wurde in Essen der zweite Standort eröffnet, an dem 13 Mitarbeiter angestellt sind.

Die Mitarbeiter von Open Knowledge übernehmen in Kundenprojekten Aufgaben bei der Analyse über die Projektziele und der aktuellen Ausgangssituationen, der Konzeption der geplanten Software, sowie der anschließenden Implementierung. Die erstellten Softwarelösungen stellen Individuallösungen dar und werden den Bedürfnissen der einzelnen Kunden entsprechend konzipiert und implementiert. Technisch liegt die Spezialisierung bei der Mobile- und bei der Java Enterprise Entwicklung, bei der stets moderne Technologien und Konzepte verwendet werden. Die Geschäftsführer als auch diverse Mitarbeiter der Open Knowledge GmbH sind als Redner auf Fachmessen wie der Javaland oder als Autoren in Fachzeitschriften wie dem Java Magazin vertreten.

2 Ausgangssituation

2.1 Browserumgebung

Web Browser haben sich seit der Veröffentlichung von Mosaic, einer der ersten populären Browser, im Jahr 1993 stark weiterentwickelt. Das Abrufen und Anzeigen von statischen HTML-Dokumenten wurde mit Hilfe von JavaScript um interaktive und später dynamische Inhalte erweitert. Heutzutage können in Browsern komplexe Webapplikationen realisiert werden, welche zudem unabhängig von einem speziellen Browser entwickelt werden können. Durch diese Entwicklung und die breiten Anwendungsfälle, besitzt die Umgebung "Browser" besondere Eigenschaften, welche nachfolgend beschrieben werden.

2.1.1 Browserprodukte

Die Vielfalt an Browsern bereitet Webentwicklern immer wieder eine Herausforderung, dass ein von ihnen bereitgestelltes Produkt für die Nutzer einwandfrei funktioniert, unabhängig ihrer Browserpräferenz. Die Häufigkeit solcher Probleme, auch Cross-Browser-Incompatibilities (XBI) genannt, hat jedoch abgenommen, unter anderem erklärbar durch den Trend von offenen Web-Standards [W3C20].

Generell lässt sich feststellen, dass auch in der Literatur die Veröffentlichungen in Bezug auf (In-)Kompatibilität von Browsern abnimmt, dies ist in Abbildung 2.1 zu betrachten. Dies spricht dafür, dass das Problem von XBIs nicht mehr so präsent ist, wie zuvor.

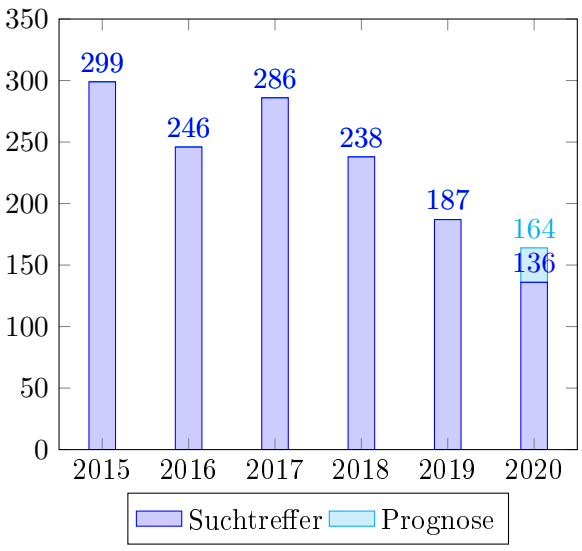

Abb. 2.1: Studien zur Browserkompatibilität, eigene Darstellung (vgl. 7.1)

Im Jahr 2020 gab es weitere Entwicklungen, die die Kompatibilität zwischen Browsern erhöhte. Microsoft ist beim Folgeprodukt zum Internet

Explorer, den Microsoft Edge Browser, von einer proprietären Browserengine zu Chromium gewechselt [Mic20b] und enthält somit den selben Kern wie Chrome und Opera. Zum Ende 2020 wird zudem der Support für den Internet Explorer 11 eingestellt [Mic20a]. Im September 2020 meldete Statista eine Marktverteilung von 69,71% Chrome, 8,73% Safari, 8,15% Firefox, 5,54% Edge, 2,51% Internet Explorer und 2,3% Opera [Sta20].

2.1.2 JavaScript

Als JavaScript 1997 veröffentlicht und in den NetScape Navigator integriert wurde, gab es die berechtigen Bedenken, dass das Öffnen einer Webseite dem Betreiber erlaubt Code auf dem System eines Nutzers auszuführen. Damit dies nicht eintritt, wurde der JavaScript Ausführungskontext in eine virtuelle Umgebung integriert, einer Sandbox. [Pow06]

Die JavaScript-Sandbox bei Browsern schränkt ein, dass unter anderem kein Zugriff auf das Dateisystem erfolgen kann. Auch Zugriff auf native Bibliotheken oder Ausführung von nativem Code ist nicht möglich [OKSK15]. Browser bieten dafür aber einige Schnittstellen an, die es erlauben z. B. Daten beim Client zu speichern oder auch Videos abzuspielen.

Microsoft nahm 1999 im Internet Explorer 5.0 eine neue Methode in ihre JavaScript-Umgebung auf: Ajax (Asynchronous JavaScript and XML). Ajax erlaubt die Datenabfrage von Webservern mittels JavaScript. Hierdurch wird ermöglicht, dass Inhalte auf Webseiten dynamisch abgefragt und dargestellt werden können, zuvor war hierfür ein erneuter Seitenaufruf notwendig. Das Konzept wurde kurz darauf von allen damals gängigen Browser übernommen. Erst jedoch mit der Standardisierung 2006 durch das W3C [W3C06] fand die Methode Anklang und Einsatz bei Entwicklern und ist seitdem der Grundstein für unser dynamisches und interaktives Web [Hop06].

Webapplikationen wurden nun immer beliebter, aber Entwickler klagten darüber, dass Browser die Abfragen von JavaScript nur auf dem bereitstellenden Webserver, also "same-origin", erlauben[Ran09]. Im selben Jahr der Standardisierung von Ajax, wurde ein erster Entwurf zur Ermöglichung und Absicherung von Abrufen domänenfremder Ressourcen eingereicht [OPBW06], das sogenannte Cross-Origin Resource Sharing.

Über die Jahre wurde der JavaScript Standard immer umfangreicher und dies erlaubte Entwicklern mächtige Werkzeuge sowie Frameworks zu entwickeln, um u. A. Webapplikationen bereitzustellen. Webapplikationen konnten nun dazu verwendet werden, um einen großen Teil der Funktionalitäten eines Produktes abzubilden. Diese "clientbasierten" Anwendungen werden im nächsten Abschnitt näher beleuchtet.

2.2 Clientbasierte Webapplikationen

2.2.1 JavaScript-basierte Webapplikationen

Eine JavaScript-basierte Webapplikation, ist eine Webapplikation, in der die Hauptfunktionalitäten über JavaScript realisiert werden. Dies umfasst unter anderem Interaktivität und dynamische Inhaltsdarstellung. Hierbei werden meist nur Grundgerüste in HTML und gegebenenfalls auch CSS bereitgestellt, und die eigentlichen Inhalte werden dynamisch mit JavaScript erstellt. Die Inhalte werden überwiegend über zusätzliche Schnittstellen der Webapplikation bereitgestellt.

2.2.2 Single-Page-Applications

Single-Page-Applications (SPAs) sind eine Submenge der JavaScript-basierten Webapplikationen und gehen bei der dynamischen Inhaltsdarstellung einen Schritt weiter. Die gesamte Anwendung wird über ein einziges HTML-Dokument und die darin referenzierten Inhalte erzeugt. Auf Basis dessen wird oftmals viel der Logik im Clientteil angesiedelt, welches die Anwendung in einen Rich- bzw. Fat-Client verwandelt.

Für das Bereitstellen einer solchen Applikation, ist unter anderem nur ein simpler Webserver notwendig und ein oder mehrere Dienste, von dem aus die SPA ihre Inhalte abrufen kann. Populäre Frameworks, um SPAs zu Erstellen, sind beispielsweise Angular [Goo20], React [Fac20] oder Vue.js [YM20].

Neben der Eigenschaft eine Alternative zu anderen Webapplikationen zu sein, bieten SPAs zudem den Stakeholdern die Möglichkeit diese als Offline-Version zur Verfügung zu stellen. Grund dafür ist, dass eine SPA bereits jedwede Logik enthält, sind zusätzlich keine externen Daten notwendig, so kann eine SPA simpel als Offline-Version bereitgestellt werden. Weiterhin steigern SPAs die User Experience (UX), indem sie unter anderem schneller agieren, da keine kompletten Seitenabrufe notwendig sind [AMWR20].

Durch diesen brachial anderen Ansatz, gibt es aber auch negative Eigenschaften. Unter anderem werden native Browserfunktionen umgangen, wie die automatisch befüllte Browserhistorie oder auch Aktionen zu Verlinkungen, wie Scrollrad-Klicks oder Lesezeichen, die mit “virtuelle“ Links und Buttons nicht möglich sind. Dies ist zu Teilen auch bei JavaScript-basierten Webapplikationen der Fall. Um diese Funktionalitäten wiederherzustellen sind für die zuvor genannten Frameworks Angular, React und auch Vue.js zusätzliche Bibliotheken notwendig.

Nichtsdestotrotz kann ein jahrelanger Trend von der Adaption von Single-Page-Applications erkannt werden und eine große Auswahl an erprobten Technologien stehen heutzutage zur Verfügung [GB20].

2.3 Nachvollziehbarkeit

Nachvollziehbarkeit bedeutet allgemein, dass über ein resultierendes Verhalten eines Systems auch interne Zustände nachvollzogen werden können. Dies ist keine neue Idee, sondern fand bereits 1960 im Gebiet der Kontrolltheorie starke Bedeutung [Ká60]. Nach Freedman [Fre91] und Scrocca *et al*[STM+20] lässt sich diese Definition auch auf Softwaresysteme übertragen.

In dieser Arbeit wird sich mit der Nachvollziehbarkeit in speziellen Situation befasst, nämlich wenn die Stakeholder das Verhalten einer Webapplikation und die Interaktionen eines Nutzers verstehen möchten.

2.3.1 Nutzen

Tritt beispielsweise ein Softwarefehler (Bug) bei einem Nutzer auf, aber die Stakeholder erhalten nicht ausreichende Informationen, so kann der Bug ignoriert werden oder gering priorisiert und in Vergessenheit geraten. Dies geschah im Jahr 2013, als Khalil Shreateh eine Sicherheitslücke bei Facebook fand und diesen bei Facebooks Bug-Bounty-Projekt Whitehat meldete [SD13]. Sein Fehlerreport wurde aufgrund mangelnder Informationen abgelehnt:

> Unfortunately your report [...] did not have enough technical information for us to take action on it. We cannot respond to reports which do not contain enough detail to allow us to reproduce an issue.

Durch den Bug konnte Shreateh auf die private Profilseite von Nutzern schreiben, ohne dass er mit ihnen vernetzt war. Um Aufmerksamkeit auf das Sicherheitsproblem zu erregen, hinterließ er eine Nachricht auf Facebooks Gründer und CEO Mark Zuckerbergs Profilseite. Erst hiernach nahm sich Facebooks Team dem Problem an.

2.3.2 Nachvollziehbarkeit bei SPAs

Wie zuvor in Abschnitt 2.2 "Clientbasierte Webapplikationen" geschildert, gibt es bei Webapplikationen und insbesondere Singe-Page-Applications besondere Barrikaden, die es den Stakeholdern erschwert das Verhalten einer Applikation und die Interaktionen eines Nutzers nachzuvollziehen.

Bei SPAs ist die Hauptursache, dass die eigentliche Applikation nur beim Client läuft und nur gelegentlich mit einem Backend kommuniziert.

2.3.3 Browserbedingte Hürden

2.3.3.1 Cross-Origin Resource Sharing (CORS)

Wie in der Geschichte zu JavaScript beschrieben, stellt CORS eine natürliche Entwicklung zur Erweiterung des Funktionsumfangs von JavaScript dar. Um nicht auf einen einzelnen Webserver beschränkt zu sein, wurde mit CORS diese Einschränkung eingegrenzt. Wichtiger Bestandteil von CORS ist aber eher die Absicherung vor Missbrauch, weswegen die Einschränkungen überhaupt existierte. Das Konzept von CORS stellt sicher, dass aus einer JavaScript-Umgebung heraus keine Ressourcen von Webservern angefragt werden, außer diese Webserver stimmen der Anfrage zu [MM20c].

In Abbildung 2.2 kann betrachtet werden, wie eine "cross-origin" Ajax-Anfrage nach dem Konzept von CORS gehandhabt wird. Hierbei ist zu sehen, dass wenn der Aufruf nicht standardmäßig[1]ist, der Browser eine zusätzliche OPTIONS Anfrage an den jeweiligen Webserver sendet - dies stellt einen sogenannten "Preflighted Request" dar. Wenn der Webserver in seiner Antwort auf die OPTIONS Anfrage nun bestätigt, dass die Methode, die Header und der "Content-Type" so erlaubt sind, wird auch die eigentliche Ajax-Anfrage ausgeführt. Ansonsten schlägt die Anfrage fehl und im JavaScript-Kontext ist lediglich der Fehlschlag zu sehen, ohne einen Hinweis auf die Diskrepanz bzgl. CORS.

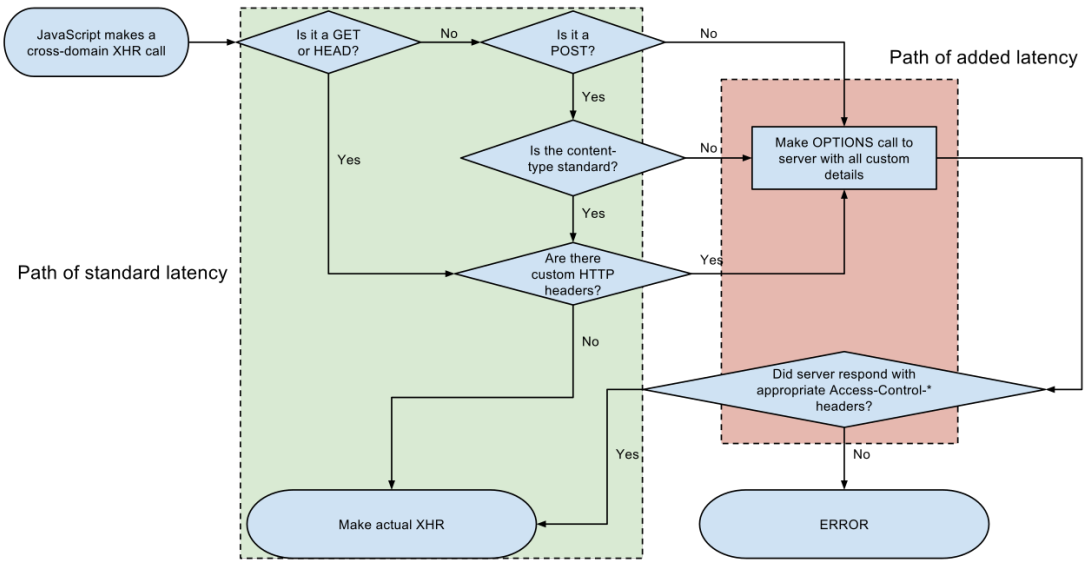

Abb. 2.2: Flowchart über den Ablauf von Ajax-Anfragen mit CORS [Blu15]

[1]Standardmäßig ist eine Anfrage, wenn 1. die Methode GET, HEAD oder POST entspricht; 2. keine eigene Headern enthält; und 3. der "Content-Type" von POST-Anfragen einem der folgenden Werte entspricht: "application/x-www-form-urlencoded", "multipart/form-data" oder "text/plain" [MM20c].

2.3.3.2 Content-Security-Policy

Eine Content-Security-Policy (CSP) definiert, welche Funktionalitäten einer Webapplikation aus geladen zur Verfügung stehen. Dies dient unter anderem dem Schutz vor Cross-Site-Scripting, indem eine Webapplikation beschränken kann, welche Funktionalitäten in JavaScript verfügbar sind und von wo aus JavaScript und CSS Skripte geladen werden dürfen [MM20b]. Weiterhin kann bei einem Versuch der Webapplikation die Regeln zu umgehen Bericht darüber erstattet werden.

2.3.4 Logdaten

Ähnlich wie bei anderen Umgebungen gibt es eine standardisierte Log- bzw. Konsolenausgabe für die JavaScript Umgebung [MM20a]. Diese Ausgabe ist aber für den Standard-Benutzer eher unbekannt und der Zugriff darauf sowie die Funktionen dessen können je nach Browser variieren. Deswegen kann in den meisten Fällen nicht darauf gehofft werden, dass die Nutzer dieses Log bereitstellen. Zusätzlich ist es durch die zuvor beschrieben Härtungsmaßnahmen von Browsern nicht möglich das Log in eine Datei zu schreiben.

Eine Automatisierung der Logdatenerhebung ist zudem auch nicht trivial, denn die Daten, welche in die Ausgabe geschrieben werden, sind nicht aus der JavaScript Umgebung aus lesbar. Alternativ können die Daten selber erhoben oder abgefangen werden. Hierbei besteht aber weiterhin die Hürde, wie die Daten an die Stakeholder gelangen.

2.3.5 Fernzugriff

Ein weiterer Punkt, der die Umgebung “Browser“ von anderen unterscheidet, ist dass die Stakeholder sich normalerweise nicht auf die Systeme der Nutzer schalten können. Bei Expertenanwendungen ginge dies vielleicht, aber wenn eine Webapplikation für den offenen Markt geschaffen ist, sind die Nutzer zahlreich und unbekannt.

Weiterhin gibt es standardmäßig keine Funktionalität wie z. B. das Remote Application Debugging [Ora20], welches Java unterstützt.

3 Methoden und Praktiken

In diesem Kapitel soll beschrieben werden, wie eine Nachvollziehbarkeit in Webapplikationen erreicht werden kann. Spezielle Methoden und Praktiken sollen vorgestellt und beleuchtet werden.

3.1 Methoden

3.1.1 Logging

Folgende Fragen sollen zur Methode beantwortet werden

1. *Gibt es Besonderheiten zu Logging in anderen Projekten (Backend vs. Frontend)?*
2. *Wie können Logs an einen auswertenden Stakeholder gelangen??*
3. *Welches Verhalten kann hiermit aufgedeckt/nachvollziehbar gemacht werden?*

3.1.2 Metriken

Folgende Fragen sollen zur Methode beantwortet werden

1. *Welche Metriken können definiert?*
2. *Wie können Metriken definiert werden?*
3. *Welches Verhalten kann hiermit aufgedeckt/nachvollziehbar gemacht werden?*

3.1.3 Tracing

Folgende Fragen sollen zur Methode beantwortet werden

1. *Welche Nutzerinteraktionen sind zu tracen?*
2. *Welches Verhalten kann hiermit aufgedeckt/nachvollziehbar gemacht werden?*

3.1.4 Fehlerberichte

Folgende Fragen sollen zur Methode beantwortet werden

1. *Was genau sind Fehlerberichte (=Bug-Reports)*
2. *Welches Verhalten kann hiermit aufgedeckt/nachvollziehbar gemacht werden?*

3.2 Werkzeuge und Technologien

In der Fachpraxis haben sich einige Technologien über die Jahre entwickelt und etabliert, die eine verbesserte Nachvollziehbarkeit als Ziel haben. Es lassen sich zudem verschiedene Funktionskategorien festlegen, auf die sich die jeweiligen Technologien konzentrieren. Die einzelnen Technologien lassen sich jedoch nicht strikt kategorisieren und weisen unterschiedliche Funktionsumfänge für dieselben Kategorien auf. Deshalb werden die Kategorien folgend beschrieben, aber bei der Vorstellung der Technologien erfolgt keine Kategorisierung oder direkte Gegenüberstellung.

3.2.1 Kategorien

3.2.1.1 System Monitoring

System Monitoring beschäftigt sich mit der Überwachung der notwendigen Systeme und Dienste in Bezug auf Hardware- und Softwareressourcen. Es handelt sich hierbei um ein projektunabhängiges Monitoring, welches sicherstellen soll, dass die Infrastruktur funktionstüchtig bleibt.

3.2.1.2 Log Management

Log Management umfasst die Erfassung, Speicherung, Verarbeitung und Analyse von Logdaten von Anwendungen. Weiterhin bieten Werkzeuge hierbei oftmals Such- und Meldefunktionen.

3.2.1.3 Application Performance Monitoring (APM)

Beim Application Performance Monitoring werden Daten innerhalb von Applikationen gesammelt, die Rückschlüsse auf die Perfomanz von bspw. Transaktionen geben sollen [ABC+16]. Mit diesen Daten können dann Regressionen der Performanz, in Aspekten wie Zeitaufwand oder Ressourcennutzung, festgestellt werden.

3.2.1.4 Real User Monitoring (RUM)

Real User Monitoring beschäftigt sich mit dem Mitschneiden von allen Nutzerinteraktionen mit bspw. einer Webapplikation. Hiermit lässt sich nachvollziehen, wie ein Nutzer die Anwendung verwednet. RUM kann dazu verwendet werden um Herauszufinden, wie ein Nutzer zu einem Zustand gelangt ist. Aber es können auch ineffiziente Klickpfade hierdurch festgestellt werden und darauf basierend UX Verbesserungen vorgenommen werden.

3.2.1.5 Synthetic Monitoring

Beim Synthetic Monitoring werden Endnutzerszenarien simuliert, um zu prüfen und sicherzustellen, dass diese Szenarien wie gewünscht ablaufen. Hierbei kann auf Aspekte wie Funktionalität, Verfügbarkeit und auch verstrichene Zeit kontrolliert werden.

3.2.1.6 Tracing

Tracing beschäftigt sich mit dem Aufzeichnen von Kommunikationsflüssen. Hierbei können einerseits die Kommunikationsflüsse innerhalb einer Applikation oder innerhalb eines Systems erfasst werden, aber auch andererseits die Kommunikationsflüsse bei verteilten Systemen erfasst werden, um diese meist komplexen Zusammenhänge zu veranschaulichen. Ein herstellerunabhängiger Standard, der sich aus diesem Gebiet entwickelt hat, ist OpenTracing [Ope20f].

3.2.1.7 Error/Crash Monitoring

Das Error Monitoring konzentriert sich auf das Erfassen und Melden von Fehlern. Es werden oftmals neben dem eigentlichen Fehler auch Aspekte vom RUM und Logging gemeldet, um mehr Kontextinformationen zu liefern.

3.2.1.8 Session Replay

Session Replay bedeuted, dass eine Sitzung eines Nutzers nachgestellt wird, so als ob sie gerade passiert. Hierbei können einzelne Aspekte der Anwendung nachgestellt werden, bspw. der Kommunikationsablauf, bei dem die tatsächliche zeitliche Abfolge von Kommunikationen nachvollzogen werden können. Desto mehr Aspekte nachgestellt werden, desto realitätsnaher ist die Simulation und entsprechend hilfreich ist sie beim Nachvollziehen.

3.2.2 Fachpraxis

3.2.2.1 OpenTelemetry

OpenTelemetry (OTel) [Ope20b] ist ein sich derzeit entwickelnder herstellerunabhängiger Standard, um Tracing-, Metrik- und Logdaten[1]zu erfassen, verarbeiten, analysieren und zu visualisieren. Der Standard fasst die beiden Standards OpenTracing und OpenCensus [Ope20a] zusammen und hat sich als Ziel gesetzt diese zu erweitern. Hinter dem Standard stehen u. A. die Cloud Native Computing Foundation (CNCF), Google, Microsoft, und führende Hersteller in Tracing und Monitoring-Lösungen. Ein erster Release ist für Ende 2020/Anfang 2021 geplant. Ziel ist es, dass Entwickler Tools und Werkzeuge benutzen können, ohne jedesmal eine hochspezifische Anbindung schreiben und konfigurieren zu müssen. Stattdessen definiert der Standard Komponenten, die spezielle Aufgabengebiete haben und mit einer allgemeinen API angesprochen werden können. Die technische Infrastruktur einer Lösung basierend auf OTel kann in Abbildung 3.1 betrachtet werden. Im groben definiert OTel folgende Komponenten: API, SDK, Exporter, Collector, Backend (vgl. Abbildung 3.2).

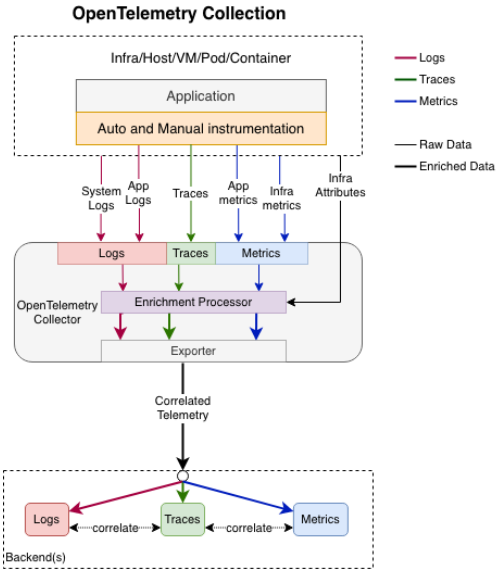

Abb. 3.1: Schaubild einer Lösung auf Basis von OTel [Ope20c]

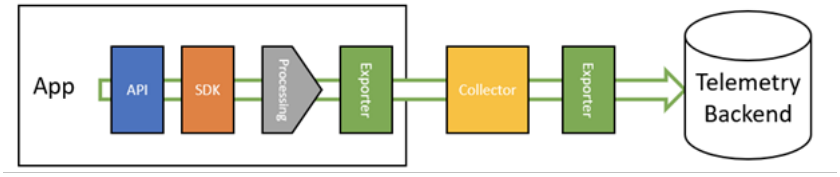

Abb. 3.2: OTel Komponenten [Dyn20c]

[1]Logging wird derzeit noch nicht unterstützt, es wird jedoch daran gearbeitet [Ope20e]

3.2.2.2 New Relic

New Relic [New20b] ist ein SaaS der gleichnamigen Firma, welcher Betreiber von Softwareprojekten dabei unterstützt das Verhalten ihrer Anwendungen zu überwachen. Der Dienst konzentriert sich auf System Monitoring, APM und RUM und erfasst die notwendigen Daten mit proprietären Lösungen. Neben den Kernfunktionalitäten unterstützt New Relic auch Log Management, Synthetic Monitoring, Tracing und Error Monitoring.

Bei New Relic wurde die kostenlose Version aufgesetzt und evaluiert. Der New Relic Agent, welcher die Daten beim Client sammelt, wird über ein Skript eingebunden und sendet in regelmäßigen Abständen Daten an New Relic. Über die Oberfläche von New Relic können dann allgemeine Charakteristika der Clients betrachtet werden, wie Ladezeiten, Browserhersteller, Ajax-Antwortzeiten. Spezielle Eigenschaften eines einzelnen sind jedoch nicht möglich, wahrscheinlich u. A. aus Aspekten des Datenschutzes. Jedoch im Fehlerfall gibt es mehr Informationen, denn hier werden spezifische Daten (Stacktrace, genaue Browserversion, Uhrzeit, ...) zum Fehler sowie zum Client erhoben und in der Oberfläche dargestellt. Diese Informationen sind gut zum Nachvollziehen, dass ein Fehler aufgetreten ist und was die Rahmenbedingungen sind - jedoch sind die Informationen nicht ausreichend um ein klares Bild des Fehlers zu erhalten. Dies könnte man mit dem Log-Management von New Relic komplementieren, jedoch müsste man diese selber erheben, an einen eigenen Server senden und diese dann New Relic weiterleiten. Des Weiteren wird der Agent standardmäßig von AdBlockern blockiert, sowie von der "Enhanced Tracking Protection" im Mozilla Firefox. Es ist nicht möglich New Relics Dienst selber zu hosten, es ist also eine sog. "OnPremise"-Lösung nicht möglich, weiterhin wird auch eine Weiterleitung über einen eigens bereitgestellten Proxy nicht unterstützt.

New Relic gibt an, dass Daten nach dem OpenTelemetry Standard selber erfasst und an New Relic gesendet werden können, ohne eine proprietäre Software [New20a]. Leider ist dieses Feature in der Testversion, die zum Evaluieren benutzt wurde, nicht enthalten und kann somit nicht bestätigt werden. Es sind jedoch offizielle und quelloffen veröffentlichte Exporter für New Relic verfügbar für .NET, Python und Java [Ope20d].

3.2.2.3 Dynatrace

Dynatrace [Dyn20b] ist ein SaaS des gleichnamigen Unternehmens, welcher Betreiber von Softwareprojekten dabei unterstützt das Verhalten ihrer Anwendungen zu überwachen. Der Dienst konzentriert sich auf System Monitoring, APM und RUM und erfasst die notwendigen Daten mit proprietären Lösungen, dem "OneAgent". Ganz ähnlich wie New Relic unterstützt Dynatrace neben den Kernfunktionalitäten auch Log Management, Synthetic Monitoring, Tracing und Error Monitoring.

Bei Dynatrace wurde die 14-tägige Testversion aufgesetzt und evaluiert. Wie zuvor genannt, erfolgt die Datenerhebung über den Dynatrace OneAgent, welcher genauso wie New Relics Agent kontinuierlich Daten sendet. Die Oberfläche von Dynatrace stellt auch ungefähr die selben Informationen dar wie New Relic, wobei Dynatrace das Ganze visuell ansprechender darstellt. Dynatrace bietet auch die Funktionalität vom Error Monitoring aber leidet unter demselben Problem wie New Relic: zu wenig Kontextinformationen.

Der Dynatrace OneAgent, wird standardmäßig von AdBlockern blockiert, aber nicht wie New Relic von der "Enhanced Tracking Protection" des Mozilla Firefox. Dynatrace kann zudem damit punkten, dass es als OnPremise-Lösung verfügbar ist, sodass Kunden genau bestimmen können, wo die Daten verarbeitet und gespeichert werden.

Dynatrace ist dem OpenTelemetry Team beigetreten und hat angegeben, an der Weiterentwicklung mitzuhelfen [Dyn20a]. Eine Integration des Dienstes Dynatrace ins Ökosystem von OTel gibt es jedoch noch nicht.

3.2.2.4 Sentry

Sentry [Fun20] ist ein SaaS Produkt der Functional Software Inc, welches sich auf das Error Monitoring spezialisiert. Deshalb beschränken sich die Kernfunktionalitäten auf das Error Monitoring, auch wenn von anderen Gebieten einige Aspekte präsent sind, stellen diese keine eigens abgeschlossene Funktionalität dar.

Für Sentry wurde die kostenlose Version aufgesetzt und evaluiert. Sentry bietet bei NPM quelloffene Pakete an [Sen20a], um Fehler zu erfassen und an Sentry zu melden. Es werden Pakete für folgende Frameworks bereitgestellt: JavaScript, Angular, AngularJS, Backbone, Ember, Gatsby, React und Vue. Das Aufsetzen stellt sich einfach dar und ermöglicht einige Konfigurations- und Verarbeitungsoptionen, bspw. können sensible Daten aus den Datenpaketen entfernt werden oder weitere Informationen gemeldet werden. Anders als bei den beiden vorherigen Tools wird zu Sentry nur kommuniziert, wenn ein Fehler auftritt. Hierbei werden dafür aber umso mehr Daten erhoben: Detaillierte Klickpfade des Nutzers, Logmeldungen der Browserkonsole sowie die Informationen, die auch die anderen Tools bereitstellen.

Gemeldete Fehler werden in "Issues" umgewandelt, welche einem Fehlerticket entsprechen und in der Oberfläche Funktionen zum Zuweisen und zum Nachhalten der Behebung bieten. Sentry versucht die eingetroffenen Fehler in bestehenden Issues zu gruppieren, sodass jeweils zusammengehörige Fehler auch zusammen behandelt werden. Bei Sentry fehlt die ganzheitliche Nachvollziehbarkeit, aber dafür sind die gesammelten Fehlerinformationen zahlreich und helfen beim Nachvollziehen.

Der Quellcode von Sentry wurde veröffentlicht und weiterhin wird bei Sentry auch eine OnPremise-Lösung, basierend auf Docker, angeboten [Sen20b].

3.2.2.5 LogRocket

LogRocket [Log20] ist ein SaaS Produkt des gleichnamigen Unternehmens und konzentriert sich auf detailliertes Session-Replay von JavaScript-basierten Clientanwendungen, um Probleme zu identifizieren, zu nachvollziehen und lösen zu können. Session-Replay ist auch die einzig identifizierbare Kernfunktionalität, die LogRocket aufweist.

Für die Evaluierung wurde die kostenlose Testversion von LogRocket verwendet. Zur Datenerhebung wird das Paket `logrocket` von NPM hinzugezogen und nach der Initialisierung sammelt es eigenständig die notwendigen Daten. Mit Hilfe der gesammelten Daten wird die gesamte Sitzung des Nutzers nachgestellt, hierbei sieht man die Anwendung, den Nutzer und seine Aktionen, die Netzwerkaufrufe sowie das DOM. Die Nachstellung wird video-ähnlich aufbereitet und erlaubt ein präzises Nachvollziehen der zeitlichen Reihenfolge und Bedeutung (vgl. Abbildung 3.3).

Neben dem JavaScript SDK bietet LogRocket quelloffenene Plugins für folgende Bibiotheken: Redux, React, MobX, Vuex, ngrx, React Native. LogRocket ist zudem als OnPremise-Lösung verfügbar. Zusätzlich bietet LogRocket auch einige Integrationen für andere Tools, wie z.B. Sentry. Die Integration mit Sentry wurde auch evaluiert und dadurch wurde ermöglicht, dass man von einem Sentry Issue direkt auf das Session-Replay des konkreten Fehlerfalls gehen konnte.

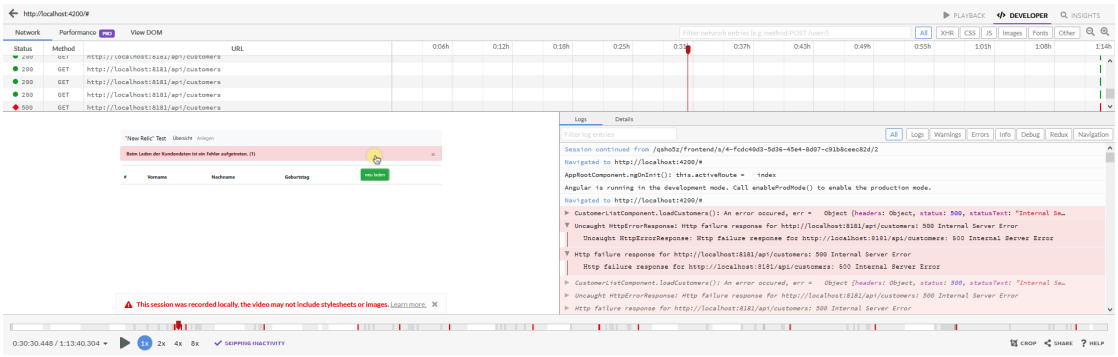

Abb. 3.3: Beispiel eines Session-Replays bei LogRocket

3.2.2.6 Splunk

Beschreibung zu grundlegenden Eigenschaften von Splunk

3.2.2.7 Honeycomb

Honeycomb [Hou20] ist ein SaaS der Hound Technology Inc. und verspricht die Speicherung vieler (Tracing-)Daten und darauf basierend effiziente Abfragen zu ermöglichen. Es ist hauptsächlich als Tracingdienst anzusehen, womit jedoch auch Aspekte des APM, RUM und Error Monitoring mit abgebildet werden können.

Honeycomb wurde mit der kostenlosen Version evaluiert. Honeycomb bietet sog. "Beelines" an, welche Werkzeuge zur automatischen Datenerfassung sind. Diese Beelines sind aber nur für Node.js, Go, Python, Java, Ruby und Rails verfügbar, aber nicht JavaScript im Browser. Deshalb wurden zur Evaluierung Werkzeuge von OpenTelemetry hinzugezogen und zusätzlich musste nur noch ein Exporter für Honeycomb geschrieben werden. Honeycomb wirkt komplett anders als New Relic und Dynatrace, denn es ist nicht ausgefertigt mit konkreten Graphen und Darstellungen, sondern bietet eine Oberfläche mit dem man auf Basis der Daten selber Abfragen, Graphen und andere Darstellungen selber erstellen kann. Auch ist es nicht mit Jaeger, welches gleich näher betrachtet wird, vergleichbar, denn es bietet deutlich mehr Möglichkeiten als striktes Tracing.

3.2.2.8 Jaeger

Jaeger wurde 2017 als ein Projekt der CNCF gestartet [Jae20]. Es ist ein System für verteiltes Tracing von der Datensammlung bis hin zur Visualisierung, es unterstützt und implementiert den Standard OpenTracing. Eine Unterstützung des OpenTelemetry Standards ist derzeit im Gange. Weiterhin kann Jaeger dazu benutzt werden, Metriken nach Prometheus [Pro20] zu exportieren, einem weiteren CNCF Projekt zur Speicherung und Visualisierung von Daten.

Jaeger wurde nicht aufgesetzt, sondern über Dokumentation und Kommunikation mit Kollegen der Open Knowledge evaluiert. Wie zuvor bei Honeycomb beschrieben, beschränkt sich Jaeger sehr auf Tracing, aber kann sich dafür auf dieses Gebiet spezialisieren und bietet eine gesamte Infrastruktur.

3.2.2.9 Weiteres

Bei meiner Recherche und Evaluierung wurden nicht alle auf dem Markt verfügbaren Werkzeuge und Technologien tiefergehend betrachtet. Deshalb werden weitere Funde, die nicht betrachtet wurden, hier kurz notiert:

- APM & RUM: AppDynamics, DataDog
- Error Monitoring: Airbrake, Instabug, Rollbar, Bugsnag, TrackJS
- Tracing: Google Cloud Trace, Zipkin

Auch diese Auflistung stellt nicht die komplette Bandbreite an Werkzeugen und Technologien dar und eine vorherige Betrachtung ist nicht als direkte Empfehlung zu verstehen.

3.2.3 Literatur

4 Erstellung Proof-of-Concept

4.1 Kriterien

Hier soll definiert werden, welche Kriterien der PoC erfüllen soll und wenn möglich sollen zusätzlich Möglichkeiten zur Überprüfung dieser Kriterien definiert werden.

4.2 Vorstellung der Demoanwendung

In diesem Abschnitt soll die Demoanwendung vorgestellt werden, anhand dessen das Proof-of-Concept erstellt wird. Damit das Proof-of-Concept erstellt werden kann, muss die Demoanwendung die zuvor beschriebenen Probleme aufweisen, hierbei sollen die Probleme möglichst realitätsnah sein und nicht frei erfunden.

4.3 Konzept

4.3.1 Architektur

Hier soll die grobe Architektur geplant werden, welche Komponente es gibt und wie diese kommunizieren sollen.

4.3.2 Datenverarbeitung

4.3.2.1 Erhebung

Wie werden die Daten erhoben (Nennung der verwendeten Methoden!)? Wie gelangen die Daten an eine auswertende Komponente?

4.3.2.2 Auswertung

Wie werden die Daten zusammengefasst und ausgewertet? Wie gelangt das Ergebnis an die darstellende Komponente?

4.3.2.3 Visualisierung

Wie werden den Stakeholdern die Informationen präsentiert?

4.4 Implementierung

Auf Basis des Konzeptes soll nun eine Implementierung erfolgen.

4.4.1 Technologie-Stack

5 Ergebnis

5.1 Demonstration

Nachdem nun eine Implementierung steht, soll die Erweiterung auf nicht-technische Weise veranschaulicht werden. Hier soll dargestellt werden, wie die Nachvollziehbarkeit nun verbessert worden ist.

5.2 Kriterien

Die zuvor definierten Kriterien in 4.1 sollen hier überprüft werden.

5.3 Übertragbarkeit

Wie gut lassen sich die ermittelten Ergebnisse im PoC auf andere Projekte im selben Umfeld übertragen?

6 Abschluss

6.1 Fazit

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6.2 Ausblick

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

7 Anhang

7.1 Studien zur Browserkompatibilität

Im Unterabschnitt 2.1.1 wurde die Anzahl an Studien zur Browserkompatiblität dargestellt. Die Daten hierfür wurden über die Literatursuchmaschine “Google Scholar“ am 29.10.2020 abgerufen

Für die Suche wurde folgender Suchterm benutzt:

```
"cross browser" compatibility|incompatibility|inconsistency|XBI
```

Die Trefferanzahl für ein spezielles Jahr wurde jeweils als Datenpunkt benutzt. Dies soll dazu dienen, um einen ungefähren Trend der Literatur zu erkennen. Der Prognosewert fürs Jahr 2020 wurde pauschal so berechnet, dass in den letzten zwei Monaten proportional gleich viele Veröffentlichungen erscheinen, also: $\left\lceil \frac{136*12}{10} \right\rceil$

Jahr	Treffer
2015	299
2016	246
2017	286
2018	238
2019	187
2020	136
2020 (Prognose)	164

Tab. 7.1: Suchtreffer zu Studien über Browserkompatibilität

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Dortmund, am ....
(Unterschrift)

Abkürzungs- und Erklärungsverzeichnis

Ajax Asynchronous JavaScript and XML

CDN Content Delivery Network

Clientseitiges Rendering Der Server stellt dem Client lediglich die Logik und die notwendigen Daten bereit, die eigentliche Inhaltsgenerierung geschieht im Client. Für ein Beispiel siehe Unterabschnitt 2.2.2

CNCF Cloud Native Computing Foundation

CORS Cross-Origin Resource Sharing

CSP Content-Security-Policy

OTel OpenTelemetry

PoC Proof-of-Concept

SaaS Software-as-a-Service

Serverseitiges Rendering Die darzustellenden Inhalte, werden beim Server generiert und der Client stellt diese dar. Beispielsweise sind Anwendungen mit PHP oder auch eine Java Web Application

W3C World Wide Web Consortium

XHR XMLHttpRequest

XSS Cross-Site-Scripting

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Literaturverzeichnis

[ABC+16] Ahmed, Tarek M. ; Bezemer, Cor-Paul ; Chen, Tse-Hsun ; Hassan, Ahmed E. ; Shang, Weiyi: Studying the effectiveness of application performance management (APM) tools for detecting performance regressions for web applications: an experience report. In: *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)* IEEE, 2016, S. 1–12

[AMWR20] Asrohah, Hanun ; Milad, Mohammad K. ; Wibowo, Achmad T. ; Rhofita, Erry I.: Improvement of academic services using mobile technology based on single page application. In: *Telfor Journal* 12 (2020), Nr. 1, S. 62–66

[Blu15] Bluesmoon: *Flowchart showing Simple and Preflight XHR.svg.* `https://commons.wikimedia.org/wiki/File:Flowchart_showing_Simple_and_Preflight_XHR.svg`, 2015. – [Online; abgerufen am 09.11.2020]

[Dyn20a] Dynatrace: *Dynatrace joins the OpenTelemetry project.* `https://www.dynatrace.com/news/blog/dynatrace-joins-the-opentelemetry-project/`, 2020. – [Online; abgerufen am 20.11.2020]

[Dyn20b] Dynatrace: *The Leader in Cloud Monitoring | Dynatrace.* `https://www.dynatrace.com/`, 2020. – [Online; abgerufen am 20.11.2020]

[Dyn20c] Dynatrace: *What is OpenTelemetry? Everything you wanted to know.* `https://www.dynatrace.com/news/blog/what-is-opentelemetry/`, 2020. – [Online; abgerufen am 20.11.2020]

[Fac20] Facebook: *React - A JavaScript library for building user interfaces.* `https://reactjs.org`, 2020. – [Online; abgerufen am 12.10.2020]

[Fil20] Filipe, Ricardo Ângelo S.: *Client-Side Monitoring of Distributed Systems*, Universidade de Coimbra, Diss., 2020

[Fre91] Freedman, Roy S.: Testability of software components. In: *IEEE transactions on Software Engineering* 17 (1991), Nr. 6, S. 553–564

[Fun20] Functional Software: *About Sentry | Sentry.* `https://sentry.io/about/`, 2020. – [Online; abgerufen am 23.11.2020]

[GB20] Greif, Sacha ; Benitte, Raphaël: *The State of JavaScript 2019*. https://2019.stateofjs.com/, 2020. – [Online; abgerufen am 29.10.2020]

[Goo20] Google: *Angular*. https://angular.io, 2020. – [Online; abgerufen am 12.10.2020]

[Hop06] Hopmann, Alex: *The story of XMLHTTP*. https://web.archive.org/web/20070623125327/http://www.alexhopmann.com/xmlhttp.htm, 2006. – [Online; abgerufen am 27.10.2020]

[Hou20] Hound Technology: *Why Honeycomb - Honeycomb*. https://www.honeycomb.io/why-honeycomb/, 2020. – [Online; abgerufen am 23.11.2020]

[Jae20] Jaeger Authors, The: *Jaeger: open source, end-to-end distributed tracing*. https://www.jaegertracing.io/, 2020. – [Online; abgerufen am 19.11.2020]

[Ká60] Kálmán, Rudolf E.: On the general theory of control systems. In: *Proceedings First International Conference on Automatic Control, Moscow, USSR*, 1960, S. 481–492

[Log20] LogRocket: *LogRocket | Logging and Session Replay for JavaScript Apps*. https://logrocket.com/, 2020. – [Online; abgerufen am 23.11.2020]

[Mic20a] Microsoft: *Microsoft 365 apps say farewell to Internet Explorer 11 and Windows 10 sunsets Microsoft Edge Legacy*. https://techcommunity.microsoft.com/t5/microsoft-365-blog/microsoft-365-apps-say-farewell-to-internet-explorer-11-and/ba-p/1591666, 2020. – [Online; abgerufen am 29.10.2020]

[Mic20b] Microsoft: *New year, new browser – The new Microsoft Edge is out of preview and now available for download*. https://blogs.windows.com/windowsexperience/2020/01/15/new-year-new-browser-the-new-microsoft-edge-is-out-of-preview-and-now-available-15, 2020. – [Online; abgerufen am 29.10.2020]

[MM20a] Mozilla ; Mitwirkende individuelle: *console - Web APIs | MDN*. https://developer.mozilla.org/en-US/docs/Web/API/Console, 2020. – [Online; abgerufen am 19.10.2020]

[MM20b] Mozilla ; Mitwirkende individuelle: *Content Security Policy (CSP) - HTTP | MDN*. https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP/, 2020. – [Online; abgerufen am 15.10.2020]

[MM20c] Mozilla ; Mitwirkende individuelle: *Cross-Origin Resource Sharing (CORS) - HTTP | MDN*. https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS, 2020. – [Online; abgerufen am 15.10.2020]

[New20a] New Relic: *Announcing OpenTelemetry Beta support in New Relic One - New Relic Blog.* https://blog.newrelic.com/product-news/opentelemetry-beta-support-new-relic-one/, 2020. – [Online; abgerufen am 20.11.2020]

[New20b] New Relic: *New Relic | Deliver more perfect software.* https://opentelemetry.io/registry/, 2020. – [Online; abgerufen am 20.11.2020]

[OKSK15] Oren, Yossef ; Kemerlis, Vasileios P. ; Sethumadhavan, Simha ; Keromytis, Angelos D.: The spy in the sandbox: Practical cache attacks in javascript and their implications. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, S. 1406–1418

[OPBW06] Oshry, Matt ; Porter, Brad ; Bodell, Michael ; W3C, World Wide Web C.: *Authorizing Read Access to XML Content Using the <?access-control?> Processing Instruction 1.0.* https://www.w3.org/TR/2006/WD-access-control-20060517/, 2006. – [Online; abgerufen am 27.10.2020]

[Ope20a] OpenCensus: *OpenCensus.* https://opencensus.io/introduction/#overview, 2020. – [Online; abgerufen am 19.11.2020]

[Ope20b] OpenTelemetry: *About | OpenTelemetry.* https://opentelemetry.io/about/, 2020. – [Online; abgerufen am 19.11.2020]

[Ope20c] OpenTelemetry: *opentelemetry-specification/unified-collection.png at 8e7b2cc17f2c572282c4e5e4d3cc54401749d8ff · open-telemetry/opentelemetry-specification.* https://github.com/open-telemetry/opentelemetry-specification/blob/8e7b2cc17f2c572282c4e5e4d3cc54401749d8ff/specification/logs/img/unified-collection.png, 2020. – [Online; abgerufen am 19.11.2020]

[Ope20d] OpenTelemetry: *Registry | OpenTelemetry.* https://opentelemetry.io/registry/, 2020. – [Online; abgerufen am 20.11.2020]

[Ope20e] OpenTelemetry: *Write guidelines and specification for logging libraries to support OpenTelemetry-compliant logs · Issue #894 · open-telemetry/opentelemetry-specification.* https://github.com/open-telemetry/opentelemetry-specification/issues/894, 2020. – [Online; abgerufen am 19.11.2020]

[Ope20f] OpenTracing: *What is Distributed Tracing?* https://opentracing.io/docs/overview/what-is-tracing/, 2020. – [Online; abgerufen am 19.11.2020]

[Ora20] Oracle: *Java Debug Wire Protocol*. `@https://download.java.net/java/GA/jdk14/docs/specs/jdwp/jdwp-spec.html`, 2020. – [Online; abgerufen am 23.10.2020]

[Pow06] Powers, Shelley: *Learning JavaScript*. O'Reilly Media Inc., 2006

[Pro20] Prometheus Authors, The: *Prometheus - Monitoring system & time series database*. `https://prometheus.io/`, 2020. – [Online; abgerufen am 19.11.2020]

[Ran09] Ranganathan, Arun: *cross-site xmlhttprequest with CORS*. `https://hacks.mozilla.org/2009/07/cross-site-xmlhttprequest-with-cors/`, 2009. – [Online; abgerufen am 27.10.2020]

[SD13] Shreateh, Khalil ; Dewey, Caitlyn: Mark Zuckerberg's Facebook page was hacked by an unemployed Web developer. In: *The Washington Post* (2013), Aug

[Sen20a] Sentry: *getsentry/sentry-javascript: Official Sentry SDKs for Javascript*. `https://github.com/getsentry/sentry-javascript`, 2020. – [Online; abgerufen am 23.11.2020]

[Sen20b] Sentry: *Self-Hosted Sentry | Sentry Developer Documentation*. `https://develop.sentry.dev/self-hosted/`, 2020. – [Online; abgerufen am 23.11.2020]

[Sta20] Statista: *Global market share held by leading desktop internet browsers from January 2015 to June 2020*. `https://www.statista.com/statistics/544400/market-share-of-internet-browsers-desktop/`, September 2020. – [Online; abgerufen am 29.10.2020]

[STM+20] Scrocca, Mario ; Tommasini, Riccardo ; Margara, Alessandro ; Valle, Emanuele D. ; Sakr, Sherif: The Kaiju Project: Enabling Event-Driven Observability. In: *Proceedings of the 14th ACM International Conference on Distributed and Event-based Systems*, 2020, S. 85–96

[W3C06] W3C, World Wide Web C.: *The XMLHttpRequest Object*. `https://www.w3.org/TR/2006/WD-XMLHttpRequest-20060405/`, 2006. – [Online; abgerufen am 27.10.2020]

[W3C20] W3C, World Wide Web C.: *Abou W3C Standards*. `https://www.w3.org/standards/about.html`, 2020. – [Online; abgerufen am 272910.2020]

[YM20] You, Evan ; Mitwirkende individuelle: *Vue.js*. `https://vuejs.org/`, 2020. – [Online; abgerufen am 12.10.2020]