Bachelorthesis

Nachvollziehbarkeit von Nutzerinteraktion und Anwendungsverhalten am Beispiel JavaScript-basierter Webapplikationen

An der Fachhochschule Dortmund
im Fachbereich Informatik
Studiengang Software- und Systemtechnik, Vertiefung Softwaretechnik
erstellte Bachelorthesis
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, 29. Oktober 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 Brief an. Im Unterschied zum klassischen Versicherer verfügt ein Direktversicherer jedoch über keinen Außendienst oder Geschäftsstellen, wo 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 der Standard. 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, steht man vor einer Herausforderung. Während man bei Server-Anwendungen Fehlermeldungen in den Log-Dateien einsehen kann, 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

Ziel dieser Arbeit ist es, eine Möglichkeit zu schaffen, dass die Stakeholder die Interaktionen eines Nutzers und das Verhalten einer Webapplikation nachvollziehen können. Dieses Ziel wird unter dem Begriff "Nachvollziehbarkeit" zusammengefasst. Folgende Fragen sollen im Zuge der Ausarbeitung beantwortet werden:

1. Wie sieht das Projektumfeld aus?
2. Was bedeutet Nachvollziehbarkeit (bei Webapplikationen)?
3. Warum ist Nachvollziehbarkeit wichtig?
4. Wie kann eine gute Nachvollziehbarkeit erreicht werden?
   a) Was wird hierzu benötigt?
   b) Wie kann Nutzerverhalten nachvollzogen werden?
   c) Wie können Fehler nachgestellt werden?
5. Wie ist eine Webapplikation zu erweitern, um dies zu erreichen?
   (Hierbei wird eine Demoanwendung untersucht.)
   a) Was für Technologien helfen hierbei?
   b) Was sind die Auswirkungen für den Nutzer?
      i. Wird die Leistung der Webapplikation beeinträchtigt?
      ii. Wie wird mit seinen Daten umgegangen (Stichwort DSGVO)?

1.2.1 Abgrenzung

Bei der Betrachtung von Webapplikationen, wird sich auf Single-Page-Applications (SPAs) konzentriert, denn hier bewegt sich das Projektumfeld der Open Knowledge. Bei der Implementierung soll ein Proof-of-Concept für eine Demoanwendung das Ziel sein, eine allgemeingültige Lösung ist nicht anzustreben. Bei der Betrachtung der Datenerhebung und -verarbeitung ist eine volle Konformität mit der DSGVO nicht zu prüfen.

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 Stakeholder 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.

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. Das Proof-of-Concept erfolgt auf Basis einer bestehenden Webapplikation der Open Knowledge GmbH.

Ist ein PoC 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-Incompatabilities (XIB) genannt, hat jedoch durch einen 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. Was dafür spricht, dass das Problem nicht mehr so präsent ist, wie zuvor.

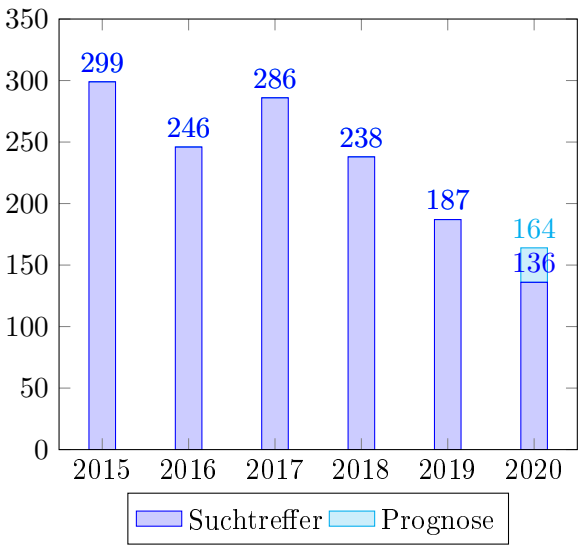

Abb. 2.1: Studien zur Browserkompatibilität, eigene Darstellung (vgl. 6.1)

Im Jahr 2020 gab es weitere Entwicklungen, die die Kompatibilität zwischen Browsern erhöhte. Microsoft hat 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]. Zum 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, um den Funktionsumfang zu erweitern: Ajax (Asynchronous JavaScript and XML). Ajax erlaubt die Datenabfrage von Webservern mittels JavaScript. Hierdurch wird ermöglicht, dass Inhalte auf Webseiten dynamisch abgefragt und dargestellt wurden, 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.

Im folgenden Abschnitt werden, für diese Arbeit relevante, Sicherheitsvorkehrungen von Browsern vorgestellt und beschrieben - darunter auch Cross-Origin Resource Sharing.

2.1.3 Sicherheitsvorkehrungen

2.1.3.1 Cross-Origin Resource Sharing (CORS)

Das Konzept von CORS stellt sicher, dass aus einer JavaScript-Umgebung heraus keine Ressourcen von Webservern angefragt werden, außer diese Webserver stimmen der An-

frage zu [MM20c]. Folgendes vereinfachtes Beispiel soll den Nutzen und den generellen Ablauf von CORS näher erläutern:

> Ein Nutzer ruft eine Webapplikation auf, welche unter `localhost:3000` bereitgestellt wird. Diese Webapplikation sendet, für den Nutzer unwissend, Anfragen an einen Facebook Dienst. Dies lässt der Browser aber nur zu, wenn der Dienst von Facebook explizit bestätigt, dass eine Anfrage von `localhost:3000` diese Aktion ausführen darf.
>
> Dafür sendet der Browser eine OPTIONS-Anfrage, in der die Herkunft ("Origin") der Anfrage notiert ist. Der Facebook Dienst antwortet daraufhin mit den entsprechenden CORS-Headern und gibt somit an, ob die Anfrage von dieser Origin aus erlaubt ist. Nun prüft der Browser, ob die von Facebook übermittelten Origins übereinstimmen. Ist dies nicht der Fall, wird die Anfrage im Browser blockiert und im JavaScript Kontext schlägt dieser fehl. Details zum Fehler werden der JavaScript-Umgebung bewusst enthalten.

2.1.3.2 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.

2.1.4 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.

2.1.5 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.

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.

Im nächsten Kapitel werden Methoden und Konzepte beschrieben, wie man in Softwareprojekten die Nachvollziehbarkeit verbessern kann.

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. Hier könnte unter anderem ***OpenTelemetry*** *betrachtet werden.*

3.1 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 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?*

3.3 Monitoring

Folgende Fragen sollen zur Methode beantwortet werden

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

3.4 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?*

3.5 Tracing

Folgende Fragen sollen zur Methode beantwortet werden

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

3.6 Werkzeuge und Technologien

Basierend auf dem Grundwissen über die Methoden und Praktiken, soll nun der Stand der Technik erörtert werden. Hierbei sollen Werkzeuge und Technologien und ihre Ansätze hervorgehoben werden und mit Hilfe welcher Methoden sie welches Ziel erreichen.

4 Erstellung Proof-of-Concept

4.1 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.

4.2 Konzept

4.2.1 Architektur

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

4.2.2 Datenverarbeitung

4.2.2.1 Erhebung

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

4.2.2.2 Auswertung

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

4.2.2.3 Visualisierung

Wie werden den Stakeholdern die Informationen präsentiert?

4.3 Implementierung

Auf Basis des Konzeptes soll nun eine Implementierung erfolgen.

4.3.1 Technologie-Stack

4.4 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 Abschluss

5.1 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.

5.2 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.

6 Anhang

6.1 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. 6.1: 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

CORS Cross-Origin Resource Sharing

CSP Content-Security-Policy

DSGVO Datenschutz Grundverordnung

PoC Proof-of-Concept

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

SPA Single Page Application

W3C World Wide Web Consortium

XHR XMLHttpRequest

XSS Cross-Site-Scripting

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Literaturverzeichnis

[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

[Fac20] Facebook Inc.: *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

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

[Goo20] Google LLC: *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]

[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

[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-availabl
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]

[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]

[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

[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

[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]