

Evaluierung von Techniken zur parallel-synchronen Bedienung einer Web-Applikation auf verschiedenen mobilen Endgeräten

vorgelegt von

Adrian Randhahn

EDV.Nr.:744818

dem Fachbereich VI – Informatik und Medien –
der Beuth Hochschule für Technik Berlin vorgelegte Bachelorarbeit
zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

im Studiengang

Medieninformatik

Tag der Abgabe 10. März 2014

Gutachter

Prof. Knabe

Beuth Hochschule für Technik

Prof. Dr. Wambach

Beuth Hochschule für Technik

Erklärung

Ich versichere, dass ich diese Abschlussarbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht. Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Datum

Unterschrift

Entwurf

Sperrvermerk

Die vorliegende Arbeit beinhaltet interne und vertrauliche Informationen der Firma New Image Systems GmbH. Die Weitergabe des Inhalts der Arbeit im Gesamten oder in Teilen sowie das Anfertigen von Kopien oder Abschriften - auch in digitaler Form - sind grundsätzlich untersagt. Ausnahmen bedürfen der schriftlichen Genehmigung der Firma New Image Systems GmbH.

Entwurf

Rechtliches

Alle in dieser Arbeit genannten Unternehmens- und Produktbezeichnungen sind in der Regel geschützte Marken- oder Warenzeichen. Auch ohne besondere Kennzeichnung sind diese nicht frei von Rechten Dritter zu betrachten. Alle erwähnten Marken- oder Warenzeichen unterliegen uneingeschränkt der länderspezifischen Schutzbestimmungen und den Besitzrechten der jeweiligen eingetragenen Eigentümern.

Kurzfassung

Es sollen bestehende Technologien analysiert werden in Bezug auf die Prozessoptimierung innerhalb der Qualitätssicherung im Entstehungszyklus von Webapplikationen, Desweiteren sollen Alleinstehende Frameworks dahingehend untersucht werden auf ihren Nutzfaktor zur Erstellung einer Software die eben diese Anforderung erfüllt.

Abstract

TOREMOVE->
english
<-TOREMOVE

translation

Entwurf

Inhaltsverzeichnis

| | | |
|----------|---|----------|
| 1 | Einleitung | 2 |
| 2 | Aufgabenstellung | 5 |
| 2.1 | Problemstellung | 5 |
| 2.2 | Annahmen und Einschränkungen | 7 |
| 2.3 | Zielsetzung | 7 |
| 2.4 | Abgrenzungskriterien | 7 |
| 3 | Grundlagen | 8 |
| 3.1 | Begriffsklärung | 8 |
| 3.1.1 | parallel-synchron | 8 |
| 3.1.2 | Web-Applikation | 8 |
| 3.1.3 | HTML | 8 |
| 3.1.4 | Webbrowser | 8 |
| 3.1.5 | Desktopcomputer / Desktops | 8 |
| 3.1.6 | Mobiles Endgerät | 8 |
| 3.1.7 | Javascript | 9 |
| 3.1.8 | Framework | 9 |
| 3.1.9 | Nodejs | 9 |
| 3.1.10 | PHP | 9 |
| 3.1.11 | NPM | 9 |
| 3.1.12 | Qualitätssicherung | 9 |
| 3.1.13 | VirtualBox / virtuelle Umgebung | 9 |
| 3.1.14 | Smartphone | 9 |
| 3.1.15 | Tablet | 9 |
| 3.1.16 | Panorama / Portrait View | 9 |
| 3.1.17 | Pixel | 9 |
| 3.1.18 | Auflösung | 9 |
| 3.1.19 | Event | 9 |
| 3.1.20 | DOM | 9 |
| 3.1.21 | Apache | 9 |
| 3.2 | technischer Aufbau | 9 |
| 3.3 | Komponenten | 9 |
| 3.3.1 | Raspberry Pi | 9 |
| 3.3.2 | Hardware | 9 |

| | | |
|----------|--|-----------|
| 4 | Technologien | 10 |
| 4.1 | Ghostlab | 10 |
| 4.2 | NodeJS | 11 |
| 4.3 | Zombie.js | 11 |
| 4.4 | W3C Touch Events Extensions | 11 |
| 4.5 | Phantom Limb | 11 |
| 4.6 | jQuery UI Touch Punch | 11 |
| 4.7 | jQuery Touchit | 11 |
| 4.8 | NPM touchit | 11 |
| 4.9 | Adobe Edge Inspect | 11 |
| 4.10 | Remote Preview | 11 |
| 4.11 | Browser-Sync | 11 |
| 4.12 | Eigenes Framework | 11 |
| 5 | Evaluation der Techniken | 12 |
| 5.1 | Ghostlab | 12 |
| 5.1.1 | Einrichtung der Testumgebung | 12 |
| 5.2 | Adobe Edge Inspect | 13 |
| 5.3 | Remote Preview | 13 |
| 5.4 | Browser-Sync | 13 |
| 5.5 | Eigenes Framework | 13 |
| 5.5.1 | Systementwurf | 13 |
| 6 | Ausblick | 14 |
| 7 | Helpers | 15 |
| 7.1 | quote | 15 |
| 7.2 | longquote | 15 |
| 7.3 | fussnote | 15 |

Abbildungsverzeichnis

| | | |
|-----|---|---|
| 1.1 | Entwicklungsprozess | 3 |
| 2.1 | Qualitätssicherung Testszenario | 6 |

Entwurf

Tabellenverzeichnis

| | | |
|-----|--|----|
| 4.1 | von Ghostlab getestete Browser (stand 10.03.2014, Version 1.2.3) | 10 |
| 5.1 | test | 12 |

Entwurf

1 Einleitung

In der modernen Webentwicklung durchläuft eine Anwendung verschiedene Etappen eines Entwicklungszykluses. Er beginnt bei einem Auftrag oder einer Idee, darauf folgt dann die Spezifikation einzelner Usecases¹. Im Anschluss folgt in der Regel die Entwicklung und Implementation² der einzelnen Komponenten. Am Ende der jeweiligen Implementationsphase durchläuft das Produkt³ die Qualitätskontrolle. Sollten in diesem Abschnitt Fehler auftreten wird das Produkt dem Entwickler zur erneuten Bearbeitung vorgelegt. Dieser Vorgang kann sich beliebig oft

wiederholen. Bei großen und komplexen Softwaresystemen ist es trotz zeitgemäßer Implementierung nicht immer Ausgeschlossen, dass Kaskadierungsfehler⁴ entstehen. Aus Sicht der Qualitätssicherung ist dies ein lästiges Problem, da diese nach jedem erneuten Modifikationsvorganges eines Softwaresegments einen größeren Segmentblock, wenn nicht sogar das gesamte Softwareystem erneut testen muss. Bei der Entwicklung auf und für mobile Endgeräte⁵ kommt noch ein erschwe-

render Faktor hinzu, nämlich die diversen, verschiedenen Bildschirmauflösungen. Diese können nicht nur die Darstellung des Inhaltes beeinflussen, sondern auch daraus folgend die Interaktionskonformität beeinflussen.

Im Optimalfall wird die Software erst nach vollständiger Homogenität auf allen unterstützen Geräten freigegeben.

Dieser zyklisch wiederkehrende Prozessablauf ist sehr Zeitintensiv und nimmt linear mit der Anzahl der zu testenden Geräte zu.

Das Ergebnis dieser Forschungsarbeit soll zeigen, wie verschiedene Softwareframeworks die Zeit, die in die Qualitätssicherung investiert wird, beeinflussen können, indem sie die Steuerung diverser Geräte parallel-synchron steuern. Die Evaluierung soll zeigen wo die Vorteile und Nachteile der einzelnen Werkzeuge liegen. Weiterhin soll gezeigt werden ob aktuelle Frameworks erweiterbar sind um beispielsweise automatisierte Testunits zu implementieren.

¹Szenario oder auch Anwendungsfall

²Einbindung

³hier: einzelne Softwarekomponente

⁴Fehler die nicht im eigentlichen Segment auftreten, sondern eine oder mehr Ebenen weiter unten in der Systemhierarchie

⁵Smartphones, Tablets oder Ähnliche

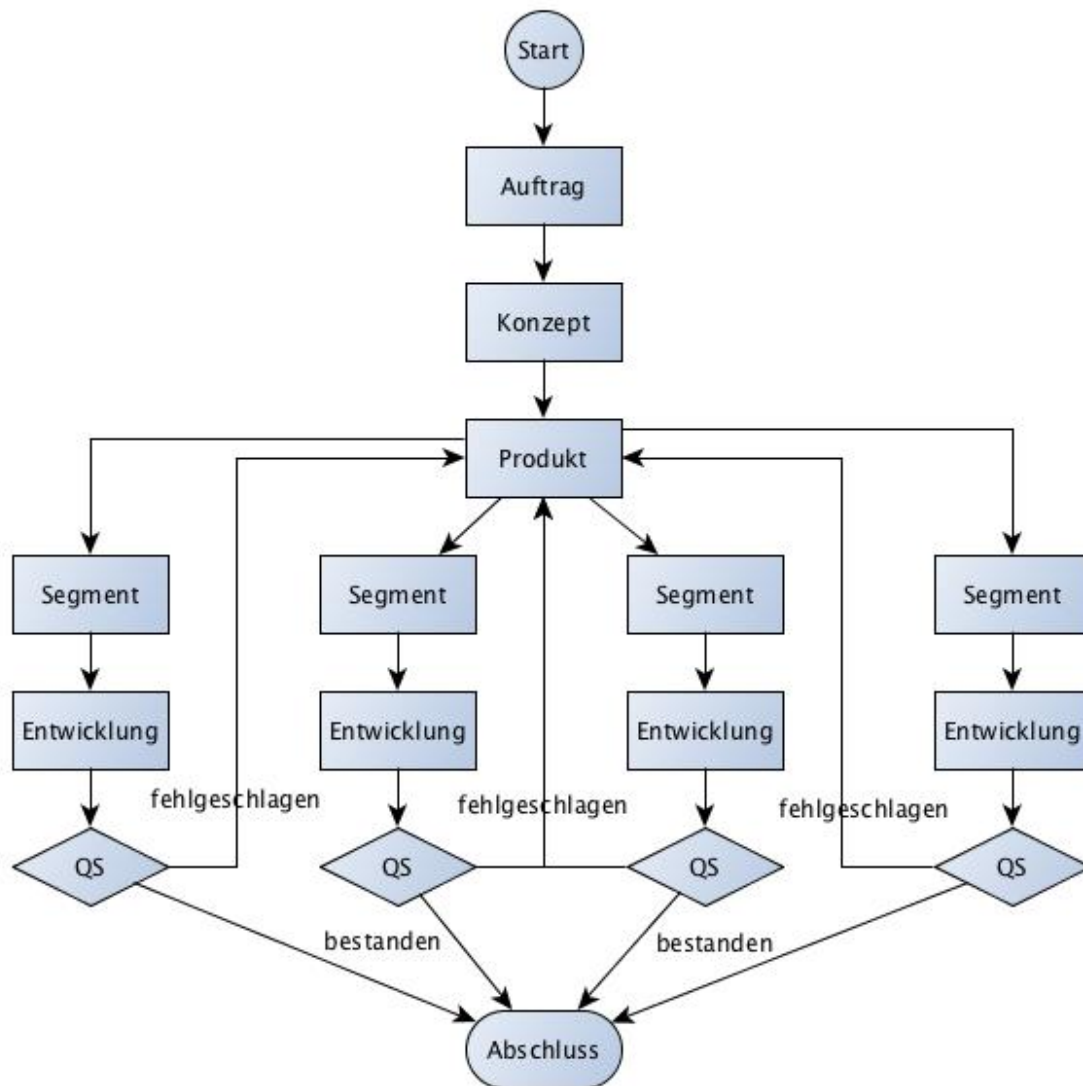


Abbildung 1.1: Vereinfachte Darstellung eines Softwareentwicklungsprozesses

Im Kapitel der Aufgabenstellung befasse ich mich ausschließlich mit der Ausformulierung der Aufgabenstellung. Ich ermittle welche Kriterien Notwendig sind für die Durchführung der Evaluation und lege feste welche Wertigkeit die einzelnen Faktoren in Bezug auf die Gesamtbewertung erhalten. Ebenfalls lege ich in diesem Kapitel die Abgrenzungskriterien fest, welche dazu dienen die Bearbeitung der Aufgabe innerhalb eines vordefinierten Rahmens zu halten.

In dem darauf folgenden Kapitel kläre ich alle allgemeinen sowie auch technischen Grundlagen, die Notwendig sind diese Abschlussthesis zu verstehen. Ich werde ausführlich auf verwendete Begriffe eingehen, sowie Begriffe die in dessen Umfeld entstanden sind. Ein weiterer Punkt innerhalb dieses Kapitels ist die Erläuterung technischer Versuchsaufbauten die im Rahmen der Thesis Notwendig waren um eine Evaluation durchzuführen.

Im Kapitel der Technologien werde ich mich kurz mit den einzelnen Frameworks befassen. Ich erläutere dessen Herkunft, womit sie werben und auf welchen Technologien sie aufbauen. Desweiteren behandle ich in diesem Abschnitt Technologien die einzelne Funktionelle Komponenten sind, welche ich in Hinsicht auf die Entwicklung eines eigenen Frameworks zur parallel-synchronen Steuerung von Webapplikationen auf mobilen Endgeräten auf einen Mehrwert untersuchen werde.

Das Kapitel der Evaluation der Techniken umfasst die Auswertung der erlangten Ergebnisse. Hier werde ich die Resultate meiner Versuchsreihen erläutern und wie man die Ergebnisse nutzen kann, eine optimierte Qualitätssicherung von Webapplikationen, mit dem Fokus auf mobilen Endgeräten, vorzunehmen .

Zum Abschluss werde ich meine Thesis noch einmal zusammenfassen und Fragen klären die während der Bearbeitungszeit auftraten. Probleme die entstanden werden hier erörtert.

Anmerkung

Aus Gründen der besseren Lesbarkeit wird für alle Personen und Funktionsbezeichnungen durchgängig das generische Maskulinum angewendet und bezieht in gleicher Weise Frauen und Männer ein.

2 Aufgabenstellung

Die Aufgaben dieser Thesis ist die Evaluierung von Techniken zur parallel-synchronen Steuerung von Webapplikationen auf mobilen Endgeräten, um damit die Produktivität der Qualitätssicherung zu optimieren.

2.1 Problemstellung

Ein Problem in der aktuellen Softwareentwicklung ist die immer mehr wachsende Anzahl an Endgeräten, welche mit verschiedenen Bildschirmauflösungen und eigenen Betriebssystemen in unterschiedlichen Versionen auftreten. Ein Qualitätsprüfer der einen hohen Qualitätsstandard hat investiert daher linear zu der Anzahl der zu testenden Geräte ansteigend Zeit, lediglich um vereinzelte Testszenarien durchzuarbeiten. Solch ein Testszenario kann Navigationsabläufe¹, das ausfüllen und validieren eines Formular oder auch das überprüfen funktionaler² Links sein. Bereits an dieser Stelle ist die zu investierende Zeit, und dies wiederholt, enorm. Wenn der

Qualitätsprüfer innerhalb eines Testszenarios einen schwerwiegenden Fehler bei einem der Geräte entdeckt, muss dieser den Vorgang beenden. Abgebrochen werden muss deshalb, da bei korrigierter Implementierung der Qualitätsprüfer nicht davon ausgehen darf, das bereits kontrollierte Abschnitte immernoch voll funktionsfähig sind, da eventuell neue Fehler in bereits Kontrollierten Segmenten auftreten können. Sollte ein Szenario aufgrund eines Fehler abgebrochen worden sein,

wird dem Entwickler das Problem möglichst konkret geschildert. Dessen Aufgabe ist es nun das Problem zu beheben. Ist dies geschehen startet der Prüfer einen erneuten Durchgang des Szenarios. Ein generelles Problem was hier noch zusätzlich entstehen kann, ist der Umstand, dass sich grade bei nur kleineren fixes³ und immer wieder auftretenden Testszenarioschleifen eine gewisse Routine einschleichen kann, worunter die Qualität des Produkts leidet.

¹ein Nutzerspezifischer Gang durch die Webseite

²aktive Links und deren Aufruf

³Problemlösungen, Codeanpassungen

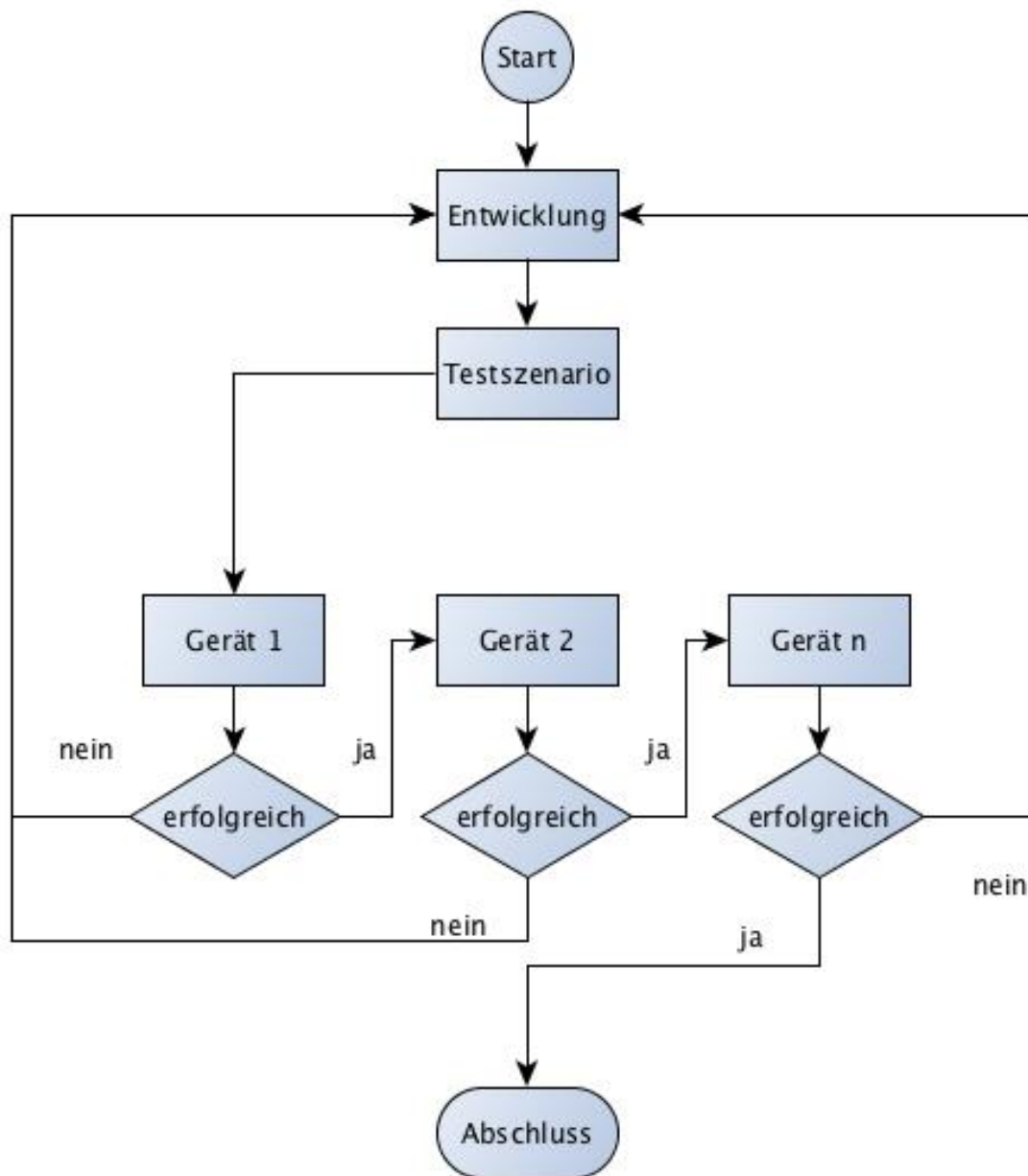


Abbildung 2.1: Darstellung eines Qualitätssicherungsablaufes in der mobilen Anwendungsentwicklung

2.2 Annahmen und Einschränkungen

2.3 Zielsetzung

Das Ziel dieser Arbeit ist es, bestehende Frameworks auf ihre Tauglichkeit in Bezug auf die parallel-synchrone Steuerung von mobilen Endgeräten zur Durchführung von Testszenarien zu evaluieren. Hierzu werden auf mobilen Endgeräten die internen Browser getestet. Hinzu kommen auf Desktopgeräten die aktuellen Versionen

Versionsnummern

<-TOREMOVE von Firefox, Chrome, Safari(nur für Mac-Desktopgeräte) und der Internet Explorer(nur für Windows-Desktops). Um eine Allgemeine Testbarkeit zu gewährleisten werden die Frameworks auch auf Genauigkeit in virtuellen Umgebungen analysiert. Dabei können Abweichungen, seien sie noch so klein, entstehen. Bereits 1 Pixel Abweichung kann bereits ausschlaggebend sein einen Umbruch zu erzeugen und damit das Layout negativ zu verändern.

2.4 Abgrenzungskriterien

- Einarbeitungszeit
- Erweiterbarkeit in Hinsicht auf mehrerer Geräte
- Erweiterbarkeit des verwendeten Frameworks durch eigene Funktionen
- Browsersupport
- Virtuelle Umgebung - ...

3 Grundlagen

In diesem Abschnitt behandle ich spezifische Definitionen wie zum Beispiel verwendetes Fachvokabular, allgemeine technische Abläufe die Notwendig sind um diese Arbeit und die darin verwendeten Techniken zu verstehen, sowie verwendete Hardwarekomponenten.

3.1 Begriffsklärung

3.1.1 parallel-synchron

3.1.2 Web-Applikation

3.1.3 HTML

Die Hypertext Markup Language ist eine Auszeichnungssprache zur Beschreibung von Inhalten. Sie dient der Strukturierung von Texten, Links¹, Listen und Bildern eines Dokumentes. Eine HTML Seite wird von einem Webbrowser interpretiert und anschließend dargestellt. Die Entwicklung von HTML geschieht durch das World Wide Web Consortium(W3C) und den Web Hypertext Application Technology Working Group (WHATWG).

3.1.4 Webbrowser

3.1.5 Desktopcomputer / Desktops

In dieser Arbeit werden gängige Modelle von Personal Computern oder Macs mit einem festen Arbeitsumfeld als Desktops bezeichnet. Hierzu zählen auch tragbare Modelle und Laptops. Im Sinne der Thesis umschließe ich nachfolgend mit dem Begriff Desktop oben genannte Komponenten. Dies dient später der Differenzierung ob es sich um ein mobiles Endgerät handelt oder einem Computer .

3.1.6 Mobiles Endgerät

Im Nachfolgenden werden Komponenten mit primärer mobiler Nutzung umfassend als mobile Endgeräte gruppiert. Hierzu zählen Smarthphones und Tablets.

¹Verweise zu anderen Inhalten

3.1.7 Javascript

3.1.8 Framework

3.1.9 Nodejs

3.1.10 PHP

3.1.11 NPM

3.1.12 Qualitätssicherung

3.1.13 VirtualBox / virtuelle Umgebung

3.1.14 Smartphone

3.1.15 Tablet

3.1.16 Panorama / Portrait View

3.1.17 Pixel

3.1.18 Auflösung

3.1.19 Event

3.1.20 DOM

3.1.21 Apache

3.2 technischer Aufbau

3.3 Komponenten

3.3.1 Raspberry Pi

3.3.2 Hardware

4 Technologien

4.1 Ghostlab

Ghostlab ist ein Framework des Schweizer Unternehmens Vanamco. Es verspricht das synchrone Testen von Websites in Echtzeit. Weiterhin wirbt das Unternehmen mit einem umfangreichen Repertoire an nützlichen Fähigkeiten. Der Funktionsumfang umschliesst das Scrollen innerhalb einer Seite, das ausfüllen von Formularen, das wahrnehmen und reproduzieren von Click-Events sowie dem neuladen einer Seite. Ghostlab soll ebenso einen Inspektor besitzen, welcher die Analyse des DOMs, der on the fly Bearbeitung der CSS und der Analyse und Bearbeitung von Javascriptdateien. Das Framework gibt an für alle folgenden Browser zu funktionieren ohne diese Konfigurieren zu müssen:

| Browser | Version |
|-------------------|-----------|
| Firefox | latest |
| Chrome | latest |
| Safari | latest |
| Internet Explorer | 8/9/10 |
| Opera Mobile | supportet |
| Opera | 11 |
| FireFox Mobile | supportet |
| Blackberry | supportet |
| Windows Phone | supportet |
| Safari mobile | supportet |
| Android | 2.3 - 4.2 |

Tabelle 4.1: von Ghostlab getestete Browser (stand 10.03.2014, Version 1.2.3)

Der Kostenpunkt der Lizenz liegt zur Erstellung dieser Arbeit bei 49\$ (entspricht 35,30€ beim aktuellen Umrechnungswert).

4.2 NodeJS

4.3 Zombie.js

4.4 W3C Touch Events Extensions

4.5 Phantom Limb

4.6 jQuery UI Touch Punch

4.7 jQuery Touchit

4.8 NPM touchit

4.9 Adobe Edge Inspect

4.10 Remote Preview

4.11 Browser-Sync

4.12 Eigenes Framework

5 Evaluation der Techniken

5.1 Ghostlab

5.1.1 Einrichtung der Testumgebung

Ghostlab kommt von Hause aus mit einer 7-Tage-Testversion. Die Installation verlief einfach und ereignislos. Nachdem das Tool installiert wurde erfolgte die Zuweisung einer Website zu dem Ghostlabserver. Ich habe in diesem Fall sowohl eine Seite auf einem lokalen Apache Server getestet, als auch die Demoseite von Ghostlab. version 1.2.3 noch kein Master/Slave Modus

| Browser | Punkte | Wertigkeit |
|-------------------|--------|------------|
| Installation | 1 | ? |
| Konfiguration | 2 | ? |
| Funktion: Desktop | 3 | ? |
| Funktion: Mobil | 4 | ? |
| Erweiterbarkeit | 5 | ? |
| Komplettl sung | 6 | ? |
| Bedienung | 7 | ? |
| ????? | 8 | ? |

Tabelle 5.1: test
TOREMOVE->
evaluationsschlussel
<-TOREMOVE

5.2 Adobe Edge Inspect

5.3 Remote Preview

5.4 Browser-Sync

5.5 Eigenes Framework

5.5.1 Systementwurf

Ablaufdiagramm

Klassendiagramm

Entwurf

6 Ausblick

Entwurf

7 Helpers

7.1 quote

Dies ist ein Zitat.

7.2 longquote

Dies ist ein längeres Zitat.

7.3 fussnote

Dies ist der Text¹

¹Und dies ist die Fußnote dazu.

Index

Abgrenzungskriterien, 4
Ablaufdiagram, 11
Abschlussthesis, 4
Adobe Edge Inspect, 10, 11

Browser-Sync, 10, 11

Evaluation, 4

Framework, 10, 11
Frameworks, 2, 4, 7

Ghostlab, 9–11

jQuery, 9

Kaskadierungsfehler, 2
Klassendiagramm, 11

NodeJS, 9
NPM, 9

Phantom Limb, 9

Raspberry Pi, 8
Remote Preview, 10, 11

Softwareframeworks, 2
Systementwurf, 11

Testunits, 2
Touchit, 9
touchit, 9

UI Touch Punch, 9
Usecases, 2

W3C Touch Events Extensions, 9

Zombie.js, 9
