

Analyse von Mausdaten
Bachelorarbeit

10. Mai 2015

Lars Graubner
Fachhochschule Lübeck
Fachbereich Elektrotechnik und Informatik
Mönkhofer Weg 239
23562 Lübeck
Deutschland

Erklärung zur Bachelorarbeit

Ich versichere, dass ich die Arbeit selbstständig und ohne fremde Hilfe verfasst habe. Bei der Abfassung der Arbeit sind nur die angegebenen Quellen benutzt worden. Wörtlich oder dem Sinne nach entnommene Stellen sind als solche gekennzeichnet.

Diese Arbeit unterliegt keinem Sperrvermerk und darf somit veröffentlicht und Dritten zur Einsicht gegeben werden.

Lübeck, 10. Mai 2015

Unterschrift

Inhaltsverzeichnis

1	Einleitung	8
1.1	Aktuelle Situation	8
1.2	Reflexion der Aufgabenstellung	8
1.3	Rechtliches	9
1.4	Aufbau der Arbeit	9
2	Anforderungsanalyse	10
2.1	Analyse Produktfeatures	10
2.2	Analyse der Softwareanforderungen	11
2.3	Userinterface	14
3	Systemarchitektur	17
3.1	Server	18
3.1.1	Apache	18
3.1.2	nginx	18
3.1.3	Node.js	18
3.1.4	Fazit	19
3.2	Portal	19
3.2.1	Tracking-Script	19
3.2.2	Visualisierungsmethoden	20
3.3	Datenbank	20
3.4	Skalierung des Systems	21
3.4.1	Parallelisierung	21
3.4.2	Horizontale Skalierung	21
3.4.3	Zuordnung von Anforderungen und Systemarchitektur-Komponenten	22
4	Softwarearchitektur und Softwarekomponenten-Design	23
4.1	Portal	23
4.1.1	Server	23
4.1.2	Client	24

<i>INHALTSVERZEICHNIS</i>	3
4.2 Tracking-Script	25
4.2.1 Übermittlung der Daten	26
4.3 Datenbank	26
5 Beschreibung der Implementierung	28
5.1 Portal	28
5.1.1 Server	28
5.1.2 Client	33
5.2 Tracking-Script	38
5.3 Skalierung	38
5.3.1 Portal	38
5.3.2 Datenbank	40
6 Test und Evaluation	41
6.1 Unittests	42
6.1.1 Server	43
6.1.2 Client	44
6.2 Integrations- und Systemtests	45
6.3 Nachweis der Anforderungen	46
7 Zusammenfassung und Ausblick	49
7.1 Zusammenfassung	49
7.2 Ausblick	49
7.2.1 Authentifizierung und Rechtesystem	49
7.2.2 Weitere Analyse der Daten	50
7.2.3 Bessere Aufschlüsselung der Endgeräte	50
A Testprotokolle	51
Literaturverzeichnis	58

Abbildungsverzeichnis

2.1	Anwendungsfall Besucher	13
2.2	Anwendungsfall Analyst	13
2.3	Wireframe Portal Einstiegsseite	15
2.4	Wireframe Portal Seite Detail	15
3.1	Abbildung der Systemarchitektur	17
3.2	Beispiel einer Heatmap	20
3.3	Skalierung der Systemarchitektur	22
4.1	Softwarekomponente Portal	23
4.2	Komponenten Diagramm	24
4.3	Softwarekomponente Tracking-Script	25
4.4	Softwarekomponente Datenbank	27
5.1	Material Design: Ebenen im Raum	36
5.2	Einstiegsseite Portal	36
5.3	Aufbau des Portals	37
5.4	Visualisierungen im Portal	38
6.1	Veranschaulichung der Teststruktur	42

Tabellenverzeichnis

2.1	Erfasste Daten beim Tracking	12
2.2	Zuordnung der Softwareanforderungen und Produktfeatures	14
3.1	Vergleich Webserver-Software	19
3.2	Zuordnung von Anforderungen und Systemarchitektur	22
5.1	Verwendete Routen des Portals	31
6.1	Auflistung der Unittests	43
6.2	Auflistung der Systemtests	46
6.3	Ergebnisse der durchgeführten Tests	47
6.4	Nachweis der Anforderungen	48

Programm-Listings

4.1	Revealing Module Pattern	24
4.2	Verwendung Revealing Module Pattern	25
4.3	Script-Tag mit async Parameter	25
4.4	Script zum asynchronen Laden von Javascript	25
5.1	Event-Model	28
5.2	Speicherung eines Event-Objektes	29
5.3	Teil eines Jade-Templates	30
5.4	Kompiliertes Jade-Template	30
5.5	Controller für die Index-Route	31
5.6	Berechnung der Positionsdaten	32
5.7	Portal Modul Heatmap	33
5.8	Erstellen eines Webbugs	38
5.9	Aktivierung des Clusters in der Konfiguration	38
5.10	Starten von Master- und Worker-Prozessen	39
5.11	nginx Beispiel-Konfiguration	39
5.12	Starten eines MongoDB Clusters	40
5.13	Hinzufügen von MongoDB Shards	40
6.1	Starten der serverseitigen Unittests	43
6.2	Unittest auf Serverseite	44
6.3	Unittest auf Clientseite	44
6.4	Systemtest für das Portal	45

Danksagung

Ich bedanke mich bei Herrn Prof. Dr. Nane Kratzke für die Geduld und hervorragende Betreuung, sowie Frau Prof. Dr. Janneck für die Zweitkorrektur. Zudem danke ich Marcel Birkhahn und Fabian Will für das Korrekturlesen dieser Arbeit.

Kapitel 1

Einleitung

Webseiten werden für Besucher gestaltet und entwickelt. Um dem gerecht zu werden, muss die Handlungsweise der Besucher verstanden werden. Dafür benötigt es einer Software, welche das Verhalten aufzeichnet und anschließend zur Analyse bereit stellt. Genau damit soll sich diese Arbeit beschäftigen. Die Analyse soll dabei möglichst einfach und intuitiv möglich sein, damit durch entsprechende Optimierungen an der Seite mehr Conversions¹ erreicht werden können.

1.1 Aktuelle Situation

Auf dem Markt gibt es sehr viele verschiedene Tracking-Lösungen zur Analyse von Webseitennutzung. Bekannte Vertreter sind beispielsweise Google Analytics² oder Piwik³. Bei Produkten dieser Art wird das Nutzerverhalten in Hinsicht auf die komplette Webseite untersucht. Allerdings bleibt hier das Verhalten des Besuchers auf einzelnen Seiten größtenteils verborgen. Was hat sich der Benutzer während der Zeit auf der Seite alles angeguckt? Womit hat er wie viel Zeit verbracht? Hat er den entscheidenden Abschnitt der Seite zum Kaufen/Buchen/Abonnieren gesehen? Diese und weitere Fragen können oftmals nicht beantwortet werden.

Für diese Art von Tracking gibt es bereits einige Lösungen, wie zum Beispiel Crazy Egg⁴ oder Inspectlet⁵, welche sich ebenfalls mit diesem Problem beschäftigen. In dieser Arbeit soll die Entwicklung einer solchen Anwendung mit aktuellen Techniken nachvollzogen und eine einfach skalier- und erweiterbare Lösung gefunden werden.

1.2 Reflexion der Aufgabenstellung

Die Bedeutung des Internets hat sich in den letzten zwei Jahrzehnten wesentlich verändert. Zunächst ausschließlich für wissenschaftliche Zwecke gedacht, wurde dieses nach und nach auch für Privatleute interessant. Informationen lassen sich so auf einfachste Weise einer großen Menge von Menschen zur Verfügung stellen. Allmählich wurde das Potential zur kommerziellen Nutzung des Internets entdeckt, was in den Onlineshops wie wir sie heute kennen, gipfelt. Um die Umsätze, oder generell Besucherzahlen auf einer Seite zu erhöhen, werden diese intensiv für das optimale Nutzungserlebnis optimiert. Um dies zu ermöglichen, benötigt es einer Software, welche die Ist-Situation und die Entwicklung der Besucherzahlen aufzeichnet.

¹Eine Conversion bezeichnet die Umwandlung eines Interessenten in einen Kunden

²<http://www.google.com/intl/de/analytics/>

³<http://piwik.org/>

⁴<https://www.crazyegg.com>

⁵<http://www.inspectlet.com>

Diese Arbeit beschäftigt sich mit der Aufzeichnung von Mausdaten zur Analyse von Webseiten. Dabei soll es nicht um die Bewegung des Besuchers zwischen den verschiedenen Unterseiten gehen, sondern lediglich um das Verhalten auf den einzelnen Seiten. Mit dem gewonnenen Wissen sollen Optimierungsmöglichkeiten einfacher interpretiert und basierend darauf umgesetzt werden können. Es soll möglich sein, das Verhalten des Besuchers zu verstehen.

Als Ergebnis dieser Arbeit soll eine Anwendung entwickelt werden, welche die Mausdaten aufzeichnen, speichern und als Visualisierungen anzeigen lassen kann. Der Fokus liegt dabei auf der Verwendung von aktuellen Techniken, Erweiterbarkeit und Skalierbarkeit. Um den Rahmen dieser Arbeit nicht zu sprengen, sind die Funktionen der Anwendung auf das wesentliche beschränkt und Komponenten wie Nutzerauthentifizierung etc. außen vor gelassen.

1.3 Rechtliches

Die Erhebung und Speicherung von Daten eines Besuchers einer Webseite ist nur rechtmäßig, wenn die Einwilligung des Besuchers vorliegt (§ 15 Abs. 1 TMG). In diesem Fall ist es fraglich, inwiefern es sich um personenbezogene Daten handelt. Dies wäre zum Beispiel bei der Speicherung der IP-Adresse der Fall. Es ist nicht Gegenstand dieser Bachelorarbeit, die Rechtmäßigkeit von personenbezogenen Daten, die durch Tracking gewonnen wurden, zu erörtern. An dieser Stelle der Hinweis, dass die Verwendung dieser Software im Produktiveinsatz unter rechtlichen Aspekten näher betrachtet werden muss. [Sch11]

1.4 Aufbau der Arbeit

In dieser Arbeit werden zunächst in Kapitel 2 die Anforderungen an die Anwendung analysiert, welche sich aus den Anwendungsfällen ergeben. Die Anforderungen werden als einzelne, konkrete Punkte formuliert, damit diese bei der Entwicklung der Anwendung als Referenz zur Verfügung stehen. Anhand dieser Punkte wird die Anwendung entwickelt und evaluiert. Darauf folgt in Kapitel 3 die Systemarchitektur. An dieser Stelle wird erörtert, wie das System, auf welchem die Anwendung läuft, aufgebaut ist und mögliche Optionen werden diskutiert. Im darauf folgenden Kapitel 4, wird die Struktur der Anwendung näher untersucht und verwendete Prinzipien und Techniken erläutert. Die konkrete Umsetzung der Anwendung wird in Kapitel 5 besprochen. Hier wird an repräsentativen Beispielen die Umsetzung der Anwendung verdeutlicht. Schlussendlich wird die Anwendung in Kapitel 6 anhand der Anforderungen evaluiert und die verwendeten Methoden zum Testen vorgestellt. Das letzte Kapitel enthält eine Zusammenfassung der Arbeit und den Ausblick auf Erweiterungsmöglichkeiten der Anwendung.

Kapitel 2

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an die Anwendung und die zu implementierenden Produktfeatures betrachtet. Diese ergeben sich aus der Ist-Situation und den Anwendungsfällen für die Software. Auf Basis dieser Anforderungen werden die Implementierungsentscheidungen getroffen. Zudem wird die Struktur der GUI¹ des Portals in Form von Wireframes skizziert.

2.1 Analyse Produktfeatures

Die zu entwickelnde Webanwendung soll als so genannte Software as a Service (SaaS) funktionieren. Das heißt, beliebig viele Endpunkte greifen auf ein zentrales System zu und nutzen dieses zu einem bestimmten Zweck. [Gmb15] Der Benutzer muss sich hierbei keine Gedanken über Installation, Wartung und Skalierung des Systems machen. Um die Software zu nutzen wird lediglich ein kleines Script auf der zu analysierenden Seite eingebunden.

Ist das Tracking-Script eingebunden, werden alle relevanten Mausaktionen der Besucher der Webseite registriert. Da eine Webseite von unzähligen verschiedenen Browsern aufgerufen werden kann, muss das Tracking-Script mit möglichst allen marktrelevanten Browsern kompatibel sein. Nur so kann eine repräsentative Menge an Daten gewährleistet werden.

Die aufgezeichneten Daten werden an einen Server gesendet, damit diese dauerhaft in einer Datenbank gespeichert werden können. Das Senden sollte parallel zur Benutzung der Seite geschehen, um möglichst alle Daten zu erfassen und Datenverlust durch unerwartetes Schließen der Seite zu verhindern. Dabei darf der Besucher nicht durch das Script beeinflusst werden. Das Nutzungserlebnis muss ungestört bleiben, um die Authentizität der Daten zu erhalten. Im Idealfall merkt der Besucher nichts von der Erhebung und Versendung der Daten. Zur rechtlichen Lage über die Speicherung von Nutzerdaten mehr im Abschnitt 1.3.

Der zentrale Server nimmt die gesammelten Daten aller Seiten entgegen und bereitet diese auf, um diese in einer Datenbank zu speichern. Damit der Analyst die aufgezeichneten Daten analysieren kann, läuft auf dem Server ein Portal. Das Portal bezieht über eine Schnittstelle die Daten aus der Datenbank und visualisiert diese. Der Benutzer hat dabei zu keiner Zeit die Möglichkeit, die Daten zu verändern. Diese können ausschließlich ausgelesen und gegebenenfalls gefiltert werden.

Das ganze System sollte zudem einfach zu Skalieren sein, da die Menge der Daten bei vielen Endpunkten schnell sehr groß werden kann und die Performance dadurch leidet. Dabei soll zum einen Parallelisierung, als auch horizontale Skalierung berücksichtigt werden. Parallelisierung bedeutet dabei, dass die Hardware-Ressourcen eines Servers erhöht werden, wohingegen horizontale Skalierung das Hinzunehmen weiterer Server als Cluster² bezeichnet.

¹Graphical User Interface

²Zusammenschluss von mehreren Computern [tec]

Daraus ergeben sich folgende Grundfunktionen für die Anwendung:

- PF1** Registrierung von Benutzeraktionen
- PF2** Senden von ermittelten Daten an Server
- PF3** Aufbereitung und Speicherung der Daten
- PF4** Bereitstellung und Filterung der Daten
- PF5** Visualisierung der Daten
- PF6** Einfache Skalierung des Systems

Die in **PF6** genannte Skalierung des Systems kann im Kontext dieser Arbeit nur in begrenztem Umfang umgesetzt und getestet werden. Die Entwicklungsumgebung bietet nicht die Ressourcen für eine Skalierung. Allerdings wird die Software in Hinblick auf diese entwickelt und entsprechende Vorkehrungen werden getroffen.

2.2 Analyse der Softwareanforderungen

Die Benutzung der Software soll durch folgenden Anwendungsfall beschrieben und daraus die Anforderungen an die Webanwendung abgeleitet werden.

Herr Meier hat eine Webseite auf der er seine handgefertigten Koffer in einem kleinen Online-Shop verkauft. Er hat im Fernsehen einen TV-Spot schalten lassen, welcher auf ein besonderes Angebot hinweist. Dafür hat er eine spezielle Landingpage erstellt, welche das Produkt und Angebot beschreibt und zum Kaufen animieren soll. Durch Google Analytics weiß er, dass er durchaus einige neue Besucher zu verzeichnen hat, jedoch wird nur selten etwas gekauft. Um zu erfahren warum dies der Fall ist, bindet er auf seiner Seite das Tracking-Script zur Mausdaten Analyse ein. Nach einiger Zeit geht er auf das Portal, in dem die Daten der Besucher seiner Seite übersichtlich und für ihn verständlich dargestellt werden. Er kann erkennen, was die Besucher gesehen und worauf sie geklickt haben. Durch die Interpretation dieser Daten passt er seine Seite an, um den Benutzer zu mehr Conversions zu bewegen.

Zwei Aspekte sind für den Analysten interessant: Was hat der Besucher gesehen und was hat er gemacht. Studien belegen, dass die Mausposition mit dem Sichtfeld des Besuchers korreliert. [Che01] Das heißt, Elemente welche in der Nähe der Mausposition liegen, wurden sehr wahrscheinlich auch gesehen. Daher wird die Mausposition zur Analyse erfasst. Weiterhin interessant für den Analysten ist, ob der Besucher mit den gesehenen Elementen interagiert hat (vorausgesetzt dies ist vorgesehen). Alle Interaktionen auf einer Webseite werden durch einen Mausklick ausgeführt, bzw. eingeleitet. Ausnahmen sind Screenreader und Bedienung ausschließlich durch eine Tastatur, allerdings ist dies nicht die Regel und wird daher in diesem Zusammenhang nicht behandelt.

Die Mauspositionen und Mausklicks müssen zeitlich und räumlich eingeordnet werden. Die Position ist ein Punkt, an dem die Maus eine bestimmte Zeit ruht, also ein Endpunkt einer Bewegung. Der Weg an sich ist nicht relevant, da die Mausbewegungen meist zielgesteuert sind. Für die zeitliche Erfassung wird ein Zeitstempel von dem Moment der Erhebung für Klick und Position gespeichert. Zur räumlichen Einordnung werden zwei Dinge erfasst: Zum einen die Position des Mauszeigers relativ zum anvisierten Element und zum anderen eine eindeutige Kennung für das Element. Für die eindeutige Kennung wird die XML Path Language³ (XPath) verwendet. Damit lassen sich Elemente in einem DOM-Tree⁴ eindeutig beschreiben und wieder auflösen. Dies ist notwendig, da es verschiedene Endgeräte mit

³<http://www.w3.org/TR/xpath20/>

⁴Document Object Model - Spezifikation einer Schnittstelle für den Zugriff auf HTML-Dokumente [Wik15]

unterschiedlichen Auflösungen gibt. Außerdem können durch responsive Webdesign und mobil-optimierte Webseiten unterschiedliche Ansichten der Seite existieren. Für eine korrekte Zuordnung der Daten muss außerdem die URL der Seite mit dem eingebundenem Tracking-Script gespeichert werden.

Um die Daten differenzierter analysieren zu können, werden weitere Daten erhoben. Besucher werden durch unterschiedliche Seiten auf die Zielseite geleitet und haben dadurch womöglich andere Intentionen. Dies kann durch die Speicherung des Referrers überprüft werden. Mit dem Browser Useragent und der Auflösung des Besuchers kann ermittelt werden, mit was für einer Art von Endgerät die Seite besucht wurde. Außerdem kann noch untersucht werden, wie viele Interaktionen durch einen Besucher verursacht wurden. Dafür wird eine eindeutige Session ID erzeugt, welche allen Daten aus einer Session angehängt wird. Da die ermittelten Eigenschaften von Mausposition und Mausklick sich überschneiden, wird beides als ein Mausevent gesehen. Die Unterscheidung erfolgt durch einen Parameter Typ. Die folgende Tabelle zeigt die erfassten Daten bei einem Mausevent.

Eigenschaft	Beschreibung
Zeitstempel	Zeitstempel vom Erfassungszeitpunkt
Koordinaten	Koordinaten des Mauszeigers relativ zu anvisiertem Element
Session ID	einmalig pro Besuch vergebene ID
Useragent	Useragent des Nutzers
Auflösung	Auflösung des Besucher-Browsers
Referrer	Referrer zur getrackten Seite
Seiten URL	URL der getrackten Seite
Event Art	Unterscheidung zwischen Klick/Position
Element	Informationen über das anvisierte Element
Typ	Die Art des Mausevents

Tabelle 2.1: Erfasste Daten beim Tracking

Genau genommen gibt es zwei Arten von Anwendungsfällen, welche durch die folgenden Anwendungsfall-Diagramme genauer dargestellt werden.

Im ersten Fall besucht eine Person eine Webseite mit eingebundenem Tracking-Script. Sie agiert mit der Webseite und alle Mauspositionen und -klicks werden vom Script registriert und an das Portal gesendet. Der Besucher interagiert somit nur passiv mit der Webanwendung, da sein Interesse der besuchten Seite gilt.

Das folgende Anwendungsfall-Diagramm beschreibt die Interaktion eines Besuchers.

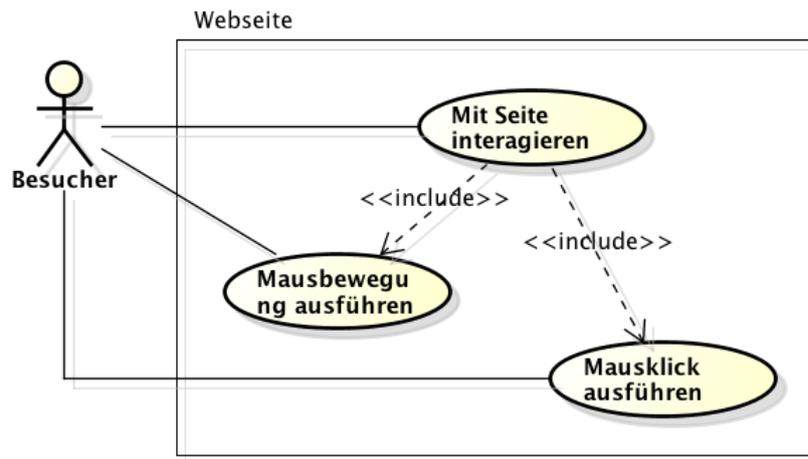


Abbildung 2.1: Anwendungsfall Besucher

Beim zweiten Anwendungsfall ruft der Analyst das Portal auf, um die aufgezeichneten Daten der Webseite anzusehen. Im ersten Schritt muss die gewünschte Seite ausgewählt werden, da unter Umständen Daten mehrerer Unterseiten einer Webseite vorhanden sind. Ist eine Seite gewählt, können die generierten Visualisierungen justiert werden. Die Daten können nach Referrer, URL, Endgerät und nach Datum gefiltert werden. Dadurch soll eine möglichst differenzierte Analyse ermöglicht werden.

Dies wird im folgenden Anwendungsfall-Diagramm verdeutlicht.

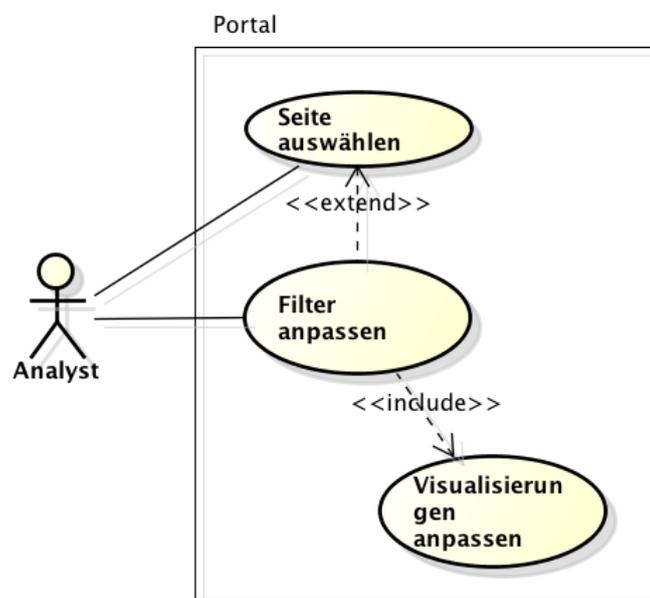


Abbildung 2.2: Anwendungsfall Analyst

Aus diesen Anwendungsfällen ergeben sich folgende Softwareanforderungen. Die ermittelten Daten werden in der Tabelle 2.1 genauer aufgeschlüsselt.

- **[Req1]** Mausclicks werden erkannt

- **[Req2]** Mauspositionen werden erkannt
- **[Req3]** Ermittelte Daten werden an Server gesendet
- **[Req4]** Ermittelte Daten werden in Datenbank gespeichert
- **[Req5]** Ermittelte Daten können vom Server abgefragt werden
- **[Req6]** Mausevents können nach Referrer gefiltert werden
- **[Req7]** Mausevents können nach Endgerät gefiltert werden
- **[Req8]** Mausevents können nach URL gefiltert werden
- **[Req9]** Mausevents können nach Datum gefiltert werden
- **[Req10]** Mausevents können nach Typ gefiltert werden
- **[Req11]** Mausevents können dargestellt werden
- **[Req12]** Seite kann ausgewählt werden

Diese werden den Produktfeatures aus dem Kapitel 2.1 zugeordnet.

Anforderung x Feature	PF1	PF2	PF3	PF4	PF5	PF6
Req1 (Mausklicks erkennen)	x					
Req2 (Mauspositionen erkennen)	x					
Req3 (Ermittelte Daten an Server senden)		x				
Req4 (Ermittelte Daten in Datenbank speichern)			x			
Req5 (Ermittelte Daten von Server abfragen)				x		
Req6 (Mausevents nach Referrer filtern)				x		
Req7 (Mausevents nach Endgerät filtern)				x		
Req8 (Mausevents nach URL filtern)				x		
Req9 (Mausevents nach Datum filtern)				x		
Req10 (Mausevents nach Typ filtern)				x		
Req11 (Mausevents darstellen)				x	x	
Req12 (Seite auswählen)				x		

Tabelle 2.2: Zuordnung der Softwareanforderungen und Produktfeatures

Dem Produktfeature 6 sind in der Tabelle keine Anforderungen zugeordnet, da dies das komplette System betrifft. Somit lassen sich einzelne Anforderungen nicht konkret der Skalierung zuordnen.

2.3 Userinterface

Das Userinterface soll simpel und intuitiv gestaltet werden und auf komplexe Menüs, Einstellungen und Seitenstrukturen verzichten. Die Einstiegsseite enthält ausschließlich eine Auswahl an Seiten, für welche aufgezeichnete Daten vorliegen. Durch Klick auf die Seiten wird man auf die nächste Seite zur Analyse der Daten weitergeleitet.

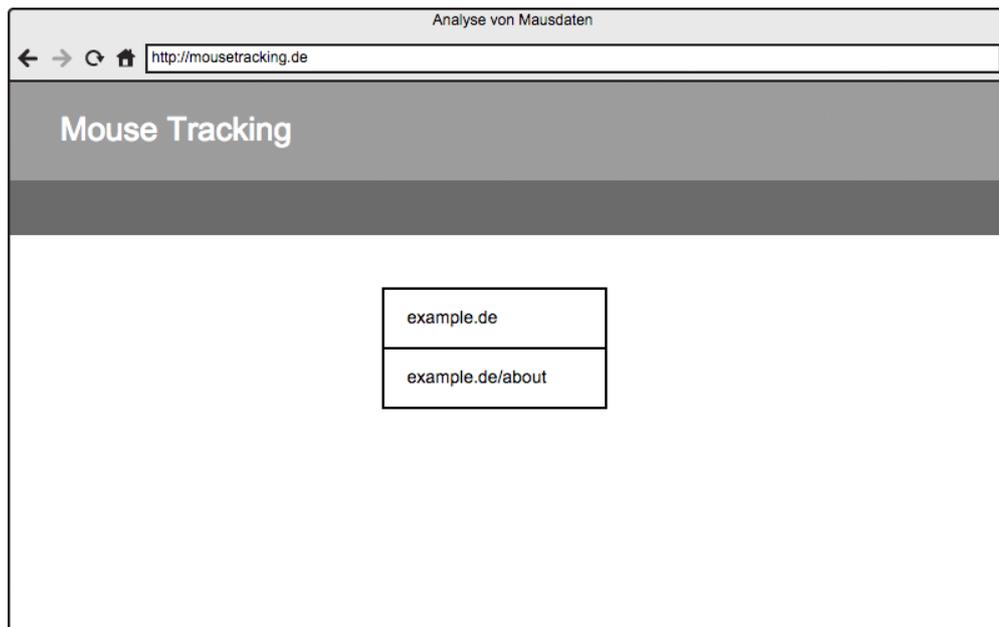


Abbildung 2.3: Wireframe Portal Einstiegsseite

Die Analyseseite besteht aus einem Header, in welchem der Titel der Anwendung steht, sowie einer Reihe von Filtern. Mit den Filtern lassen sich die Visualisierungen anpassen. So hat man direkt die ausgewählten Einstellungen vor Auge und weiß, was man sich gerade ansieht. Darauf folgen im Inhaltsbereich die Visualisierungen.

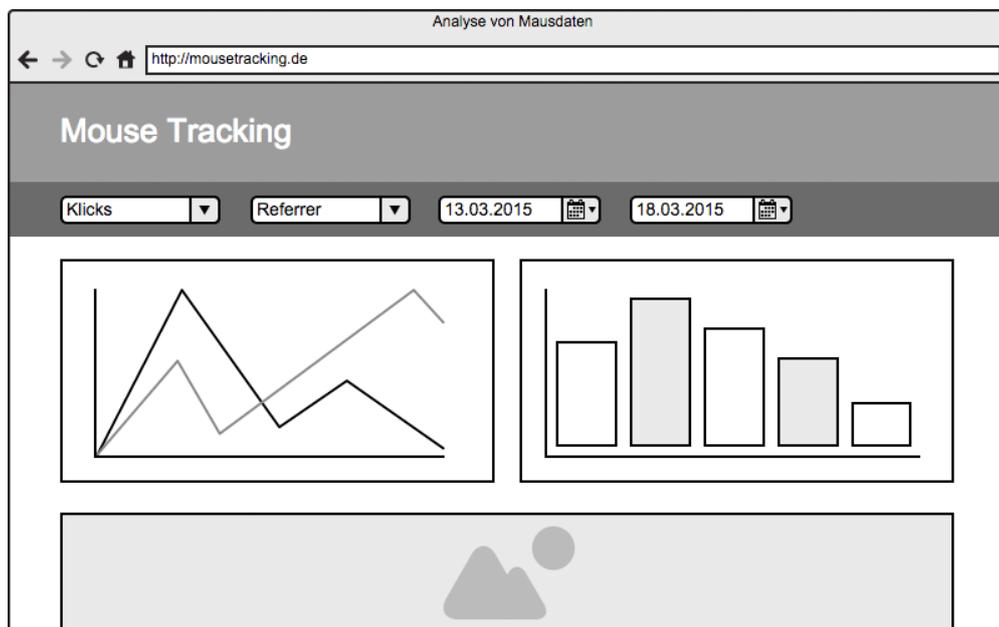


Abbildung 2.4: Wireframe Portal Seite Detail

Die Umsetzung des Userinterfaces soll sich am Material Design⁵ von Google orientieren. Dies ist eine Designsprache, welche gewisse Regeln und Konventionen vorgibt. Alle Elemente auf einer Seite haben dabei neben einer horizontalen und vertikalen Position auch eine bestimmte Position in der Tiefe. Jedes Element ist also ein Objekt in einem dreidimensionalen Raum. Dies wird mit Licht und Schatten in

⁵<http://www.google.com/design/spec/material-design/introduction.html>

Abhängigkeit von der Entfernung zum Benutzer verdeutlicht. Durch diese Einordnung im Raum soll die Orientierung für den Benutzer verbessert und Bewegungen und Interaktionen verdeutlicht werden. Diese und weitere Richtlinien sollen ein einheitliches Bild und ein konsistentes Nutzungserlebnis gewährleisten.

Kapitel 3

Systemarchitektur

Im folgenden Kapitel wird die grundlegende Systemarchitektur der Webanwendung mit ihren einzelnen Komponenten erörtert. Das System besteht aus einem zentralen Knoten, der alle ermittelten Daten der Endpunkte entgegennimmt. Die Endpunkte sind in diesem Fall verschiedene Webseiten, welche das Tracking-Script eingebunden hat. Das Portal kommuniziert mit einer Datenbank zur Speicherung der gesendeten Daten und dient somit als Schnittstelle für den Analysten. Die Datenbank kann gegebenenfalls auch auf einem externen oder auch mehreren externen Servern laufen. Diese Skalierung kann bei einem hohen Traffic nützlich sein, um eine gute Performance gewährleisten zu können.

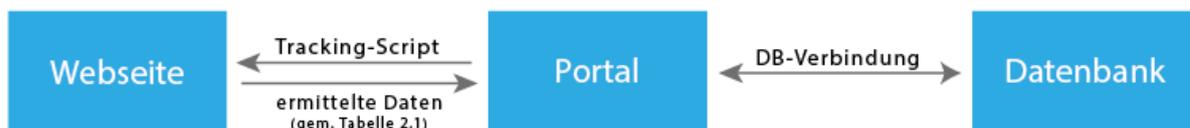


Abbildung 3.1: Abbildung der Systemarchitektur

Der Server, auf dem das Portal läuft, und die Datenbank werden als Docker Container bereitgestellt. Mit Docker¹ lassen sich Betriebssystemvirtualisierungen als leichtgewichtige Container erstellen, welche ausschließlich die benötigten Skripte und Pakete enthalten. Dabei sind diese vollkommen von dem Host-System getrennt. Dies ermöglicht eine einfache Bereitstellung auf jedem System, welches die Docker-Software unterstützt. Daraus ergeben sich erhebliche Vorteile für die Skalierung von Anwendungen. Es können beispielsweise weitere Server zur Verfügung gestellt werden auf denen die selben Container gestartet werden (vorausgesetzt die Software ist für die Nutzung im Cluster optimiert).

Für diese Bachelorarbeit gibt es einen weiteren Docker Container, in welchem ein Testclient läuft. Dieser besteht aus einem Docker Container mit Node.js auf welchem das freie Template Transit² läuft. Dieses besteht aus einfachen HTML-Dateien in welche das Tracking-Script eingebunden ist. Damit lässt sich die Datenerhebung der Anwendung testen.

In den nächsten Abschnitten werden die genannten Komponenten und deren Verbindung untereinander genauer beschrieben.

¹<https://www.docker.com>

²<http://templated.co/transit>

3.1 Server

Es wird eine Webserver-Software benötigt, auf der das Portal läuft. Die bekanntesten Vertreter sind Apache³, nginx⁴ und IIS⁵ von Microsoft, sowie der Newcomer NodeJs⁶. IIS ist ein sehr stabiler Webserver und unterstützt die meisten gängigen Scriptsprachen wie zum Beispiel PHP und Microsoft eigene Sprachen (z.B. ASP). Allerdings wird dafür ein Windows Server benötigt. Da dies ein kommerzielles Produkt, und zu umfangreich für eine derartige Software ist, scheidet ein IIS Server aus. Stattdessen sollte der Server auf einem UNIX-System laufen können. Zur Auswahl stehen verschiedene Linux Distributionen wie zum Beispiel Debian, Ubuntu oder Fedora, welche Open-Source und relativ leichtgewichtig sind. Die Wahl der Distribution ist für diese Anwendung eher unwesentlich, da auf allen populären Distributionen die verwendete Software läuft. Für diese Anwendung wird Debian verwendet, da es dafür aktuelle Pakete und eine sehr aktive Community gibt.

3.1.1 Apache

Der Apache Webserver ist ein Open Source HTTP Server, welcher sowohl unter UNIX-Systemen als auch Windows läuft. Es werden verschiedene Script-Sprachen wie PHP und PERL unterstützt. Jede Anfrage wird von einem eigenen Workerprozess bearbeitet (Multithreading). Der Prozess wird dadurch blockiert und beeinflusst dadurch keine anderen Prozesse (Anfragen). Bei jeder Anfrage werden alle installierten Apache Module mit an den Prozess gebunden, was zu einer Verschwendung von Ressourcen führen kann. Bei der Anfrage für eine Bilddatei wird zum Beispiel PHP im Prozess eingebunden, auch wenn dies gar nicht benötigt wird. Bei jeder Anfrage wird ein neuer Prozess erstellt, sodass auch eine Skalierung des Systems nur eingeschränkt eine bessere Performance leisten kann. Dadurch wird bei sehr vielen Anfragen die Performance eines Apache sehr schnell beeinträchtigt.

3.1.2 nginx

Durch eine modulare Struktur ist der nginx schlanker als Apache und soll unter hoher Last eine bessere Performance aufweisen. [Led12] Dies wird unter anderem dadurch erreicht, dass die Bearbeitung von Anfragen ereignisgesteuert ist. Jede Anfrage wird dabei als Ereignis gesehen und einem Prozess zugeordnet. Ein Prozess kann dabei mehr als eine Anfrage bearbeiten. Dadurch kann bei vielen Anfragen die Warteschlange schnell und performant abgearbeitet werden. Die Stärken von nginx liegen beim Load Balancing⁷ und Ausliefern von statischen Inhalten, nicht jedoch beim Ausliefern von dynamischen Inhalten. Daher ist nginx für diesen Anwendungsfall nur begrenzt geeignet.

3.1.3 Node.js

Ein noch sehr junger Vertreter ist Node.js. Bei diesem Webserver wird durch Nutzung der V8 JavaScript Engine Javascript auf den Server portiert. Alle Anfragen werden durch einen einzelnen Prozess verarbeitet. Dies wird durch eine ereignisgesteuerte, asynchrone Bearbeitung der Anfragen ermöglicht. Dadurch entsteht der Nachteil, dass sich Anfragen an den Server gegenseitig negativ, durch Fehler oder eine zu hohe Auslastung, beeinflussen können, dafür aber den Prozess nicht blockieren. Allerdings hat diese Softwarearchitektur den Vorteil, dass ein Node.js Server sehr viele Anfragen sehr performant bearbeiten kann. Es muss nicht erst ein neuer Prozess für eine Anfrage gestartet werden. Im Gegensatz zu Apache

³<http://httpd.apache.org>

⁴<http://nginx.org>

⁵<http://www.iis.net>

⁶<http://nodejs.org>

⁷Verteilen von Anfragen auf verschiedene Instanzen einer Anwendung [ngi]

und nginx bietet Node.js von Haus aus nur wenig Funktionen. Vieles muss durch weitere Pakete hinzugeinstalliert werden. Dies kann allerdings auch als Vorteil ausgelegt werden, da wirklich nur die benötigten Module geladen werden und dafür viele optionale Erweiterungspakete zur Verfügung stehen. [Hir13]

3.1.4 Fazit

Da die Erhebung und das Senden der Mausdaten sehr viele Anfragen an den Server stellt, ist es wichtig, dass dieser alle Anfragen performant verarbeiten kann. Daher scheint Node.js prädestiniert. Es kann die Abarbeitung der Anfragen für diesen Anwendungsfall am besten bewerkstelligen. Wenn nötig, kann die Infrastruktur des Servers einfach horizontal skaliert werden, um eine größere Anzahl an Anfragen zu verarbeiten. [Zga] Die Architektur ist schlank gehalten und kann beliebig durch weitere Pakete, um viele Funktionen, erweitert werden. Die folgende Tabelle vergleicht essentielle Funktionen und Eigenschaften der besprochenen Webserver.

	Apache	nginx	Node.js
UNIX kompatibel	+	+	+
non-blocking	-	+	+
dynamische Inhalte	+	-	+
Skalierbarkeit	-	+	+
Summe +	2	3	4

Tabelle 3.1: Vergleich Webserver-Software

Der Webserver nginx schneidet anhand der Tabelle ähnlich ab wie Node.js. Verschiedene Benchmark Tests zeigen jedoch, dass Node.js noch um einiges performanter als nginx ist. [Mod] Noch ausschlaggebender ist allerdings die bessere Unterstützung von dynamischen Scriptsprachen. Daher fällt die Wahl auf Node.js.

Die Webserver-Software wird in Form eines Docker Containers mit allen benötigten Paketen bereitgestellt.

3.2 Portal

Das Portal ist die zentrale Anlaufstelle der Anwendung. Es stellt das Tracking-Script zur Verfügung, nimmt erfasste Daten entgegen und stellt die Visualisierungen zur Verfügung. Damit dient das Portal als Schnittstelle für den Analysten zu den Daten in der Datenbank.

3.2.1 Tracking-Script

Das Tracking-Script wird durch das Portal bereitgestellt und auf der zu untersuchenden Seite eingebunden. Das Script wird initialisiert und registriert alle relevanten Mausevents. Die Daten werden an das Portal gesendet, welches diese dauerhaft in einer Datenbank speichert.

Damit das Tracking-Script den Aufbau der Seite nicht behindert, wird dieses asynchron geladen. Beim synchronen Vorgang wird das Javascript geladen und eingelesen. Alles was danach im Quellcode angegeben ist, wird erst geladen, wenn dies komplett abgeschlossen ist. Das kann die Aufbaugeschwindigkeit einer Seite spürbar beeinflussen. Durch das asynchrone Laden werden die nachfolgenden Ladevorgänge nicht blockiert.

3.2.2 Visualisierungsmethoden

Zweck der ganzen Datenerfassung ist es, die Daten zu einem späteren Zeitpunkt zur Verfügung zu stellen und einem Analysten die Möglichkeit zu geben, diese intuitiv zu analysieren. Dafür müssen die Daten visuell aufbereitet werden, da diese sonst schwer zu interpretieren sind.

Die erfassten Daten müssen bei der Analyse den entsprechenden Elementen der Seite zugeordnet werden. Dafür bietet sich eine Abbildung der Seite mit einem Overlay an. Das Overlay enthält dabei die Daten. In dem Overlay müssen zwei Ebenen dargestellt werden. Zum einen die räumliche Position und zum anderen die Anzahl der erfassten Mausklicks und Mauspositionen. Dafür eignet sich eine Heatmap. Eine Heatmap visualisiert Daten mit Farben. Die Präsenz von Farbe deckt die erste Ebene ab und bedeutet, dass an dieser Stelle Daten vorhanden sind. Die zweite Ebene wird durch den Farbton angegeben. Kalte Farben bedeuten in den meisten Fällen, dass an dieser Stelle im Verhältnis eine geringe Anzahl an Datenpunkten gegeben ist, warme Farben bedeuten, dass sehr viele Datenpunkte an dieser Stelle vorhanden sind. Die genaue Farbgebung und die entsprechende Anzahl an Datenpunkten wird durch eine Legende genauer erklärt.

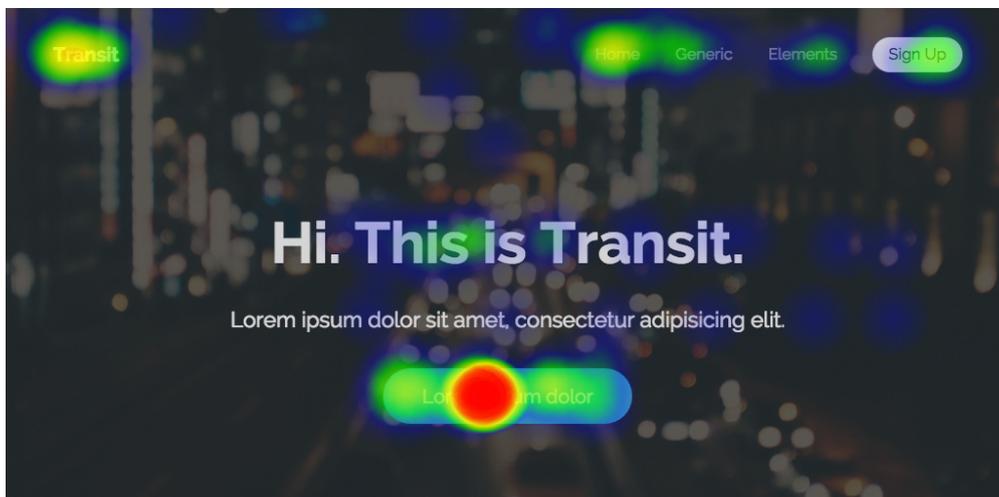


Abbildung 3.2: Beispiel einer Heatmap

Bei dieser Art der Darstellung wird allerdings die zeitliche Komponente vernachlässigt. Es könnte für den Analysten zusätzlich interessant sein, wie sich die Anzahl der Klicks im Laufe der Zeit entwickelt. Dafür wird ein Diagramm verwendet, welches die Klicks ins Verhältnis zur Zeit stellt. So kann schnell erfasst werden, ob sich die Anzahl der Klicks erhöht, oder verringert. Für einen schnellen Vergleich wird zusätzlich die Gesamtanzahl der Klicks als einfache Zahl dargestellt.

Durch diese Methoden lässt sich für den Analysten die Verteilung der Mausklicks und Mauspositionen leicht erfassen.

3.3 Datenbank

Die erfassten Daten müssen dauerhaft gespeichert werden, um diese auch zu einem späteren Zeitpunkt analysieren zu können. Daher wird eine Datenbank benötigt. Um eine geeignete Datenbank-Software zu finden, muss das passende Daten-Modell gefunden werden. Dabei stehen Relationale Datenbanken, Key-Value Datenbanken, dokumentorientierte Datenbanken und spaltenorientierte Datenbanken zur Verfügung. Eine Key-Value Datenbank fällt raus, da hier der Zugriff nur über einen einzigen, bestimmten Index erfolgen kann. Für diesen Anwendungsfall muss allerdings durch variable Entitäten auf die Daten zugegriffen werden können.

Ebenfalls nur eingeschränkt geeignet ist eine spaltenorientierte Datenbank. Die Stärke bei Datenbanken dieser Art liegt im Lesen von bestimmten Spalten anstatt von ganzen Zeilen. Die Selektion von Daten ist bei spaltenbasierten Datenbanken wesentlich effizienter als bei anderen Datenbanktypen. [Köl14] Da die Anwendung aber in den meisten Fällen mehrere Spalten abfragen muss, ist eine zeilenbasierte Datenbank sinnvoller.

Übrig bleiben relationale und dokumentorientierte Datenbanken. Der größte Unterschied ist, dass die zur NoSQL-Gattung gehörenden dokumentorientierten Datenbanken Schema-frei sind. Das heißt, die Datenbankstruktur muss im Vorfeld nicht festgelegt werden und die Struktur der Daten kann variabel sein. Dies hat Vorteile bei der Skalierung, jedoch auch Nachteile bei der Konsistenz. Relationale Datenbanken hingegen basieren auf Abhängigkeiten der Daten, was Einbußen in der Performance mit sich bringt. Das heißt, bei großen Datenmengen können dokumentorientierte Datenbanken punkten, da sie nicht durch aufzulösende Abhängigkeiten verlangsamt werden.

Da beim Tracken von Mausdaten sehr viele Daten anfallen, erscheint eine dokumentorientierte Datenbank als bessere Wahl. Die bekanntesten Vertreter sind MongoDB⁸ und CouchDB⁹. Beide nutzen JSON als Dokumentenform und sind somit einfach mit Node.js zu nutzen. Mit MongoDB ist es etwas einfacher dynamische Queries auszuführen, daher fällt die Wahl darauf. Verschiedene Node.js-Pakete bieten die Möglichkeit, auf einfache Weise mit MongoDB zu arbeiten.

Die Datenbank wird in einem eigenen Docker Container gestartet, welcher ausschließlich die Datenbank enthält. Da MongoDB sich als Cluster aufsetzen lässt, kann die Datenbank einfach skaliert werden, indem der selbe Docker Container auf weiteren Servern gestartet wird. Mehr dazu im folgenden Abschnitt 3.4.

3.4 Skalierung des Systems

Basierend auf den in den Kapiteln 3.1 und 3.3 getroffenen Entscheidungen wird eine Strategie zur Skalierung der Anwendung erstellt. Dabei wird zum einen die Parallelisierung auf dem Host und zum anderen die horizontale Skalierung behandelt.

3.4.1 Parallelisierung

Node.js verwendet standardmäßig nur einen CPU-Kern zur Verarbeitung der Anfragen. Durch Clustering können mehrere Prozesse gestartet werden, um die Leistung zu erhöhen. Dabei kann pro CPU-Kern ein Workerprozess gestartet werden. Ein Master-Prozess verteilt dabei die Anfragen gleichmäßig auf die Workerprozesse. Die Anwendung wird somit auf einem Host parallelisiert.

Auch die Performance der Datenbank kann durch das Hinzufügen von mehr CPU-Kernen erhöht werden. Die Verarbeitung von Anfragen erfordert Rechenleistung, die bei hoher Auslastung die Anwendung verlangsamen könnten. Allerdings ist dies physikalisch begrenzt, da nicht unendlich viele CPU-Kerne in einem System verbaut werden können. Vor allem steigt der Preis im Verhältnis zur Performancesteigerung für weitere Systeme stark an. Daher ist noch eine weitere Art der Skalierung je nach Anwendung sinnvoll.

3.4.2 Horizontale Skalierung

Mit Node.js ist eine horizontale, über mehrere Server verteilte, Installation nicht möglich. Dafür gibt es besser geeignete Software, die auf das Load Balancing ausgelegt ist. Wie im Kapitel 3.1 besprochen, ist nginx sehr leistungsstark auf diesem Gebiet. Daher ist es ohne Weiteres möglich, mehrere Node.js Instanzen auf verschiedenen Servern zu starten und die Anfragen mittels nginx zu routen.

⁸<https://www.mongodb.org>

⁹<http://couchdb.apache.org>

Die Datenbank ist im Gegensatz zum Server vor allem durch die Schreib- und Lesezugriffe auf die Festplatten beschränkt. MongoDB lässt sich ebenfalls durch sogenanntes Sharding auf mehreren Servern verteilen, um die Performance zu erhöhen. Dabei werden die Daten aufgeteilt, sodass jeder Datenbankserver nur einen bestimmten Teil der Daten enthält. So genannte Query-Router nehmen die Anfragen an die Datenbank an und holen die Daten vom entsprechenden Shard. Drei weitere Config-Server enthalten Metadaten, welche von den Query-Routern abgefragt werden können. Auch von den Query-Routern kann es mehr als einen geben, um die Rechenlast der Anfragen performanter verarbeiten zu können. [Monc]

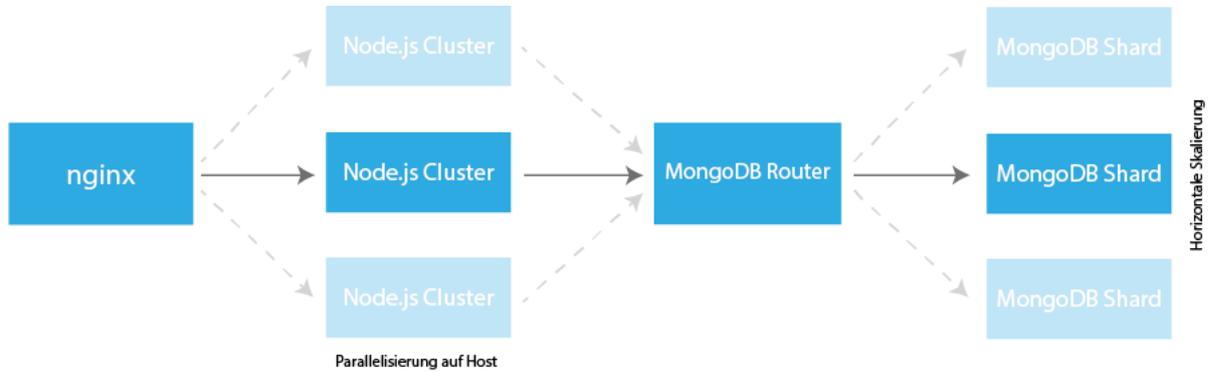


Abbildung 3.3: Skalierung der Systemarchitektur

3.4.3 Zuordnung von Anforderungen und Systemarchitektur-Komponenten

Die folgende Tabelle 3.2 ordnet die in Kapitel 2 erörterten Anforderungen den Komponenten der Systemarchitektur zu. Somit kann sicher gestellt werden, dass alle Anforderungen durch eine, oder mehrere Komponenten erfüllt werden. Dies wird durch ein Kreuz markiert.

Anforderung x Feature	Webseite	Portal	Datenbank
Req1 (Mausklicks erkennen)	x		
Req2 (Mauspositionen erkennen)	x		
Req3 (Ermittelte Daten an Server senden)	x		
Req4 (Ermittelte Daten in Datenbank speichern)		x	x
Req5 (Ermittelte Daten von Server abfragen)		x	x
Req6 (Mausevents nach Referrer filtern)		x	x
Req7 (Mausevents nach Endgerät filtern)		x	x
Req8 (Mausevents nach URL filtern)		x	x
Req9 (Mausevents nach Datum filtern)		x	x
Req10 (Mausevents nach Typ filtern)		x	x
Req11 (Mausevents darstellen)		x	
Req12 (Seite auswählen)		x	

Tabelle 3.2: Zuordnung von Anforderungen und Systemarchitektur

Kapitel 4

Softwarearchitektur und Softwarekomponenten-Design

In diesem Kapitel wird die Umsetzung der einzelnen Komponenten der Anwendung, sowie deren Schnittstellen erläutert. Dies geschieht auf Basis der zuvor besprochenen Systemarchitektur.

4.1 Portal

Das Portal besteht aus einer serverseitigen und einer clientseitigen Komponente. Auf dem Server selber werden die ermittelten Daten angenommen und verarbeitet. Außerdem wird das Portal für den Client ausgeliefert und das Tracking-Script für die Webseiten bereitgestellt. Im Browser läuft beim Client die Anwendung für das Portal, welches mit dem Server kommuniziert.



Abbildung 4.1: Softwarekomponente Portal

4.1.1 Server

Die Anwendung auf dem Server verwendet das MVC-Prinzip. Dies bedeutet, dass Model, View und Controller logisch voneinander getrennt werden. Die Grundlage der Anwendung ist das Node.js Framework Express. Mit diesem werden die Anfragen an den Server bearbeitet, indem entsprechende Routen angelegt werden. Die Routen sind die Controller und beinhalten dynamische Berechnungen, wie zum Beispiel die Abfrage von Daten aus der Datenbank. Die Abfrage von Daten aus der MongoDB Datenbank wird mit dem Node.js Package mongoose realisiert. Dieses bietet die Möglichkeit, Models zu definieren und diese in der Datenbank zu speichern, ändern, löschen etc. Für die View-Komponente wird die Template-Engine Jade verwendet. Die Templates werden von den Controllern kompiliert und an den Benutzer ausgeliefert.

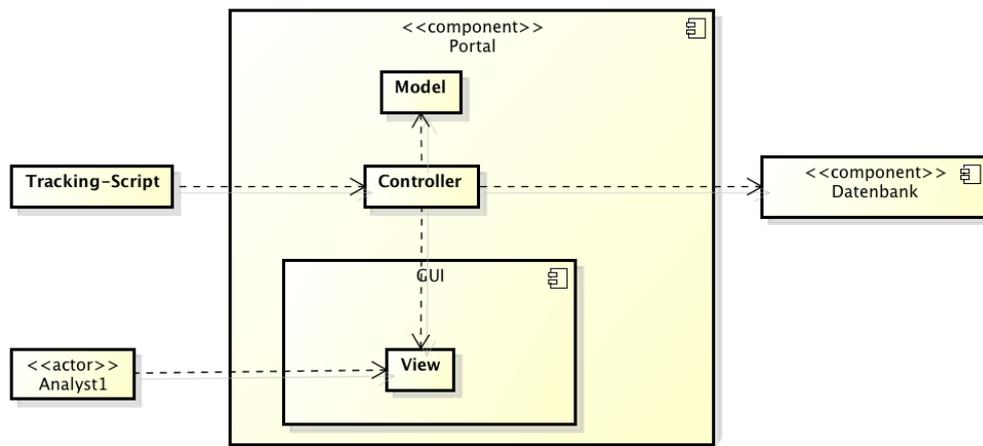


Abbildung 4.2: Komponenten Diagramm

Das Portal bietet zusätzlich noch eine Datenschnittstelle für den Client. Diese dient dazu, Daten für die Visualisierungen asynchron per Ajax abzufragen. Somit muss nicht bei jeder Änderung des Filters die komplette Seite neu geladen werden. Zudem lädt das Portal initial schneller, da die Daten asynchron nachgeladen werden und den Aufbau der Seite nicht behindern.

4.1.2 Client

Für das clientseitige Portal für den Analysten wird das Revealing Module Pattern verwendet. [Osm14] Bei diesem werden gekapselte Module erstellt, welche private und öffentliche Methoden haben. Um dies zu realisieren wird für jedes Modul eine Variable im globalen Namespace¹ angelegt. Dies wird gemacht, um möglichst globale Variablen zu vermeiden. Tut man dies nicht kann es schnell passieren, dass Variablen unwissentlich überschrieben werden. [Tru13] Durch eine anonyme Funktion ist es zunächst nicht möglich auf Funktionen und Variablen innerhalb des Moduls zuzugreifen.

Um Zugriff auf bestimmte Funktionen und Variablen zu erhalten, werden diese von dem Modul als Objekt zurückgeliefert. Dadurch ist eine klare Trennung von öffentlichen und privaten Variablen und Funktionen gegeben, was es so in der Javascript Spezifikation nicht gibt. Für jedes abkapselbare Element des Portals wird ein Modul erstellt, welche mittels der öffentlichen Methoden untereinander kommunizieren können.

Folgender Code zeigt das Prinzip des Revealing Module Patterns. Voran gestellte Unterstriche dienen der Übersichtlichkeit, um private von öffentlichen Funktionen bzw. Variablen zu unterscheiden.

```

1 var Module = (function() {
2
3   var _privateVar;
4
5   // private Funktion
6   var _privateFunc = function() {
7     console.log("Hallo Welt!");
8   };
9
10  // oeffentliche Funktion
11  var publicFunc = function() {
12    console.log("Hallo Welt!");
13  };
14
15  // oeffentliche Funktionen und Variablen zurueckliefern

```

¹Gruppierung von Code unter einem bestimmten Namen

```

16     return {
17         publicFunc: publicFunc
18     };
19 }();

```

Listing 4.1: Revealing Module Pattern

Das definierte Modul kann folgendermaßen (nicht) verwendet werden:

```

1 Module.publicFunc();
2 // => Hallo Welt!
3
4 Module._privateFunc();
5 // => Fehler

```

Listing 4.2: Verwendung Revealing Module Pattern

4.2 Tracking-Script

Über das Portal wird das Tracking-Script geladen. Dies ist ein kleines Script, welches sich auf jeder beliebigen Seite einbinden lässt. Alle Daten, die es erfasst, werden zurück zum Portal gesendet, um dort weiter verarbeitet und gespeichert zu werden. Das Tracking-Script folgt ebenfalls dem in Abschnitt 4.1.2 vorgestelltem Revealing Module Pattern, um möglichst die unnötige Registrierung von globalen Variablen zu vermeiden. So wird nur eine einzige öffentliche Variable als Namespace erstellt.



Abbildung 4.3: Softwarekomponente Tracking-Script

Um die zu analysierende Seite durch das Tracking-Script nicht negativ zu beeinflussen, wird es asynchron geladen. In modernen Browsern ist dies durch ein Attribut im Script-Tag möglich.

```

1 <script src="http://example.com/script.js" async></script>

```

Listing 4.3: Script-Tag mit async Parameter

Dadurch wird dieses parallel zu den normalen Inhalten geladen und der Aufbau der Seite nicht blockiert. Allerdings funktioniert dies vor allem in älteren Browsern nicht. Um diese Funktionalität für alle Browser zu gewährleisten ist ein kleines Script nötig.

```

1 (function() {
2     // Script-Element erstellen
3     var el = document.createElement("script"),
4         // erstes Script-Element
5         scripts = document.getElementsByTagName("scripts")[0];
6     // async-Attribut setzen
7     el.async = 1;
8     // Quelle setzen

```

```
9     el.src = "http://example.com/script.js";
10     // neues Script-Element einfüegen
11     scripts.parentNode.insertBefore(el, scripts);
12 }());
```

Listing 4.4: Script zum asynchronen Laden von Javascript

Das Script erstellt ein Script-Tag mit der Quelle des asynchron zu ladenden Codes und dem `async`-Attribut (für neuere Browser und als Hinweis). Dieses Script-Tag wird vor das erste im DOM-Tree stehende Script-Tag gesetzt. Da dieses beim anfänglichem Laden der Seite noch nicht vorhanden war, wird es asynchron heruntergeladen.

Das Script wird im Produktivbetrieb noch verkleinert und von unnötigen Zeichen befreit, sodass möglichst wenig zusätzliche Bytes geladen werden müssen.

4.2.1 Übermittlung der Daten

Die naheliegendste Möglichkeit zum dynamischen Übermitteln der Daten an den Server wäre ein AJAX-Request² an den Server. Da die Seite mit dem eingebundenem Tracking-Script unter einer anderen Domain erreichbar ist (bzw. in diesem Fall unter einem anderen Port), wird dieser den AJAX-Request ablehnen. Dieses Verhalten wird durch die Same-origin policy hervorgerufen. [Rud08]

Um dies zu umgehen, gibt es das Cross-Origin Resource Sharing Prinzip (CORS). Dafür werden entweder Zugriffe von allen externen Domains oder von bestimmten Domains zugelassen. [Hos11] Um eine AJAX-Abfrage zu starten wird zunächst beim Server um Erlaubnis gefragt und bei positiver Antwort ausgeführt. Die Unterstützung für CORS ist relativ groß, jedoch in älteren Browsern nicht zwangsläufig gegeben. Daher wird diese Option nicht verwendet.

Anders als bei AJAX-Requests gibt es mit WebSockets die Möglichkeit, eine dauerhafte Verbindung zwischen einer Seite und einem Server aufzubauen. Dies fällt nicht unter die Same-origin policy. Allerdings ist es für diese Art der Anwendung nicht nötig eine dauerhafte Verbindung aufrecht zu erhalten. Diese müsste ebenfalls zusätzlich abgesichert werden. Da dies außerdem eine sehr neue Technologie und noch nicht von allen Browsern ausreichend unterstützt wird, scheidet diese Möglichkeit ebenfalls aus.

Eine andere Alternative ist die Verwendung eines sogenannten Web Bugs (oder auch Tracking Bug, Tracking-Pixel, Web Beacon). Bei dieser Methode wird durch einen einfachen HTTP GET Befehl ein 1x1 Pixel großes Bild angefordert. Dies ist möglich, da sich die Same-origin Policy auf AJAX-Requests beschränkt. An die Anfrage für das Bild werden weitere Parameter angehängt, welche von dem Server ausgelesen werden können. Somit können Daten von der Webseite an den Server übermittelt werden.

Die Verwendung eines Web Bugs ist für diese Anwendung die beste Methode, um die Daten zu transferieren. Es wird keine zusätzliche Software benötigt und diese Methode wird von allen Browsern unterstützt, da das Abfragen eines Bildes zum Standard eines Browsers gehört. Daher wird für diese Software ein Web Bug zur Datenübertragung gewählt.

4.3 Datenbank

Die Datenbank wird über das Portal angesprochen. Dafür nutzt das Portal das Object Modelling Paket `mongoose` für `Node.js`. Über dieses werden die Abfragen an die `MongoDB` Datenbank generiert. Die Verbindung wird über ein `TCP/IP` Socket hergestellt. [Monb] Der benutzte Port ist der Standard-Port 27017. Die Datenbank muss nicht zwangsläufig auf dem gleichen Server wie das Portal liegen. Weiteres dazu in Kapitel 3.4.2.

²Asynchronous Javascript and XML: Asynchrone Datenübertragung zwischen Browser und Server

In der Datenbank wird mit dem Speichern des ersten Datensatzes eine Kollektion angelegt, welche die ermittelten Daten beinhaltet.



Abbildung 4.4: Softwarekomponente Datenbank

Kapitel 5

Beschreibung der Implementierung

In diesem Kapitel werden die implementierten Komponenten genauer erläutert. Dafür wird mit exemplarischen Programmteilen aus der Anwendung die Funktionsweise erklärt. Die Gliederung orientiert sich an der Softwarearchitektur aus Kapitel 4.

5.1 Portal

Die Programmierung des Portals unterscheidet sich zwischen Client, welcher im Browser läuft, und dem Server, auf dem der Programmcode für die Bereitstellung des Clients läuft. Auf dem Server werden die Berechnungen und Datenbankabfragen vorgenommen, um dies anschließend dem Client zur Verfügung zu stellen. Dieser kann im Browser mit der Anwendung interagieren.

5.1.1 Server

Wie in der Softwarearchitektur bereits erläutert, wird das Portal mit dem MVC-Pattern erstellt. Nachfolgend wird die Implementierung dieser Herangehensweise exemplarisch dargestellt und erläutert.

Model

Entsprechend der Anforderungsanalyse werden alle erfassten Mausevents durch ein einziges Model repräsentiert. Die Unterscheidung erfolgt durch den Parameter Type. Alle anderen Eigenschaften überschneiden sich. Models werden über das Node.js Package mongoose realisiert. Es wird festgelegt, welche Parameter das Model hat, welcher Art diese Parameter sind und ob diese zwingend benötigt werden. Nachfolgend das Model Event, welches für alle Mausevents verwendet wird. Das Model entspricht den Attributen aus der Tabelle 2.1, ergänzt durch einige parsed-Felder, welche geparste Eigenschaften enthalten.

```
1 var eventSchema = mongoose.Schema({
2   // Zeitstempel
3   timestamp: {
4     type: Date,
5     default: Date.now
6   },
7   // Koordinaten relativ zu Ziel-Element
8   coordinates: {
9     type: mongoose.Schema.Types.Mixed,
10    required: true
11  },
12  // eindeutige Session-ID
```

```
13 session_id: {
14     type: String,
15     required: true
16 },
17 // Original User-Agent-String
18 user_agent: {
19     type: String,
20     required: true
21 },
22 // geparster User-Agent-String
23 parsed_user_agent: {
24     type: mongoose.Schema.Types.Mixed,
25     required: false
26 },
27 // original Seiten-URL
28 page_url: {
29     type: String,
30     required: true
31 },
32 // geparste Seiten-URL
33 parsed_page_url: {
34     type: mongoose.Schema.Types.Mixed,
35     required: true
36 },
37 // Aufloesung
38 resolution: {
39     type: mongoose.Schema.Types.Mixed,
40     required: true
41 },
42 // Referrer-URL
43 referrer_url: {
44     type: String
45 },
46 // DOM-Element des Ziels
47 element: {
48     type: mongoose.Schema.Types.Mixed,
49     required: true
50 },
51 // Art des Mausevents
52 type: {
53     type: String,
54     required: true
55 }
56 });
```

Listing 5.1: Event-Model

Empfängt das Portal Daten über das Tracking-Script wird ein Model-Objekt über das Schema angelegt und mongoose prüft, ob alle benötigten Eigenschaften gesetzt und ob die entsprechenden Datentypen korrekt sind. Ist dies nicht der Fall gibt es einen Fehler, welcher in der Konsole protokolliert wird. Ist alles korrekt, wird das neue Objekt in der Datenbank gespeichert.

```
1 // neues Event erstellen
2 var event = new Event(data);
3
4 // in Datenbank speichern
5 event.save(function(err) {
6     if (err) {
7         // Fehler in der Konsole ausgeben
8         console.log(err);
9     }
10 });
```

Listing 5.2: Speicherung eines Event-Objektes

View

Der View besteht aus Jade Template-Dateien. Diese haben eine vereinfachte Syntax für HTML-Tags und können dynamische Elemente durch einfache Schleifen und if-Abfragen abbilden. Das folgende Beispiel zeigt einen Teil der Startseite zur Auswahl der zu analysierenden Seite. Es werden einige Container angegeben und eine Liste aus den vorhandenen Seiten generiert. Die Werte dafür werden im Controller aus der Datenbank geholt.

```

1 extends layout
2 block content
3   .content.start
4     .tab-content
5       .container
6         .tab-pane.active#sites
7           .row
8             .col-xs-4.col-xs-offset-4
9               .card.no-padding
10                ul
11                  if sites
12                    each val, key in sites
13                      li.site
14                        a(href=key title=val) #{val}
15                  else
16                    li.no-data
17                      | Keine Daten vorhanden!

```

Listing 5.3: Teil eines Jade-Templates

Diese Dateien werden durch die Controller zu validem HTML kompiliert.

```

1 <div class="content start">
2   <div class="tab-content">
3     <div class="container">
4       <div class="tab-pane active" id="sites">
5         <div class="row">
6           <div class="col-xs-4 col-xs-offset-4">
7             <div class="card no-padding">
8               <ul>
9                 <li class="site">
10                  <a href="http://192.168.59.103:7623/site/192.168.59.103%3A7000%2F"
11                    title="192.168.59.103:7000">192.168.59.103:7000</a>
12                </li>
13              </ul>
14            </div>
15          </div>
16        </div>
17      </div>
18    </div>
19  </div>

```

Listing 5.4: Kompiliertes Jade-Template

Controller

Die Controller sind in dieser Anwendung die Routen, welche vom Client im Browser angesteuert werden können. Eine Anfrage wird immer von einem bestimmten Controller bearbeitet. Dafür wird im Controller

zunächst die Methode und der Pfad angegeben, auf welchen dieser hören soll. Der Request wird empfangen und eine Response wird zurückgegeben. Analog zu dem Beispiel des Views folgt der Controller für die Startseite. Es werden aus der Datenbank alle möglichen Werte für die Seiten URL einmalig herausgesucht und formatiert. Das entsprechende Template index wird gerendert und die Seiten URLs werden als Variable übergeben. Mit diesen Daten wird das HTML-Dokument generiert.

```

1 // Index Route
2 app.get("/", function(req, res) {
3   var sites;
4
5   // Alle URL Vorkommen aus der Datenbank auslesen
6   Event.distinct("page_url").exec(function(err, urls) {
7
8     var sites = {};
9     // verschoenerte URL Version generieren
10    _.forEachRight(urls, function(value, key) {
11      sites["http://192.168.59.103:7623/site/" + encodeURIComponent(value)] = utils.
        beautifyURL(value);
12    });
13
14    // Jade-Template komilieren
15    res.render("index", {
16      sites: sites
17    });
18  });
19 });

```

Listing 5.5: Controller für die Index-Route

Die folgende Tabelle zeigt alle Routen, welche das Portal nutzt. Zum einen gibt es die Routen, welche zur Darstellung des Portals dienen, und zum anderen Routen, welche ausschließlich Daten im JSON-Format ausliefern. Diese werden genutzt, um Daten asynchron zu laden.

Route	Beschreibung
/collect	Verwendet vom Tracking-Script, sammelt Daten
/	Startseite des Portals
/site/%URL%	Detailseite für zu analysierende Webseiten
/api/heatmap	Liefert Heatmap Koordinaten im JSON-Format
/api/count	Liefert Anzahl Klicks im JSON-Format
/api/history	Liefert Zeit-Daten für Diagramm im JSON-Format

Tabelle 5.1: Verwendete Routen des Portals

Für die Generierung der Heatmap müssen zunächst die Koordinaten der Datenpunkte berechnet werden. Dies ist notwendig, da die Browser bei der Aufzeichnung höchst wahrscheinlich verschiedene Auflösungen haben und die Einordnung in ein einfaches Koordinatensystem verschiedene Ergebnisse liefern würde. Daher werden mittels des Zielelements und der relativen Position des Mauszeigers zu diesem bei jeder Generierung der Heatmap neue Koordinaten berechnet.

Um die Daten für die Heatmap zu generieren, wird mit PhantomJS¹ ein sogenannter headless Browser gestartet. Headless bedeutet dabei, dass der Browser nicht sichtbar ist, was normalerweise der Zweck eines Browsers ist. Dieser nimmt als Argumente die zu öffnende URL, die Größe des Browserfensters und die gespeicherten Daten entgegen. Alle Datenpunkte werden durchlaufen und Anhand der geöffneten Seite berechnet. So können Daten mit unterschiedlichen Auflösungen für eine bestimmte Browsergröße berechnet und korrekt dargestellt werden.

¹<http://phantomjs.org>

```
1 // Viewport Groesse setzen
2 page.viewportSize = {
3   width: system.args[3],
4   height: system.args[4]
5 };
6
7 // Seite oeffnen
8 page.open(system.args[1], function(status) {
9   var data = JSON.parse(system.args[2]);
10
11   // lodash laden
12   var injected = page.injectJs("./public/bower_components/lodash/lodash.min.js");
13
14   if (injected) {
15     var events = page.evaluate(function(data) {
16       var ret = [], el, xpathres, i;
17
18       // Hilfsmethode um Position eines Elements herauszufinden
19       var _getElementPosition = function(el) {
20         var box = el.getBoundingClientRect(),
21             docElem = document.documentElement;
22
23         return {
24           top: box.top + window.pageYOffset - docElem.clientTop,
25           left: box.left + window.pageXOffset - docElem.clientLeft
26         };
27       };
28
29       // Daten durchlaufen
30       _.forEach(data, function(n, key) {
31         // XPATH auflösen
32         xpathres = document.evaluate(n.element.path, document.body, null, XPathResult.
33           ANY_TYPE, null);
34         el = xpathres.iterateNext();
35
36         if (el !== null) {
37           var elPos = _getElementPosition(el);
38
39           // Koordinaten berechnen
40           var pos = {
41             x: Math.round(elPos.left + n.coordinates.x),
42             y: Math.round(elPos.top + n.coordinates.y)
43           };
44
45           // Daten zu Array hinzufuegen
46           var search = _.findIndex(ret, pos);
47           if (search === -1) {
48             ret.push(_.merge(pos, {
49               value: 1
50             }));
51           } else {
52             ret[search].value = ret[search].value + 1;
53           }
54         }
55       });
56       return ret;
57     }, data);
58
59     // Daten zurueckgeben
60     console.log(JSON.stringify(events));
61   }
62
63   // PhantomJS beenden
```

```
63     phantom.exit();
64 });
```

Listing 5.6: Berechnung der Positionsdaten

Bei der Berechnung ist allerdings zu beachten, dass es gegebenenfalls verschiedene Ansichten der Seite bei verschiedenen Auflösungen geben kann. Daher sollten gewisse Abstufung gemacht und dafür jeweils Koordinaten berechnet werden. Eine erste grobe Abstufung ist die Einstufung in die drei Gruppen Desktop, Tablets und Mobile Endgeräte. Dafür werden bestimmte Maße für jede Gruppe festgelegt. Allerdings muss dabei beachtet werden, dass allein durch die Auflösung nicht jedes Gerät eindeutig einer Gruppe zugeordnet werden kann. Im Rahmen dieser Arbeit soll dies zunächst genügen. Die Daten werden dabei in die entsprechenden Gruppen eingeordnet.

5.1.2 Client

Programmierung

Der Client besteht aus den kompilierten Templates und wird als HTML ausgeliefert. Mittels Javascript können diese dynamisch verändert werden. Mit dem Revealing Module Pattern wird dies in die Komponenten Portal, GUI, Heatmap und Diagramme aufgeteilt. Nachfolgend beispielhaft das Modul Heatmap.

```
1  var Heatmap = (function(window, document, $, undefined) {
2      "use strict";
3
4      var _heatmap,
5          _config = {},
6          _params = {};
7
8      // Daten vom Server laden und Heatmap aktualisieren
9      var update = function() {
10         // Status setzen
11         Portal.isloading = true;
12
13         // jQuery promise zurueckliefern
14         return $.getJSON(Portal.generateURL("heatmap"), function(res) {
15             // Daten der Visualisierung setzen
16             _heatmap.map.setData({
17                 max: res.max,
18                 data: res.data
19             });
20
21             // Legende aktualisieren
22             $(".legend .max").text(res.max);
23             // Status setzen
24             Portal.isloading = false;
25         });
26     };
27
28     // iFrame fuer die Visualisierung erstellen
29     var _createIframe = function(url, cont) {
30         return $("&amp;amp;amp;lt;/div&amp;amp;amp;gt;
```

```
40     return {
41         map: h337.create({
42             container: el,
43             backgroundColor: "rgba(0,0,0,.1)"
44         }),
45         el: $(el).parent("div")
46     };
47 };
48
49 // Visualisierung erstellen
50 var create = function(wrapper) {
51     // Ladeanimation anzeigen
52     GUI.loader("show");
53     // Status setzen
54     Portal.isloading = true;
55     var $wrapper = $(wrapper),
56         $iframe = _createIframe("http://" + _params.page_url).appendTo($wrapper),
57         $heatmapWrap = $('<div id="heatmap" style="width:100%;height:100%;" />').appendTo(
58             wrapper);
59
60     // Breite der Heatmap setzen
61     $(".heatmap-wrapper").width(_config.devices[_params.device]);
62
63     // Groesse des iFrame-Wrappers abhaengig von der iFrame-Groesse setzen
64     $(window).one("message onmessage", function(e) {
65         var height = e.originalEvent.data;
66         $iframe.height(height);
67         $wrapper.height(height);
68
69         _heatmap = _createHeatmap($heatmapWrap.get(0));
70
71         Heatmap.update().done(function() {
72             GUI.loader("hide");
73         });
74     });
75
76     // warten bis iFrame geladen ist
77     $iframe.load(function() {
78         this.contentWindow.postMessage("size", "*");
79         Portal.isloading = false;
80     });
81 };
82
83 // Heatmap entfernen
84 var remove = function() {
85     _heatmap = [];
86     $(".heatmap-wrapper").empty();
87 };
88
89 // Heatmap initialisieren
90 var init = function() {
91     // Einstellungen laden
92     _config = Portal.getConfig();
93     // Parameter laden
94     _params = Portal.getParams();
95
96     if $(".heatmap-wrapper").length > 0 {
97         Heatmap.create(".heatmap-wrapper");
98     }
99 };
100
101 // oeffentliche Methoden zurueckliefern
102 return {
```

```
102     update: update,  
103     create: create,  
104     remove: remove,  
105     init: init  
106   };  
107  
108 }(this, document, jQuery);
```

Listing 5.7: Portal Modul Heatmap

Alle Module haben eine init-Funktion, welche einmalig beim Aufrufen des Portals ausgeführt wird. Hier werden beispielsweise Einstellungen gesetzt oder geladen und Elemente generiert. Zudem gibt es in jedem Modul eine update-Funktion, welche die aktuellsten Daten vom Server lädt und die Visualisierungen entsprechend aktualisiert. Weitere Modul spezifische und interne Methoden werden für die jeweilige Darstellung verwendet. Es werden dabei nur für andere Komponenten benötigte Methoden nach außen freigegeben. Als Basis für die Komponenten werden die aktuellen Parameter in der Haupt-Komponente gespeichert. So haben alle Komponenten die gleichen Parameter und die Visualisierungen zeigen entsprechend die selben Daten. Beim Anpassen der Parameter über die Menüleiste werden die gespeicherten Parameter angepasst und alle Komponenten können sich darauf basierend aktualisieren.

Die Daten für die Heatmap werden über die entsprechende Route mittels AJAX abgefragt und durch die Bibliothek `heatmap.js`² zu einer Heatmap generiert. Das halbtransparente Canvas-Element wird über ein iFrame gelegt, in welchem die zu untersuchende Seite geladen wird. So können die Datenpunkte den Elementen der Seite zugeordnet werden. Für die Darstellung von Diagrammen wird `C3.js`³ verwendet. Die auf `D3.js`⁴ basierende Bibliothek ermöglicht das einfache Erstellen von Diagrammen jeglicher Form. Die Daten werden ebenfalls über eine Route asynchron nachgeladen.

Gestaltung

Gemäß dem Google Material Design Guide sind die Elemente auf unterschiedlichen Ebenen im Raum platziert. Es gibt zunächst die Grundfläche, auf der unmittelbar die Inhalte der Seite folgen. Diese sind entweder direkt auf der Grundfläche, oder in sogenannten Cards organisiert. Die Cards bilden dabei die zweite Ebene. Die nächste Ebene wird nur bei Ladevorgängen angezeigt und enthält eine Ladeanimation. Da diese Ebene die komplette Browsergröße umfasst, wird hier auf einen Schatten verzichtet. Alle Inhalte der Seite werden dabei verdeckt. Der Header und die Menüleiste bilden eine weitere Ebene. Die Inhalte der Seite schieben sich unter diese. Die Menüleiste bleibt am oberen Bildschirmrand zur einfacheren Bedienung hängen. Die letzte Ebene, die dem Betrachter auch gleichzeitig räumlich am dichtesten angeordnet ist, sind die Auswahlfelder der Bedienelemente. Dies sind zum Beispiel der Datepicker für die Datumsauswahl und die Optionen der Select-Felder. Die folgende Grafik verdeutlicht das Prinzip der Ebenen.

²<http://www.patrick-wied.at/static/heatmapjs>

³<http://c3js.org>

⁴<http://d3js.org>

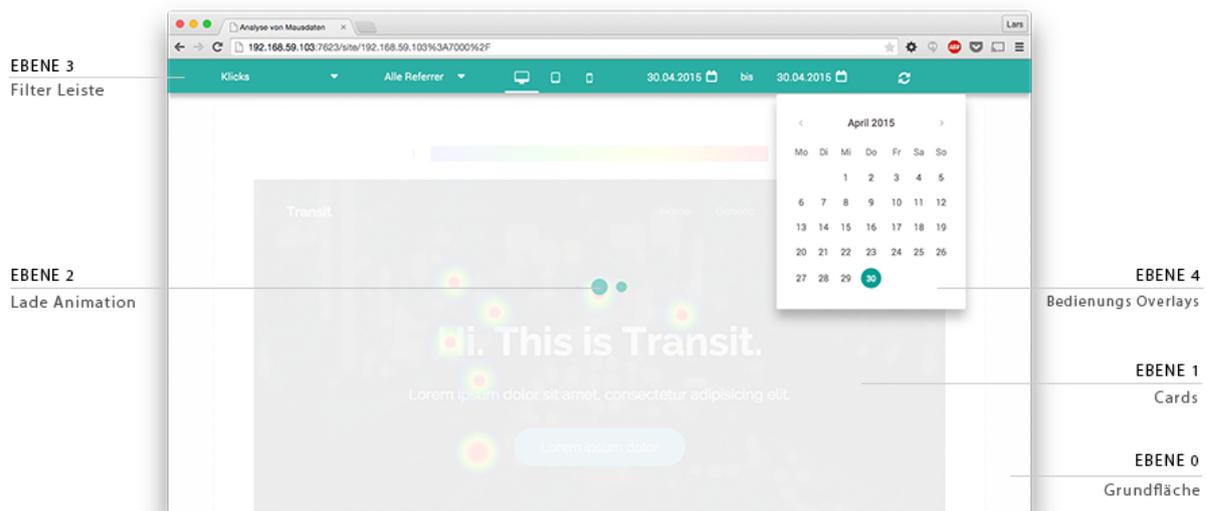


Abbildung 5.1: Material Design: Ebenen im Raum

Auf der Einstiegsseite wird für jede Seite, für die Daten vorliegen, ein Auswahlfeld mit der URL angezeigt. Die URL ist dabei vereinfacht, indem das Protokoll weggelassen wird. Dies dient der Ästhetik und Übersichtlichkeit. Klickt man auf eines der Auswahlfelder wird man auf eine weitere Seite geleitet, auf welcher die Daten der ausgewählten Seite dargestellt werden. Sind keine Daten vorhanden, wird eine entsprechende Meldung angezeigt. Ein weiterer Tab zeigt den Tracking-Code an, welcher benötigt wird, um Daten zu erfassen.

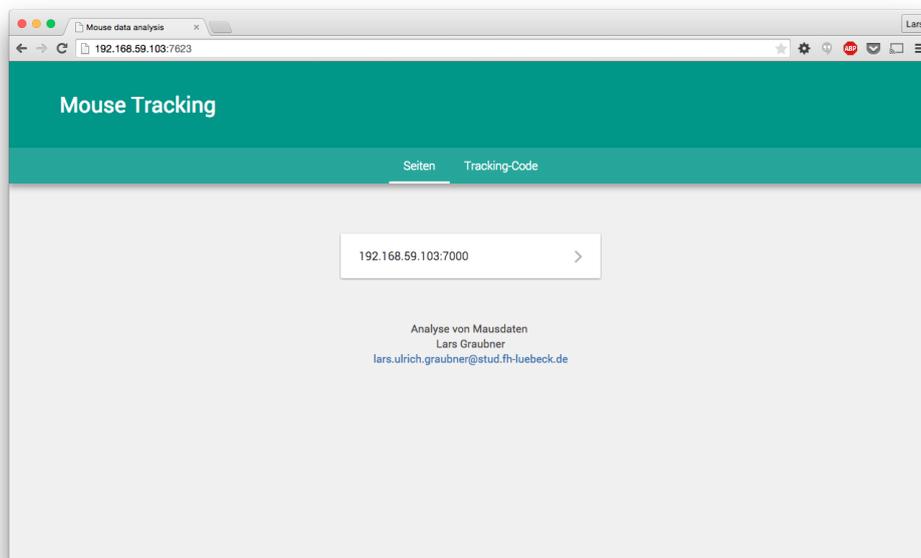


Abbildung 5.2: Einstiegsseite Portal

Hat man eine Seite ausgewählt, gelangt man auf eine die Detail-Seite. Im Header wird die URL der Seite angezeigt, damit man weiß zu welcher Seite die Daten gehören. In der Menüleiste werden verschiedene Elemente zur Anpassung der Visualisierungen angezeigt. Sobald ein Wert geändert wird, blendet eine Ladeanimation ein und die Visualisierungen passen sich dynamisch an. Mittels eines Select-Feldes kann

ausgewählt werden, ob die Heatmap die Klick-, oder die Mauspositionsdaten anzeigen soll. Sofern es verschiedene Referrer gibt, können diese ebenfalls über ein Select-Feld gefiltert werden. Drei Icons stellen verschiedene Endgeräte dar. Standardmäßig ist die Desktop-Option ausgewählt. Wählt man ein anderes Endgerät, werden die Daten entsprechend gefiltert und die Visualisierungen aktualisiert. Die Heatmap ändert zudem die Größe entsprechend des Endgeräts. Zwei weitere Felder zeigen das Start- und das Enddatum an. Mittels eines Datepickers lassen sich diese anpassen. Ein weiterer Button aktualisiert alle Visualisierungen, somit muss nicht das komplette Portal neu geladen werden, wenn es neue Daten gibt. Um die Funktion der Buttons mit Icon zu verdeutlichen, erscheint ein Tooltip, sobald man mit der Maus über diese fährt. Die Tooltips sind gemäß dem Google Material Design gestaltet. [Goo14a] Um dem Benutzer Feedback zu Interaktionen mit der Seite zu geben, gibt es bei Klick auf die Buttons und Auswahlfelder einen visuellen Effekt. Bei dem ripple-Effekt füllt sich das Element durch einen größer werdenden Kreis, ausgehend vom Punkt des Klicks, mit Farbe. [Goo14b]

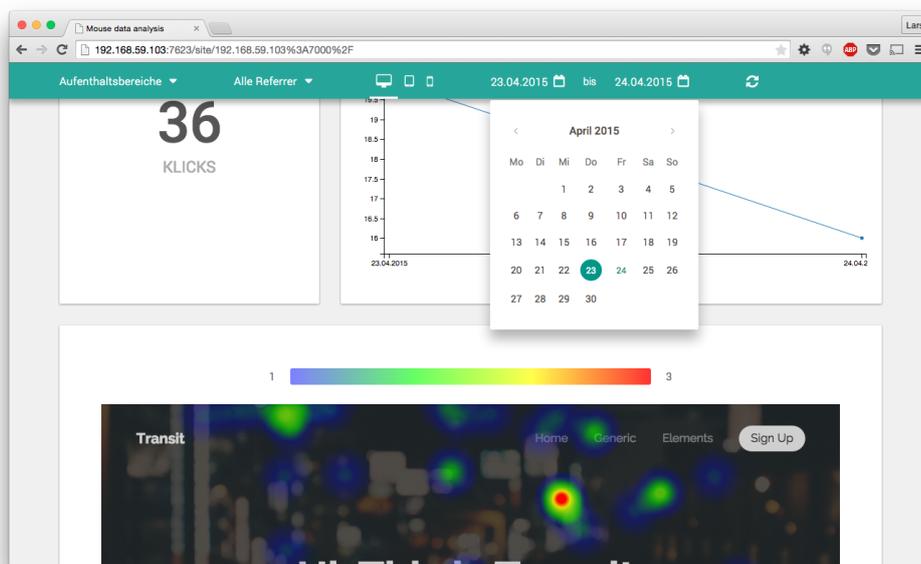


Abbildung 5.3: Aufbau des Portals

Die Inhaltselemente und Visualisierungen werden auf Cards dargestellt. Somit ist auch die Anordnung modular aufgebaut. Auf der ersten Card wird lediglich die Anzahl der Klicks für die aktuellen Parameter angezeigt. Die zweite Card enthält ein Diagramm, welches die Entwicklung der Klickzahlen anzeigt. Es besteht aus einer Abszisse, welche das Datum anzeigt, und einer Ordinate mit der Anzahl der Klicks zu dem entsprechendem Zeitpunkt. Die dritte Card enthält die Heatmap. Die Heatmap besteht aus einem iFrame, in welchem die zu untersuchende Seite geladen wird. Über diesem wird die generierte Heatmap gelegt. Somit ist jegliche (hier unerwünschte) Interaktion mit der Seite unterbunden. Über der Heatmap wird eine Legende angezeigt, in der die Farben der Heatmap eingeordnet werden. Blau ist dabei immer 1, also ein Datensatz für die entsprechende Position. Rot passt sich dynamisch an und gibt die maximale Anzahl von Datensätzen für einen Punkt an.

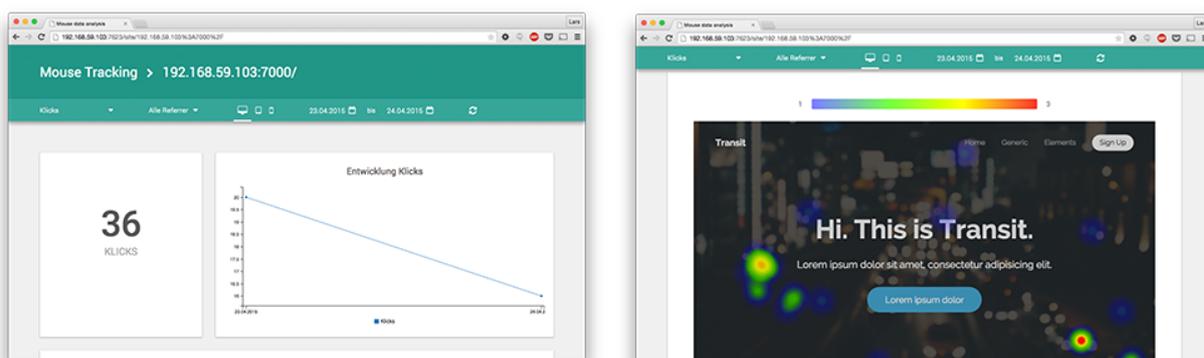


Abbildung 5.4: Visualisierungen im Portal

5.2 Tracking-Script

Das Tracking-Script ist eine Javascript-Datei, welche auf dem Server des Portals liegt. Wurde das Tracking-Script auf einer Seite erfolgreich asynchron geladen, initialisiert es sich selbst und legt verschiedene Event-Handler an, welche die erforderlichen Daten erfassen. Diese Daten werden dann durch einen Web Bug versendet. Dafür wird ein neues Bild mit der Größe 1x1 Pixel angelegt und die Quelle für das Bild gesetzt. Der Browser versucht nun automatisch das Bild von der angegebenen Adresse zu laden. An dieser Adresse sind die Parameter angehängt, welche an den Server gesendet werden sollen. Dieser kann die Parameter auslesen und in der Datenbank speichern. Das Bild selber wird nicht in den DOM der Seite eingefügt und ist somit auch nicht sichtbar.

```

1 var img = new Image(1, 1),
2   url = "http://192.168.59.103:7623/collect?event=";
3
4 // Bild wird angefordert/Daten werden versendet
5 img.src = url + JSON.stringify(data);

```

Listing 5.8: Erstellen eines Webbugs

5.3 Skalierung

Damit die Anwendung auch eine hohe Auslastung bewältigen kann, ist diese für eine einfache Skalierung optimiert. Dabei gibt es, wie in Kapitel 3.4 beschrieben, verschiedene Möglichkeiten.

5.3.1 Portal

Parallelisierung

Die einfachste Form der Skalierung ist die Parallelisierung auf dem Host. Das heißt, dass Portal nutzt mehr als einen CPU-Kern, sofern auf dem Server vorhanden. Um dies zu nutzen, muss in der Konfiguration "cluster" auf true gesetzt werden.

```

1 module.exports = {
2
3   //...
4
5   cluster: true

```

```
6 };
```

Listing 5.9: Aktivierung des Clusters in der Konfiguration

Ist die Option aktiviert, wird der Server wie gewohnt gestartet. Es wird ein Master-Prozess initialisiert, welcher für jeden CPU-Kern einen Worker-Prozess erstellt. Die Anfragen werden dann über den Master je nach Auslastung an die Worker verteilt.

```
1 // pruefen ob cluster aktiviert und Master-Prozess aktiv ist
2 if (config.cluster && cluster.isMaster) {
3   // Anzahl CPU-Kerne
4   var cpuCount = require("os").cpus().length,
5       i;
6
7   // Worker starten
8   for (i = 0; i < cpuCount; i++) {
9     cluster.fork();
10  }
11
12  console.log("Tracking server started, using " + _.size(cluster.workers) + " workers.");
13
14  // Worker neu starten, wenn abgestuerzt
15  cluster.on("exit", function(worker) {
16    console.log("Worker #" + worker.id + " crashed. Restarting...");
17    cluster.fork();
18  });
19
20 } else {
21   // Starte eine Server-Instanz als Worker-Prozess
22 }
```

Listing 5.10: Starten von Master- und Worker-Prozessen

Horizontale Skalierung

Für eine horizontale Skalierung müssen mehrere Node.js Cluster auf verschiedenen Servern gestartet werden. Damit diese auch Anfragen erhalten, muss davor ein nginx-Server geschaltet werden. Dafür wird auf einem Server nginx installiert und mittels einer einfachen Konfigurations-Datei die Anfragen verteilt. [ngi]

```
1 http {
2   upstream mousetracking {
3     server srv1.mousetracking.de;
4     server srv2.mousetracking.de;
5     server srv3.mousetracking.de;
6   }
7
8   server {
9     listen 80;
10
11     location / {
12       proxy_pass http://mousetracking;
13     }
14   }
15 }
```

Listing 5.11: nginx Beispiel-Konfiguration

Auf allen drei Servern muss dabei das Portal gestartet sein.

5.3.2 Datenbank

Für die Datenbank kann ebenfalls eine horizontale Skalierung vorgenommen werden. Dafür müssen verschiedene Instanzen von MongoDB gestartet werden. Es werden mindestens zwei Routing-Server, drei Config-Server und zwei oder mehr Shards benötigt. Zunächst werden die Config-Server und die Query-Server gestartet.

```
1 # config server
2 mongod --configsvr --dbpath /data/configdb --port 27019
3
4 # query server
5 mongos --configdb cfg0.mousetracking.de:27019,cfg1.mousetracking.de:27019,cfg2.mousetracking.de
   :27019
```

Listing 5.12: Starten eines MongoDB Clusters

Nun können auf den Query-Servern Shards hinzugefügt werden. Dafür wird folgender Code auf den Query-Servern ausgeführt. Die Query-Server leiten die Anfragen entsprechend weiter. [Mona] Auf diese Weise lässt sich die Performance der Datenbank bei hoher Auslastung verbessern und nahezu unendlich nach oben skalieren. Diese Skalierung ist nicht mit der entwickelten Anwendung getestet, da die erforderlichen Ressourcen in der Testumgebung nicht gegeben sind. Allerdings ist diese nicht von der Anwendung abhängig, weshalb eine Skalierung mit der Anwendung keine Probleme verursachen sollte (sofern Datenbank richtig initialisiert wurde).

```
1 mongo --host mongos0.mousetracking.de --port 27017
```

Listing 5.13: Hinzufügen von MongoDB Shards

Kapitel 6

Test und Evaluation

Um die Funktionalität der Webanwendung zu gewährleisten, müssen Tests mit der Software durchgeführt werden. Dabei gibt es verschiedene Arten von Tests, welche an unterschiedlichen Punkten in der Entwicklung durchgeführt werden.

Unittests

Bei der Durchführung von Unittests werden kleine Programmteile durch Testcode auf ihre Funktionalität überprüft. Es wird ein zu erwartendes Ergebnis angegeben, welches mit der Ausgabe vom Programmcode verglichen wird. Stimmen die beiden Werte nicht überein, ist der Code höchstwahrscheinlich fehlerhaft oder falsch implementiert und der Test gibt eine Fehlermeldung zurück. Die Unittests werden als White-Box-Tests entwickelt. Dies bedeutet, dass der Programm-Code bekannt ist und die Tests entsprechend entwickelt werden.

Die Webanwendung wird mit dem Test-driven Development (TDD) Prinzip entwickelt. Das heißt, als erster Schritt wird ein Test geschrieben, welcher zunächst fehlschlägt, da die Funktion noch nicht implementiert ist. Im folgenden Schritt wird dieser implementiert und der Test wird erneut ausgeführt. Schlägt dieser fehl wird der Code angepasst und der Test wiederholt. Diese Schritte werden wiederholt, bis der Test erfolgreich ist. Auch bei nachträglichen Änderungen am Code werden Fehler sofort durch den bestehenden Test aufgedeckt und können direkt behoben werden. [Wam13]

Durch Unittests werden zwei verschiedene Teile der Software getestet: Zum einen wird die serverseitige Implementierung des Portals getestet. Dafür werden die Tests über die Konsole gestartet und die Ergebnisse direkt in dieser ausgegeben. Auf der anderen Seite wird das Portal und das Tracking-Script clientseitig durch Unittests geprüft. Das heißt, die Tests werden in einem Browser gestartet und die Ergebnisse ebenfalls in diesem ausgegeben. Dies ist notwendig, da die Scripte die jeweiligen Laufzeitumgebungen (Node.js Server, Browser) zum Laufen benötigen.

Diese Methode funktioniert für die kleinen Code-Einheiten recht zuverlässig, allerdings heißt dies noch nicht, dass die Anwendung als Ganzes fehlerfrei funktioniert. Um dies zu garantieren, werden weitere Tests benötigt.

Integrations- und Systemtests

Zusätzlich zu dem Testen der einzelnen Module muss auch das Zusammenspiel und die Kommunikation zwischen den Komponenten geprüft werden. Dafür werden Integrationstests durchgeführt. Es werden zwei Komponenten ausgewählt und untersucht, ob diese zusammen funktionieren. Ist dies der Fall werden nach und nach weitere Komponenten hinzugenommen und die Tests wiederholt.

Als letzte Überprüfung werden Systemtests durchgeführt. Diese sollen möglichst realitätsnah sein. Daher werden hier ausschließlich Black-Box-Tests durchgeführt. Das heißt, die Tests werden ohne Kenntnisse des Programmcodes geschrieben. Das System empfängt dabei Nutzereingaben und liefert Systemantworten zurück. [Mit] Durch die Tests können Fehler, welche durch die Bedienung der GUI auftreten, entdeckt und gegebenenfalls reproduziert werden.

Da die Anwendung nur über wenige Komponenten verfügt, werden die Integrations- und Systemtests zusammengefasst.

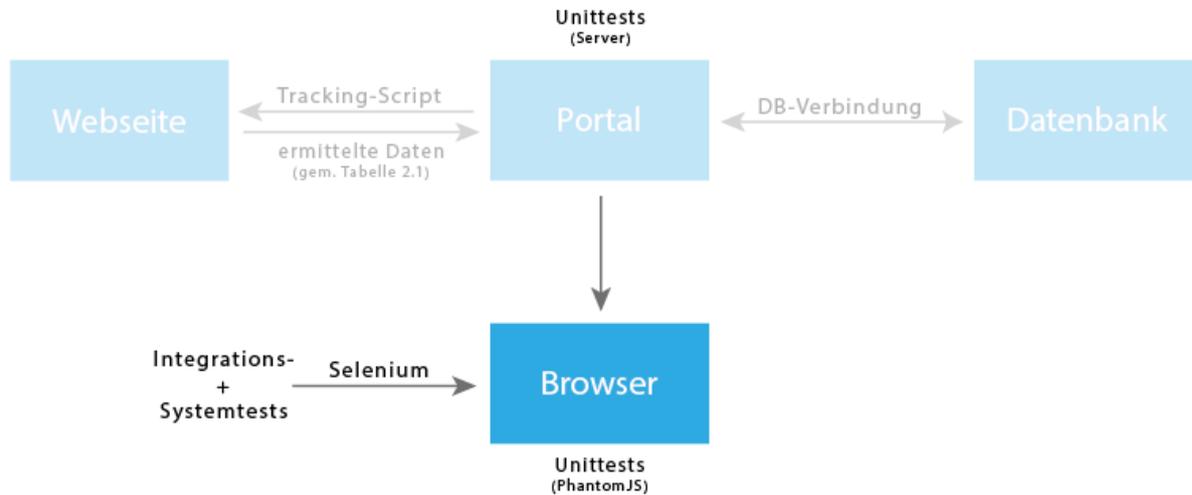


Abbildung 6.1: Veranschaulichung der Teststruktur

6.1 Unittests

Die Unittests werden in serverseitige und clientseitige Tests unterschieden und haben das Kürzel UTs, bzw. UTc. Getestet wird dabei der serverseitige Code direkt auf dem Server, sowie der clientseitige Code in einer Browser Laufzeitumgebung. Die Tests werden den entsprechenden Anforderungen zugeordnet.

Nummer	Beschreibung	Anforderung
UTs1	Konfiguration ist vorhanden	Req4, Req5
UTs2	Verbindet mit der Datenbank	Req4
UTs3	Model für Events ist definiert	Req3, Req4, Req5
UTs4	GET / gibt HTML zurück	Req12
UTs5	GET /site reagiert auf Seitenparameter	Req8, Req12
UTs6	GET /collect gibt ein Bild zurück	Req3, Req4
UTs7	GET /collect verarbeitet Daten korrekt	Req3, Req4
UTs8	GET /api/heatmap gibt komprimierte Daten zurück	Req5
UTs9	GET /api/history gibt komprimierte Daten zurück	Req5
UTc1	Tracking-Script wird initialisiert	Req1, Req2
UTc2	Mausevent-Daten werden versendet	Req3
UTc3	Tracking-Script wird von der Seite entfernt	Req1, Req2
UTc4	GUI wird initialisiert	Req6, Req7, Req8, Req9, Req10
UTc5	GUI-Elemente werden aktualisiert	Req5
UTc6	Blendet Ladeanimation ein/aus	Req5, Req11
UTc7	Liefert die Base URL der API zurück	Req5
UTc8	Liefert das Konfigurations-Objekt zurück	Req5, Req11
UTc9	Setzt einen oder mehr Werte in der Konfiguration	Req5, Req11
UTc10	Liefert das Parameter-Objekt zurück	Req5, Req6, Req7, Req8, Req9, Req10
UTc11	Setzt einen oder mehr Werte im Parameter-Objekt	Req5, Req6, Req7, Req8, Req9, Req10
UTc12	URL zur Abfrage von Daten wird generiert	Req5
UTc13	Portal wird initialisiert	Req11, Req12
UTc14	GUI und Visualisierungen werden aktualisiert	Req5, Req11
UTc15	Ladestatus ist abrufbar	Req5
UTc16	Heatmap wird initialisiert	Req5, Req11
UTc17	Heatmap wird aktualisiert	Req5, Req11
UTc18	Heatmap wird erstellt	Req5, Req11
UTc19	Heatmap wird entfernt	Req5, Req11
UTc20	Diagramm wird initialisiert	Req5, Req11
UTc21	Diagramm wird aktualisiert	Req5, Req11

Tabelle 6.1: Auflistung der Unittests

6.1.1 Server

Für die folgenden Unittests wurde das Javascript Test Framework Mocha¹ als Node.js Plugin in Verbindung mit Chai² verwendet. Die Tests werden dabei im Test Driven Development Style geschrieben. Das heißt, die Tests werden erstellt, bevor der zu testende Code geschrieben wird. Um die Tests zu starten, wird in der Konsole Mocha aufgerufen und die Art der Berichterstattung angegeben.

```
1 node_modules/.bin/mocha --reporter spec
```

Listing 6.1: Starten der serverseitigen Unittests

Die Tests werden automatisch durchlaufen und in der Konsole aufgelistet. Ein Indikator zeigt an, ob der jeweilige Test erfolgreich war. Folgend exemplarisch ein mit Mocha und Chai durchgeführter Unittest. Es wird geprüft, ob mittels mongoose eine Verbindung zur Datenbank hergestellt werden kann und kein Fehler auftritt. Ist der Test durchlaufen, wird die Datenbankverbindung wieder geschlossen.

¹<http://mochajs.org>

²<http://chaijs.com>

```
1 // Modul Server
2 describe("Server", function() {
3   after(function(done) {
4     // Verbindung zu MongoDB schliessen
5     mongoose.disconnect();
6     done();
7   });
8
9   it("connects to MongoDB", function(done) {
10    // zu MongoDB verbinden
11    mongoose.connect("mongodb://" + config.mongodb.host + "/" + config.mongodb.database,
12      function(err) {
13        assert.isUndefined(err);
14        done();
15      });
16 });
```

Listing 6.2: Unittest auf Serverseite

Für die Ausführung der Tests muss das Portal auf dem Server gestartet werden.

6.1.2 Client

Um die clientseitigen Unittests auszuführen müssen, diese in einem Browser aufgerufen werden, da der Code nur in einer Browserumgebung funktioniert. Um nicht manuell eine Seite aufrufen und die Ergebnisse kontrollieren zu müssen, wird hier ebenfalls ein headless Browser mit PhantomJS verwendet. Die Tests selber werden mit Mocha und Chai durchgeführt und mit Hilfe des Node.js Pakets `mocha-phantomjs`³ in einer PhantomJS-Instanz durchlaufen. Die Ergebnisse werden in der Konsole ausgegeben.

```
1 // Modul Heatmap
2 describe("Heatmap", function() {
3   // Funktion init()
4   describe("#init()", function() {
5     it("is a function", function() {
6       assert.isFunction(Heatmap.init);
7     });
8   });
9
10  // Funktion update()
11  describe("#update()", function() {
12    it("is a function", function() {
13      assert.isFunction(Heatmap.update);
14    });
15
16    it("returns a jQuery promise", function() {
17      var promise = Heatmap.update();
18      assert.isFunction(promise.then);
19    });
20  });
21
22  // Funktion create()
23  describe("#create()", function() {
24    it("is a function", function() {
25      assert.isFunction(Heatmap.create);
26    });
27  });
28
```

³<https://www.npmjs.com/package/mocha-phantomjs>

```

29 // Funktion remove()
30 describe("#remove()", function() {
31     it("is a function", function() {
32         assert.isFunction(Heatmap.remove);
33     });
34 });
35 });

```

Listing 6.3: Unittest auf Clientseite

Die Tests können lokal durchgeführt werden, das Portal muss dabei aber auf dem Server gestartet werden.

6.2 Integrations- und Systemtests

Mit Unittests können die meisten Fehler in den einzelnen Komponenten aufgedeckt werden. Allerdings bedeutet dies noch nicht, dass die Anwendung damit fehlerfrei ist. Es muss zusätzlich getestet werden, ob die Komponenten korrekt untereinander kommunizieren und die Anwendung als Ganzes funktioniert. Das Portal muss auf dem Server gestartet werden. Die Tests können dabei lokal durchgeführt werden. Hier wird ebenfalls Mocha in Verbindung mit Chai als Testframework verwendet. Anders als bei den Unittests wird mit der Software Selenium ein Browser gestartet und bestimmte Schritte automatisiert durchgespielt. So lassen sich eventuelle Fehler erkennen und reproduzieren.

```

1 test.describe("Client", function(done) {
2     // Timeout erhoehen (Da jedesmal der Browser gestartet werden muss)
3     this.timeout(10000);
4
5     // Driver erstellen
6     test.before(function(done) {
7         driver = new webdriver.Builder().usingServer().withCapabilities({ "driverName": "chrome", "
8             browserName": "chromium" }).forBrowser("chrome").build();
9         var handle = driver.getWindowHandle();
10        done();
11    });
12
13    // Driver beenden
14    test.after(function(done) {
15        driver.quit();
16        done();
17    });
18
19    test.it("script is added to dom", function(done) {
20        // Seite oeffnen
21        driver.get("http://192.168.59.103:7000/");
22        // pruefen, ob Element vorhanden ist
23        driver.isElementPresent(By.css('script[src="//192.168.59.103:7623/mt.js"]')).then(function(
24            el) {
25            assert.ok(el);
26            done();
27        });
28    });
29 });

```

Listing 6.4: Systemtest für das Portal

Folgende Tabelle listet die durchgeführten Systemtests auf. Die Tests haben das Kürzel ST.

Nummer	Beschreibung	Entsprechende Anforderung
ST1	Tracking-Script laden	Req1, Req2
ST2	Ladeanimation anzeigen	Req6, Req7, Req8, Req9, Req10, Req11
ST3	Startdatum mittels Datepicker ändern	Req9
ST4	Tab wechseln	Req12
ST5	Seite auswählen	Req5, Req11
ST6	Heatmap anpassen	Req5, Req7, Req11

Tabelle 6.2: Auflistung der Systemtests

6.3 Nachweis der Anforderungen

Alle Tests konnten erfolgreich durchgeführt werden. Bei der Durchführung von lokalen Tests gab es gelegentlich längere Ladezeiten bei dem Abrufen von Google Fonts und anderen Ressource von CDNs. Dies wird im Produktivbetrieb allerdings keine Probleme darstellen und ist damit vernachlässigbar. Bei den Unittests der Client-Anwendung wird eine Warnung ausgegeben, dass es einen Fehler bei den AJAX-Requests gibt, was allerdings ebenfalls zu vernachlässigen ist, da der Unittest sich nicht mit der Antwort des Servers beschäftigt. Hierbei handelt es sich lediglich, um einen Access-Control-Allow-Origin Fehler, welcher im späteren Betrieb so nicht auftritt, da die Funktion auf dem selben Server wie das Portal aufgerufen wird.

Nummer	Anforderung	Ergebnis
UTs1	Req4, Req5	erfolgreich
UTs2	Req4	erfolgreich
UTs3	Req3, Req4, Req5	erfolgreich
UTs4	Req12	erfolgreich
UTs5	Req8, Req12	erfolgreich
UTs6	Req3, Req4	erfolgreich
UTs7	Req3, Req4	erfolgreich
UTs8	Req5	erfolgreich
UTs9	Req5	erfolgreich
UTc1	Req1, Req2	erfolgreich
UTc2	Req3	erfolgreich
UTc3	Req1, Req2	erfolgreich
UTc4	Req6, Req7, Req8, Req9, Req10	erfolgreich
UTc5	Req5	erfolgreich
UTc6	Req5, Req11	erfolgreich
UTc7	Req5	erfolgreich
UTc8	Req5, Req11	erfolgreich
UTc9	Req5, Req11	erfolgreich
UTc10	Req5, Req6, Req7, Req8, Req9, Req10	erfolgreich
UTc11	Req5, Req6, Req7, Req8, Req9, Req10	erfolgreich
UTc12	Req5	erfolgreich
UTc13	Req11, Req12	erfolgreich
UTc14	Req5, Req11	erfolgreich
UTc15	Req5	erfolgreich
UTc16	Req5, Req11	erfolgreich
UTc17	Req5, Req11	erfolgreich
UTc18	Req5, Req11	erfolgreich
UTc19	Req5, Req11	erfolgreich
UTc20	Req5, Req11	erfolgreich
UTc21	Req5, Req11	erfolgreich
ST1	Req1, Req2	erfolgreich
ST2	Req6, Req7, Req8, Req9, Req10, Req11	erfolgreich
ST3	Req9	erfolgreich
ST4	Req12	erfolgreich
ST5	Req5, Req11	erfolgreich
ST6	Req5, Req7, Req11	erfolgreich

Tabelle 6.3: Ergebnisse der durchgeführten Tests

Die nachfolgende Tabelle ordnet die durchgeführten Tests den Anforderungen zu. Somit ist sichergestellt, dass alle Anforderungen an die Anwendung getestet sind.

	Req1	Req2	Req3	Req4	Req5	Req6	Req7	Req8	Req9	Req10	Req11	Req12
UTs1				x	x							
UTs2				x								
UTs3			x	x	x							
UTs4												x
UTs5								x				x
UTs6			x	x								
UTs7			x	x								
UTs8					x							
UTs9					x							
UTc1	x	x										
UTc2			x									
UTc3	x	x										
UTc4						x	x	x	x	x		
UTc5					x							
UTc6					x						x	
UTc7					x							
UTc8					x						x	
UTc9					x						x	
UTc10					x	x	x	x	x	x		
UTc11					x	x	x	x	x	x		
UTc12					x							
UTc13											x	x
UTc14					x						x	
UTc15					x							
UTc16					x						x	
UTc17					x						x	
UTc18					x						x	
UTc19					x						x	
UTc20					x						x	
UTc21					x						x	
ST1	x	x										
ST2						x	x	x	x	x	x	
ST3									x			
ST4												x
ST5					x						x	
ST6					x		x				x	

Tabelle 6.4: Nachweis der Anforderungen

Kapitel 7

Zusammenfassung und Ausblick

Dieses Kapitel fasst die Arbeit abschließend zusammen und bietet einen Ausblick auf Erweiterungsmöglichkeiten, die an der Anwendung vorgenommen werden können.

7.1 Zusammenfassung

Im Zuge dieser Arbeit sollte eine erweiter- und skalierbare Infrastruktur zur Analyse von Mausdaten geschaffen und wesentliche Funktionen implementiert werden. Dafür wurde ein Tracking-Script zur Erhebung der Daten entwickelt, welches von einem Portal bereitgestellt wird. Das Portal empfängt und speichert diese Daten in einer Datenbank und stellt diese in Form von Visualisierungen zur Verfügung. Außerdem wurde für Testzwecke ein weiterer Docker Container mit einem Testclient erstellt, um die Funktion der Anwendung testen zu können. Auf dieser Grundlage ist es möglich, Daten von beliebig vielen Seiten im Portal zu sammeln und auszuwerten.

Eine besondere Herausforderung war die Generierung einer Heatmap aus den Mauskoordinaten, da die Auflösung des Browsers sich bei jedem Besucher unterscheidet. Dafür müssen mehrere Eigenschaften der Mausdaten hinzugezogen werden und die Koordinaten jedes mal neu berechnet werden. So wird gewährleistet, dass die Heatmap die Daten korrekt widerspiegelt. Außerdem stellte die Übertragung der Daten hinsichtlich der Machbarkeit und Performance eine besondere Herausforderung dar, da dies eine essentielle Funktion der Datenerfassung ist.

Zusammenfassend kann man sagen, dass es gelungen ist, eine robuste Infrastruktur zu schaffen, welche den Anforderungen gerecht wird. Im Folgenden Abschnitt werden mögliche Erweiterungen erörtert.

7.2 Ausblick

Die entwickelte Anwendung kann als erster Prototyp gesehen werden und sollte in der Form nicht produktiv eingesetzt werden. Für die produktive Nutzung sollten Funktionen wie ein Rechtesystem integriert und der Sicherheitsaspekt genauer betrachtet werden. Auf Grundlage dieser Anwendung können weitere Daten erfasst und in dem Portal sinnvoll visualisiert werden. Durch die modulare Struktur der Anwendung ist es einfach, weitere Komponenten den Bestehenden hinzuzufügen.

7.2.1 Authentifizierung und Rechtesystem

In den meisten Fällen ist es von Webseitenbetreibern nicht beabsichtigt, dass Dritte Einsicht in die Statistiken ihrer Seite haben. Um den Zugriff auf die erfassten Daten zu beschränken, ist eine Art

Rechtesystem notwendig. Für den Webseitenbetreiber ist es wahrscheinlich nicht interessant, wie die Statistiken anderer Seiten sind. Daher müssen die erfassten Daten einem Account oder ähnlichem zugeordnet werden, welcher vor unberechtigten Zugriffen geschützt ist. Dafür wird eine entsprechende Struktur in der Datenbank, Frontend Templates und Abfragen in den Controllern benötigt.

7.2.2 Weitere Analyse der Daten

Ausgehend von den erhobenen Daten können weitere Aspekte der Nutzung visualisiert werden. Dazu könnte beispielsweise das Verhältnis von Mausbewegungen und Mausclicks an einem Punkt, oder die genauere Betrachtung einzelner Sessions gehören. Durch das kachelartige Design können weitere Cards mit Visualisierungen den Bestehenden hinzugefügt werden. Mit den bereits erhobenen Eigenschaften können ebenfalls noch andere Zusammenhänge untersucht werden. Dafür müssen relevante Zusammenhänge für die Optimierung untersucht werden.

7.2.3 Bessere Aufschlüsselung der Endgeräte

Das Portal bietet die Möglichkeit, die Daten nach den drei Endgerät-Typen Desktop, Tablet und Mobile zu filtern. Dies ist ein guter Anfang, Besucher zu differenzieren. In der Praxis müsste dies individuell auf die Seite und eventuelle Breakpoints zugeschnitten werden. Momentan ist es lediglich möglich global die Auflösungen der drei Kategorien festzulegen. Durch das Hinzufügen von Optionen für die einzelnen getrackten Seiten kann dieses Problem besser angegangen werden, um eine optimale Analyse zu gewährleisten.

Anhang A

Testprotokolle

Testprotokoll

Testnummer	ST1
Beschreibung / erwartetes Ergebnis	Das Tracking-Script wurde der Testseite hinzugefügt und alle relevanten Mausdaten werden erfasst.
Anforderung	Req1, Req2
Ausgangssituation	Das Portal ist auf dem Server gestartet.
Testverlauf	Die Seite wird aufgerufen und dem DOM-Tree wird ein Script-Element hinzugefügt.
Ergebnis	Das Tracking-Script wurde dem DOM-Tree hinzugefügt und geladen.

Testprotokoll

Testnummer	ST2
Beschreibung / erwartetes Ergebnis	Wird ein Parameter über die Bedienungselemente geändert blendet sich eine Ladeanimation ein und die Visualisierungen aktualisieren sich.
Anforderung	Req6, Req7, Req8, Req9, Req10, Req11
Ausgangssituation	Das Portal ist auf dem Server gestartet. Es liegen Daten für eine Seite vor. Eine Seite ist ausgewählt.
Testverlauf	Die Seite wird aufgerufen. Durch Klick auf ein Bedienungselement wird ein Parameter angepasst. Die Ladeanimation blendet sich ein.
Ergebnis	Die Ladeanimation blendet sich wie erwartet ein und die Visualisierungen aktualisieren sich.

Testprotokoll

Testnummer	ST3
Beschreibung / erwartetes Ergebnis	Bei Klick auf ein Datumsfeld wird ein Datepicker angezeigt.
Anforderung	Req9
Ausgangssituation	Das Portal ist auf dem Server gestartet. Es liegen Daten für eine Seite vor. Eine Seite ist ausgewählt.
Testverlauf	Die Seite wird aufgerufen. Es wird auf ein Datumsfeld geklickt. Ein Datepicker zur Auswahl des Datums erscheint.
Ergebnis	Der Datepicker wird angezeigt.

Testprotokoll

Testnummer	ST4
Beschreibung / erwartetes Ergebnis	Wird auf eine Tab-Schaltfläche geklickt ändert sich der angezeigte Inhalt.
Anforderung	Req12
Ausgangssituation	Das Portal ist auf dem Server gestartet.
Testverlauf	Die Seite wird aufgerufen. Mit einem Klick auf die Tab-Schaltfläche ändert sich der sichtbare Inhalt.
Ergebnis	Der zuvor sichtbare Inhalt ist ausgeblendet und anderer Inhalt ist sichtbar.

Testprotokoll

Testnummer	ST5
Beschreibung / erwartetes Ergebnis	Bei Klick auf einen Seitenlink änder sich die Seite und die entsprechenden Daten werden geladen.
Anforderung	Req5, Req11
Ausgangssituation	Das Portal ist auf dem Server gestartet. Es liegen Daten für mindestens eine Seite vor.
Testverlauf	Die Seite wird aufgerufen. Es wird auf einen Seitenlink geklickt. Die Seite ändert sich. Die entsprechenden Daten werden geladen.
Ergebnis	Die Seite wird gewechselt und die richtigen Daten werden geladen.

Testprotokoll

Testnummer	ST6
Beschreibung / erwartetes Ergebnis	Wird ein Endgerät ausgewählt wird eine neue Heatmap mit den richtigen Daten und der richtigen Größe erstellt.
Anforderung	Req5, Req7, Req11
Ausgangssituation	Das Portal ist auf dem Server gestartet. Es liegen Daten für eine Seite vor. Eine Seite ist ausgewählt.
Testverlauf	Die Seite wird aufgerufen. Es wird auf einen nicht ausgewählten Endgerät-Button geklickt. Die Ladeanimation wird eingeblendet. Eine neue Heatmap wird erstellt.
Ergebnis	Eine neue Heatmap wurd erstellt und zeigt die korrekten Daten an.

Literaturverzeichnis

- [Che01] CHEN, Mon C.: *What can a mouse cursor tell us more?: correlation of eye/mouse movements on web browsing*. <http://dl.acm.org/citation.cfm?id=634234>. Version: 2001, Abruf: 12.03.2015
- [Gmb15] GMBH, Vertical M.: *Software-as-a-Service (SaaS)*. <http://www.gruenderszene.de/lexikon/begriffe/software-as-a-service-saas>. Version: 2015, Abruf: 20.01.2015
- [Goo14a] GOOGLE: *Google Material Design Tooltips*. <http://www.google.com/design/spec/components/tooltips.html>. Version: 2014, Abruf: 27.05.2015
- [Goo14b] GOOGLE: *Responsive Interaction*. <http://www.google.com/design/spec/animation/responsive-interaction.html#responsive-interaction-radial-action>. Version: 2014, Abruf: 30.04.2015
- [Hir13] HIRSCHAUER, Jim: *What is Node.js and why should I care?* <http://java.dzone.com/articles/what-nodejs-and-why-should-i>. Version: Februar 2013, Abruf: 10.01.2015
- [Hos11] HOSSAIN, Monsur: *Using CORS*. <http://www.html5rocks.com/en/tutorials/cors/>. Version: 2011, Abruf: 13.03.2015
- [Köl14] KÖLN, FH: *Spaltenorientierte Datenbanksysteme*. http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/SpaltenorientierteDatenbank. Version: September 2014, Abruf: 10.05.2015
- [Led12] LEDFORD, Mauvis: *Nginx, the non-blocking model, and why Apache sucks*. <http://readystate4.com/2012/07/08/nginx-the-non-blocking-model-and-why-apache-sucks/>. Version: Juli 2012, Abruf: 16.01.2015
- [Mit] MITTELHESSEN, Technische H.: *Kurzanleitung Testen*. <https://homepages.thm.de/~hg11260/mat/testentwurf.pdf>, Abruf: 05.04.2015
- [Mod] MOD, Centmin: *Nginx vs Node.js (clustered) static file Siege Benchmarks*. <http://centminmod.com/siegebenchmarks/2013/020313/index.html>, Abruf: 19.03.2015
- [Mona] MONGODB: *Deploy a Sharded Cluster*. <http://docs.mongodb.org/manual/tutorial/deploy-shard-cluster/>, Abruf: 01.05.2015
- [Monb] MONGODB: *MongoDB Wire Protocol*. <http://docs.mongodb.org/meta-driver/latest/legacy/mongodb-wire-protocol/>, Abruf: 17.03.2015
- [Monc] MONGODB: *Sharding Introduction*. <http://docs.mongodb.org/manual/core/sharding-introduction/>, Abruf: 17.03.2015
- [ngi] NGINX: *Using nginx as HTTP load balancer*. http://nginx.org/en/docs/http/load_balancing.html, Abruf: 30.04.2015

- [Osm14] OSMANI, Addy: *Learning JavaScript Design Patterns*. <http://addyosmani.com/resources/essentialjsdesignpatterns/book/#revealingmodulepatternjavascript>. Version: 2014, Abruf: 05.03.2015
- [Rud08] RUDERMAN, Jesse: *Same-origin policy*. https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy. Version: Mai 2008, Abruf: 15.01.2015
- [Sch11] SCHIRMBACHER, RA Dr. M.: *Rechtliche Zulässigkeit von Webtracking-Tools*. <http://www.karlkratz.de/onlinemarketing-blog/webtracking-recht/>. Version: Februar 2011, Abruf: 2014
- [tec] TECHTERMS.COM: *Cluster*. <http://techterms.com/definition/cluster>, Abruf: 10.05.2015
- [Tru13] TRUYERS, Kenneth: *JavaScript Namespaces and Modules*. <http://www.kenneth-truyers.net/2013/04/27/javascript-namespaces-and-modules/>. Version: April 2013
- [Wam13] WAMBLER, Scott: *Introduction to Test Driven Development (TDD)*. <http://agiledata.org/essays/tdd.html>. Version: April 2013, Abruf: 15.01.2015
- [Wik15] WIKIPEDIA: *Document Object Model*. http://de.wikipedia.org/wiki/Document_Object_Model. Version: März 2015, Abruf: 30.04.2015
- [Zga] ZGADZAJ, Mcej: *Benchmarking Node.js - basic performance tests against Apache + PHP*. <http://zgzadzaj.com/benchmarking-nodejs-basic-performance-tests-against-apache-php>, Abruf: 10.01.2015