Technology
Arts Sciences
TH Köln

Konzeption und Implementierung einer mobilen web-basierten App zur Einschränkung der Smartphonenutzung bei Gruppenaktivitäten

Bachelorarbeit

ausgearbeitet von

Duc Giang Le

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

vorgelegt an der

Technischen Hochschule Köln
Campus Gummersbach
Fakultät für Informatik und
Ingenieurwissenschaften

im Studiengang
Medieninformatik

Erster Prüfer: Prof. Dr. Matthias Böhmer
Technische Hochschule Köln

Zweiter Prüfer: Prof. Dr. Kristian Fischer
Technische Hochschule Köln

Gummersbach, im Februar 2018

Adressen: Duc Giang Le
Kaiserstr. 8
27793 Wildeshausen
duc.giang.le@web.de

Prof. Dr. Matthias Böhmer
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
matthias.boehmer@th-koeln.de

Prof. Dr. Kristian Fischer
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
kristian.fischer@th-koeln.de

Kurzfassung

Die folgende Arbeit befasst sich mit dem Thema der *Konzeption und Implementierung einer mobilen web-basierten App zur Einschränkung der Smartphonenutzung bei Gruppenaktivitäten.* Diese Idee entstand aus der Erweiterung des Praxisprojekts, das in Zusammenarbeit mit dem Kommilitone Vu Phi Hai Dinh durchgeführt wurde. In diesem Praxisprojekt *Controlling Mobile Apps in Social Contexts* ist eine auf Android basierte Anwendung entwickelt worden, mit der die Smartphonenutzung in sozialen Kontexten kontrolliert werden kann.

In der vorliegenden Arbeit wird auf Grundlage der Ergebnisse aus dem Praxisprojekt weitere Probleme berücksichtigt, die nach dem Projekt noch bestehen blieben. Ein Problem, das nach der Fertigstellung des Praxisprojekts unter anderem vorhanden war, ist die geringe Reichweite der Kompatibilität zu verschiedenen mobilen Betriebssystemen. Durch eine mobile web-basierte Anwendung sollen neben den Smartphones mit dem Android Betriebssystem weitere Smartphones miteinbezogen werden können. Dazu muss untersucht werden, welche Möglichkeiten eine mobile web-basierte Anwendung hat, um das Ziel zur Einschränkung der Smartphonenutzung bei Gruppenaktivitäten zu erreichen. Zur Lösung der Probleme werden verwandte Arbeiten analysiert, die diese Problematik adressieren. Außerdem werden vorhandene Web-Technologien untersucht und getestet.

Auf Grundlage der Erkenntnisse können Anforderungen abgeleitet und konzeptionelle Aussagen über die zu entwickelnde Anwendung getroffen werden. Für den Abschluss dieser Arbeit wird basierend auf dem entwickelten Konzept eine mobile web-basierte Anwendung implementiert, die das Ziel hat, die Probleme der geringen Reichweite der Kompatibilität zu verschiedenen mobilen Betriebssystemen und die Smartphonenutzung bei Gruppenaktivitäten zu adressieren.

Abstract

The topic of this bachelor-thesis is the *conception and implementation of a mobile web-based application for limiting smartphone usage in group activities*. The idea of this thesis results from the practical project which is carried out in cooperation with the fellow student Vu Phi Hai Dinh. In that practical project *Controlling Mobile Apps in Social Contexts* a mobile application in Android is developed with the aim to control the smartphone usage in social contexts.
This bachelor-thesis considers the persisted problems after the completion of the practical project. One of these problems is the low range of compatibility with various mobile operating systems. With a mobile web-based application the aim is to include all available smartphones with their various mobile operating systems in order to increase the user density. The technical potential of a mobile web-based application will also be examined in order to show if it is possible for a mobile web-based application to achieve the aim of limiting smartphone usage in group activities. In order to solve the problems and to achieve the aim of this thesis related works will be analyzed. Furthermore, existing web technologies will be examined and tested.
On the basis of the gained knowledge, requirements and conceptual statements about the application to be developed can be made. To complete this bachelor-thesis, a mobile web-based application will be implemented based on the developed concept with the aim to solve the issues with the low range of compatibility with various mobile operating systems and with the smartphone usage in group activities.

Danksagung

Hier möchte ich mich an all diejenigen bedanken, die mich bei der Anfertigung dieser Arbeit motiviert und unterstützt haben.

Ein besonderer Dank gilt Herrn Prof. Dr. Matthias Böhmer, der mich bereits im Praxisprojekt und in der Bachelorarbeit als Erstprüfer betreut hat. Seine konstruktive Kritik, seine hilfreichen Denkanstöße und Anregungen und seine Betreuung, in der ich vieles gelernt habe, hat mir bei der Anfertigung der Bachelorarbeit sehr geholfen. Dafür möchte ich mich herzlich bedanken.

Außerdem möchte ich mich bei Herrn Prof. Dr. Kristian Fischer herzlich bedanken, dass er sich für die Rolle des Zweitprüfers dieser Arbeit zur Verfügung gestellt hat.

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung

In den letzten Jahren ist die Smartphonenutzung in Deutschland stark angestiegen. Waren es im Januar 2009 noch ca. 6,31 Millionen Nutzer, sind es laut einer Statistik von comScore (2016) April 2016 ca. 49 Millionen Nutzer. Folglich benutzt fast mehr als jede zweite Person in Deutschland ein Smartphone. Es ist in der heutigen Gesellschaft nicht mehr wegzudenken. Durch seine Mobilität und Vielfalt von Funktionen ist es vielen Situationen unverzichtbar, sei es zu Hause, bei der Arbeit oder auch unter Freunden. Es bietet den Menschen die Möglichkeit, standortunabhängig zu kommunizieren und in den meisten Fällen mit dem Internet verbunden zu sein, so dass man ständig erreichbar ist. Auch wenn das Smartphone dem Menschen viele Vorteile bietet und seinen Alltag auf vielfältige Weise erleichtern kann, bringt die Smartphonenutzung nicht stets etwas Positives mit sich.
Durch den ständigen Zugang zum Internet neigen die Menschen dazu, öfters auf das Smartphone zu schauen, um die Neuigkeiten in der Welt zu erfahren. Des Weiteren lenkt jede neue Benachrichtigung, die der Mensch wahrnimmt, die Aufmerksamkeit des Menschen auf das Smartphone. Oft lassen sie sich durch die beiden genannten Fakten so stark ablenken, dass sie ihre Umgebung für den Moment kaum noch wahrnehmen. Geschieht dieses Phänomen in Anwesenheit von anderen Menschen, mit denen man sich im Moment trifft, spricht man von Phubbing. Es bezeichnet die Angewohnheit in Gegenwart anderer Personen, sich mehr mit dem Smartphone zu beschäftigen und seine Mitmenschen dadurch zu vernachlässigen. Diese Angewohnheit lässt sich heutzutage in vielen Situationen beobachten, z.B. bei einem Treffen in einem Café, in einer Lerngruppe oder in einem Restaurant.
Im Zuge des Praxisprojekts, das vor dieser Arbeit durchgeführt wurde, ist mit der Entwicklung einer Android-Anwendung versucht worden, die Smartphonenutzung in sozialen Kontexten zu kontrollieren und so das Phubbing zu reduzieren. Die Anwendung hat den Nutzern dabei ermöglicht, bestimmte Anwendungen auf dem Smartphone des Freundes zu blockieren, so lange sich die Freunde treffen. Ein Problem, das nach Durchführung des Praxisprojektes bestehen blieb, ist die geringe Reichweite der Kompatibilität zu verschiedenen mobilen Betriebssystemen. Die Anwendung funktionierte bisher nur auf Smartphones mit dem Betriebssystem Android. Dadurch war es Nutzern anderer Betriebssysteme nicht möglich, die Anwendung einzusetzen. Sie würden bei Treffen unter Freunden weiterhin ihr Smartphone nutzen können. Laut einer Statistik von Kantar (2018) befand sich der Marktanteil von Android zum November 2017 bei 78% und der restlichen Betriebssysteme bei 22%, was ca. ein Fünftel der Smartphonenutzer dargestellt. Daher ist es wichtig, diesen Anteil der anderen Betriebssysteme für das Einschränken der Smartphonenutzung bei Gruppenaktivitäten miteinzubeziehen.

1.2 Zielsetzung

Diese Arbeit verfolgt das Ziel, die in der Problemstellung geschilderten Probleme zu lösen. Dazu zählen zum einen die Smartphonenutzung bei Gruppenaktivitäten und zum anderen die geringe Kompatibilität aller mobilen Betriebssysteme. Um die Ziele dieser Arbeit genauer erleuchten zu können, wird im Folgenden eine Zielhierarchie dargestellt, die die bedeutsamen Ziele in strategische, taktische und operative Ziele unterteilt. Die einzelnen Ziele werden mithilfe der Verben „muss, soll und kann“ nach ihrer Wichtigkeit für diese Arbeit priorisiert.

strategische Ziele:

- Bei Gruppenaktivitäten muss die Smartphonenutzung unabhängig vom dem Smartphone und seinem Betriebssystem einschränkt werden.

taktische Ziele:

- Es muss eine mobile web-basierte Anwendung entwickelt werden, die die Smartphonenutzung des Nutzers bei Gruppenaktivitäten einschränkt.
- Es soll eine mobile web-basierte Anwendung entwickelt werden, die auf allen mobilen Internetbrowsern genutzt werden kann.

operative Ziele:

- Es muss ein Prototyp entwickelt werden, der die zu entwickelnde Anwendung repräsentiert.
- Es müssen für die mobile web-basierte Anwendung spezifische Technologien untersucht und getestet werden, um die Realisierbarkeit der geplanten Anwendung zu prüfen.
- Es sollen verwandte Arbeiten und Produkte untersucht werden, die ganz oder teils versuchen, die in der Problemstellung geschilderten Probleme zu lösen.
- Es sollen Informationen über die Ablenkung durchs Smartphone und grundlegende Informationen über mögliche einzusetzende Technologien beschafft werden.

1.3 Struktur der Arbeit

Zu Beginn dieser Bachelorarbeit werden im zweiten Kapitel **Grundlagen** essentielle Informationen vermittelt, die für das bessere Verständnis dieser Arbeit dienen. Hierbei werden zwei wichtige Themengebiete dargestellt: Informationen über die Ablenkung durch Smartphones und grundlegende Informationen über die in der Arbeit eingesetzten Technologien.
Als Nächstes werden zum Anfang des dritten Kapitels **Konzeption** verwandte Arbeiten und Produkte analysiert. Anhand dieser Analyse können erste Anforderungen für die zu entwickelnde Anwendung ermittelt werden. Anschließend können auf Grundlage der Anforderungen konzeptionelle Aussagen über die Systemarchitektur und die geplante Implementierung gemacht werden. Hier soll die Architektur der zu entwickelnden Anwendung erst modellhaft erläutert und anschließend die einzelnen Bestandteile des Systems dargestellt werden.
Im vierten Kapitel **Implementierung** werden die wichtigsten realisierten Anwendungslogiken in der Anwendung erläutert, die zur Lösung der in der Arbeit behandelten Probleme beitragen. Hier wird die Realisierung der Anwendung detailliert an Code-Beispielen dargestellt.
Der fünfte Kapitel **Prototypische Anwendung der App** beschreibt den Ablauf der entwickelten Anwendung aus Sicht eines Nutzers anhand eines Beispiels.

2 Grundlagen

In dem folgenden Kapitel werden elementare Informationen vermittelt, die als Grundlage für das Verständnis der vorliegenden Arbeit dienen. Zum Einen wird das Themengebiet der Ablenkung durch Smartphones genauer erleuchtet und welche Probleme und Folgen es verursachen kann und zum Anderen sollen die technischen Grundlagen für das bessere Verständnis der eingesetzten Technologien beschrieben werden.

2.1 Ablenkung durch das Smartphone

2.1.1 Das Smartphone und wie es den Alltag verändert

Das Smartphone wird im Gabler Wirtschaftslexikon vom Herausgeber Sjurts (2011) wie folgt definiert:

„[Das Smartphone ist ein] Mobiltelefon mit erweitertem Funktionsumfang. Dazu zählen neben der Telefonie und Short Message Service (SMS) üblicherweise Zusatzdienste wie Electronic Mail (E-Mail), World Wide Web (WWW), Terminkalender, Navigation sowie Aufnahme und Wiedergabe audiovisueller Inhalte. Auf Smartphones laufen gegenüber herkömmlichen Mobiltelefonen komplexere Betriebssysteme wie etwa [Android OS], Blackberry OS oder das iPhone OS. Die hierdurch geschaffene Möglichkeit zur Installation weiterer Applikationen durch den Endnutzer verleiht Smartphones einen erweiterbaren und individualisierbaren Funktionsumfang.“

Es besitzt weitere Merkmale wie einen berührungsempfindlichen Bildschirm und eine virtuelle Tastatur. Aufgrund seiner Mobilität ist es stets griffbereit. Außerdem bietet das Smartphone heutzutage einen ständigen Zugang ins Internet, so dass jeder orts- und zeitunabhängig „online“ sein kann.
Wie in der Definition von Sjurts (2011) beschrieben, laufen verschiedene mobile Betriebssysteme auf Smartphones. Sie stellen die Basis zur Entwicklung von mobilen Anwendungen dar. Zu den meist genutzten mobilen Betriebssysteme in Deutschland gehören laut einer Statistik von Kantar (2018) Android OS von Google und iOS von Apple. Das Smartphone ist in der heutigen Gesellschaft fest in unseren Alltag integriert. Es verändert unser Leben auf positive, aber auch auf negative Weise.

Aufgrund des großen Funktionsumfangs eines Smartphones ist man stets mit dem Smartphone beschäftigt. Es lassen sich Fotos schießen, Musik hören, Filme anschauen oder Spiele spielen. Mit dem Smartphone sind die Menschen wegen der ständigen Verbindung zum Internet stets und überall erreichbar und können auf schnellste Weise mit anderen Menschen kommunizieren. Diese Kommunikation über elektronischem Wege ist kritisch zu betrachten. Die US-Soziologin Sherry Tuckle beschreibt in einem Interview ihre Sichtweise zum Wandel der Kommunikation aufgrund von Smartphones:

„Es gibt zunehmend weniger echte Unterhaltungen, dafür E-Mails, SMS, Messenger-Nachrichten: Statt direkt miteinander zu sprechen, wird mehr schriftlich kommuniziert. Ich ruf dich nicht an, ich schick dir nur eine Textnachricht. Das ist ein klar erkennbarer Wandel, und der ermöglicht es, menschlichen Kontakt zu reduzieren und sich ganz ungeniert vor sozialen Situationen zu verstecken.[...] [Man] [...] entscheide[t] [allein], wem oder was [man] wann Aufmerksamkeit schenke [und] [...] wann wir uns für soziale Situationen interessieren.“ (Schulz (2012))

2.1.2 Folgen auf das Individuum

Durch die häufige Nutzung des Smartphones ist man dazu geneigt, seine Aufmerksamkeit auf das Smartphone zu richten und seine Umwelt nicht zu beachten. Mit dem Jugendwort des Jahres 2015 „Smombie“ wird eine Person beschrieben, die die zuvor geschilderte Eigenschaft aufweist. Das Wort Smombie setzt sich zusammen aus den Wörtern Smartphone und Zombie (o.V. (2015)). Durch den großen Funktionsumfang eines Smartphones ist man stets mit dem Gerät beschäftigt, so dass der Smartphone ein Teil des Alltags wird.
Ein Beispiel, bei der die Ablenkung durch Smartphones gravierende Folgen haben können, ist die Smartphonenutzung im Straßenverkehr. Oft ist man am Steuer dazu verleitet auf das Smartphone zu schauen, wenn man z.B. eine neue Benachrichtigung erhält. Jeder kleinste Aufmerksamkeitsmangel im Straßenverkehr kann zu Unfällen führen. Auch Fußgänger, die ihre Aufmerksamkeit auf das Smartphone richten, können durch ihre geringe Wahrnehmung der Umgebung den Straßenverkehr und sich selber gefährden.
Das Ablenkungspotential des Smartphones kann auch in anderen Bereichen des Alltags Folgen auf das Individuum haben. In der heutigen Gesellschaft ist man dazu verleitet jederzeit alles erfahren zu wollen und „up-to-date“ zu sein. Dies ist durch das Smartphone möglich geworden.

2.1.3 Folgen auf die Umwelt des Individuums

Die Ablenkung durch das Smartphone hat nicht nur Auswirkungen an die Person an sich, sondern auch an seine Umwelt. In gesellschaftlichen Treffen und Gruppenaktivitäten lässt sich aufgrund des Ablenkungspotential eines Smartphones oft beobachten, dass die Aufmerksamkeit der Personen oftmals dem Smartphone gelten. Dieses Phänomen lässt sich mit dem Begriff Phubbing beschreiben. „Dieses Wort setzt sich zusammen aus den englischen Begriffen Phone und snubbing - wobei „to snub“ übersetzt so viel bedeutet wie „jemanden abblitzen lassen“ oder „vor den Kopf stoßen““ (Weidlich (2018)). Phubbing bezeichnet die Angewohnheit in Gegenwart anderer Personen, sich mit dem Smartphone zu beschäftigen und seine Mitmenschen dadurch zu vernachlässigen. Durch den Drang stets erreichbar zu sein, wird man verleitet bei jeder neuen Benachrichtigung oder Neuigkeit auf das Smartphone zu schauen. Obwohl man sich mit anderen Personen trifft, um „echte“ Konversationen zu führen und Zeit mit ihnen zu verbringen, wird in diesen Treffen dennoch das Smartphone benutzt, um Neuigkeiten zu erfahren oder sogar mit anderen Menschen über soziale Netzwerke zu kommunizieren. Der Gegenüber fühlt sich durch das Phubbing ausgegrenzt und vernachlässigt. David u. Roberts (2017) beschreiben in ihrer Studie, dass Opfer von Phubbing durch die soziale Ausgrenzung dazu neigen, ihre Aufmerksamkeit ebenfalls auf das Smartphone zu richten, um über soziale Netzwerke die Aufmerksamkeit anderer Menschen zu gewinnen, da sie im Moment vom Gegenüber vernachlässigt werden. „Der Grund liegt in [der] angeborenen Angst, ausgeschlossen zu werden“ (Weidlich (2018)).

2.2 technische Grundlagen

2.2.1 Echtzeit-Kommunikation durch WebSockets mit dem Framework socket.io

WebSocket ist eine auf TCP basierte Technologie für das Web, die eine Echtzeit-Kommunikation zwischen einer Webanwendung und einem Webserver ermöglicht. Dadurch, dass nach dem Verbindungsaufbau zwischen Client und Server die Verbindung offen gehalten wird, kann der Server ohne Anfrage des Clients neue Informationen ausliefern oder der Client dem Server Daten senden. Man spricht von einer bidirektionalen Verbindung zwischen Webanwendung und Webserver (Rouse (2016)).

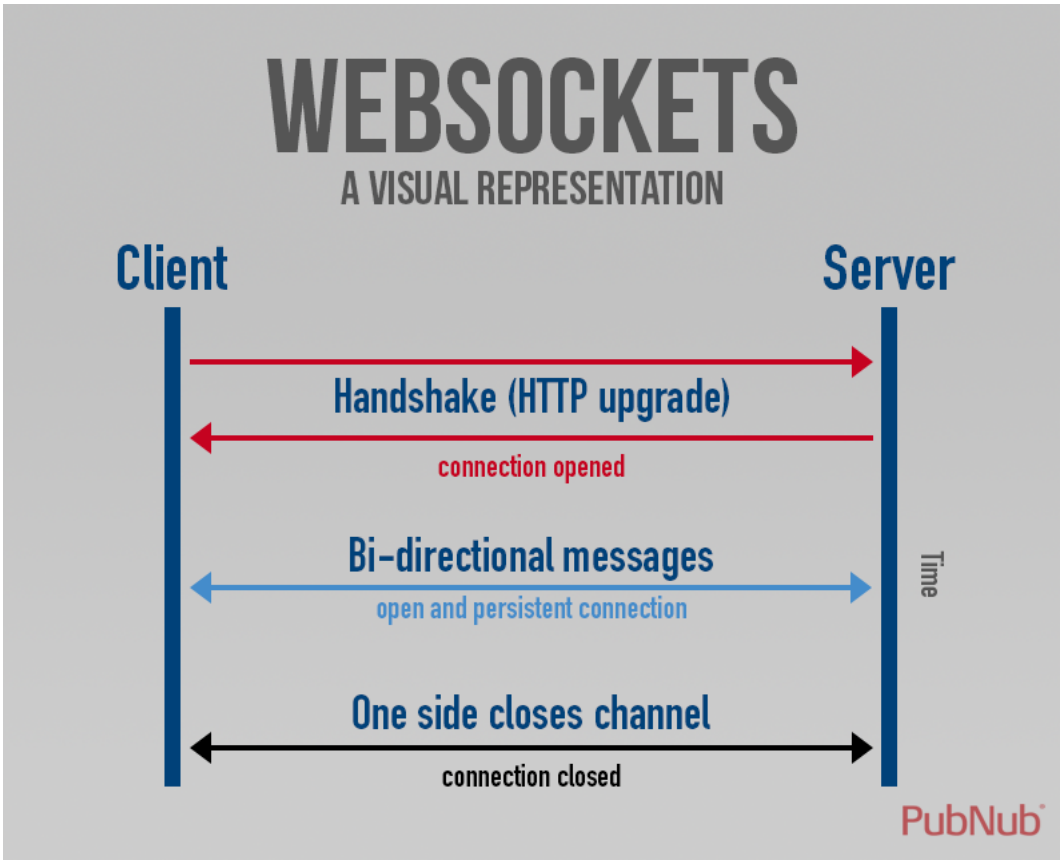


Abbildung 2.1: WebSockets - A Visual Representation (Quelle: Hanson (2013))

In der Abbildung 2.1 ist das Prinzip von WebSockets dargestellt. Im Detail sendet der Client als Erstes eine HTTP Anfrage an den Server. Diese Anfrage beinhaltet ein Handshake für einen Verbindungsaufbau zwischen Client und Server. Außerdem fordert der Client in dieser Anfrage einen Upgrade von HTTP auf das Websocket-Protokoll bzw. TCP. Nach Antwort und Bestätigung des Servers wird eine Verbindung zwischen beiden Komponenten aufgebaut, die persistent bestehen bleibt. Über diese bidirektionale Verbindung können jeweils Client oder Server Nachrichten an die andere Komponente senden. Dadurch ist es möglich, jederzeit Daten zwischen Client und Server auszutauschen und einer Webanwendung die Fähigkeit der Echtzeit-Kommunikation zu ermöglichen. Aufgrund dessen werden in einer Webanwendung neue Funktionalitäten wie Chat oder Kollaboration realisierbar. Eines der bekanntesten Frameworks, die die Technologie der WebSockets benutzt, ist socket.io.

Es ist eine auf JavaScript basierende Library für Echtzeit-Webanwendungen. Socket.io ermöglicht es auf Grundlage von WebSockets, zwischen Client und Server eine bidirektionale Verbindung aufzubauen und darüber Daten auszutauschen. Es bietet eine clientseitige Library für den Browser sowie eine serverseitige Library für node.js an, wobei beide nahezu identisch sind. Folgende Features werden von socket.io angeboten: Real-time analytics, Instant messaging and chat, Binary streaming und Document collaboration (socket.io (2015)). Die Kommunikation über socket.io erfolgt über ein ereignisgesteuertes Prinzip. Der Client oder der Server kann mithilfe der Methode emit selbstdefinierte Ereignisse auslösen und so Nachrichten zu seinem Gegenüber senden. Diese können die Ereignisse wahrnehmen, wenn sie die zuvor registriert haben, und darauf reagieren, sobald diese ausgelöst werden.

2.2.2 Der Architekturstil REST

REST ist die Abkürzung für Representational State Transfer und ist ein Architekturstil, der beschreibt, wie verteilte Systeme miteinander kommunizieren können. Im Jahre 2000 hat Roy Fielding, einer der Hauptautoren der HTTP-Spezifikation, in seiner Dissertation „Architectural Styles and the Design of Network-based Software Architectures“ (Fielding (2000)) das Konzept der REST-Architektur vorgestellt. Es ist jedoch nur ein Konzept bzw. Architekturstil und gilt nicht als fest definierter Standard im World Wide Web. Dennoch findet man heutzutage im World Wide Web viele verteilte Systeme, die nach dem REST-Architekturstil entwickelt worden sind. Eines der größten Vorteile, die REST bietet, ist die hohe Skalierbarkeit des Systems durch eine strikte Trennung von Webclient und Webserver. Dadurch ist es möglich, ohne großen Aufwand die einzelnen Bestandteile des Systems zu erweitern. Für das bessere Verständnis des Architekturstils muss das Konzept der Ressourcen beschrieben werden, die eine der Kernideen von REST darstellt. Roy Fielding beschreibt den Begriff Ressource wie folgt: „The key abstraction of information in REST is a resource. Any information that can be named can be a resource [...]“ (Fielding, 2000, Abschnitt 5.2.1.1) In einer REST-Architektur werden somit alle vorkommenden Entitäten bzw. Informationen, die sich benennen lassen können, als eine Ressource bezeichnet. Damit Webanwendungen als eine REST-konforme Anwendung gelten, müssen sie bestimmte REST-Prinzipien einhalten, die im Folgenden beschrieben werden:

- **Adressierbarkeit:** Jede Ressource ist über einen eindeutigen Uniform Ressource Identifier (URI) identifizierbar. Beispielsweise lässt sich ein Nutzer mit der Nutzer-ID 12345 in einem REST-konformen Webservice mithilfe der URI *http://example.com/users/12345* adressieren.
- **Hypermedia:** Die einzelnen Ressourcen sind durch Einsatz von Hypermedia miteinander verlinkt. Dadurch ist es dem Client möglich, ausschließlich über den Uniform Ressource Locator (URL) auf andere Ressourcen zu navigieren.

- **allgemeingültige Operatoren:** In einer REST-Architektur werden auf Ressourcen stets über einen einheitlichen Satz von allgemeingültigen Operatoren bzw. Standardmethoden zugegriffen. Da es in einem REST-konformen Webservice üblich ist, das Transportprotokoll HTTP zu benutzen, greift man auf die Standard-HTTP-Methoden GET, POST, PUT, DELETE usw. zurück.

- **Repräsentationen:** Die Ressourcen können verschiedene Repräsentationen haben. Dazu zählen die Darstellung in verschiedenen Sprachen oder Formate (HTML, JSON oder XML) und die Darstellung für verschiedene Endgeräte (PC, Smartphone oder Tablet usw.). Je nach Anfrage eines Client kann der Webserver z.B. eine Ressource im HTML- oder auch im JSON-Format ausliefern.

- **Zustandslosigkeit:** Dieses wichtige Prinzip von REST besagt, dass die Kommunikation zwischen Client und Webserver zustandslos erfolgen muss. Bei jeder Anfrage des Clients an den Webserver müssen jedes Mal alle erforderlichen Informationen zum Verstehen der Anfrage mitgesendet werden. Client und Webserver speichern keine Zustandsinformationen. Beispielsweise werden in REST-konformen Webservices keine mit Cookies und Sessions realisierten Benutzersitzungen eingesetzt.

2.2.3 node.js - eine Plattform zur Entwicklung von Netzwerkanwendungen

node.js ist eine serverseitige Plattform zur Entwicklung von Netzwerkanwendungen. Da es auf JavaScript basiert, können Entwickler beispielsweise für den Webclient und den Webserver die selbe Programmiersprache benutzen, die die Entwicklung eines Webservices vereinfachen. node.js besitzt eine ereignisgesteuerte Architektur, in der das Zusammenspiel der Komponenten in node.js durch Ereignisse gesteuert werden. Dies ermöglicht eine asynchrone nicht blockierende I/O, was einer der größten Vorteile von node.js ist. Sobald eine I/O-Anfrage kommt, wird diese asynchron bearbeitet und node.js wird bei Fertigstellung dieser Anfrage benachrichtigt. Dadurch kann es weiterhin I/O-Anfragen annehmen und diese im asynchronen Modus bearbeiten. So muss node.js nicht warten, bis eine I/O-Anfrage komplett fertiggestellt wird, um weitere bearbeiten zu können.
Außerdem bietet node.js sogenannte Modules an, die vergleichbar wie JavaScript Libraries sind. Sie stellen weitere Funktionen zur Verfügung. Express.js ist einer der bekanntesten Modules. Es beschreibt sich selber als ein „[...] einfaches und flexibles Node.js-Framework [für] Webanwendungen, das zahlreiche leistungsfähige Features und Funktionen für Webanwendungen und mobile Anwendungen bereitstellt“ (Expressjs (2017)).

2.2.4 Firebase - die Entwicklungsplattform für mobile und web-basierte Anwendungen

Firebase ist eine Plattform, die für die Entwicklung von mobilen und web-basierten Anwendungen spezialisiert ist. Sie bietet verschiedene Funktionen an, die einem Entwickler und Anbieter von Anwendungen bei der Entwicklung ihrer Anwendung unterstützen. Dazu gehören unter anderem automatische Datensynchronisierung, Authentifizierungsdienste, Messaging, Dateispeicherung, Analyse usw. (Google (2018)). Firebase gehört zu den Services, die als Mobile Backend as a Service (MBaaS) bezeichnet werden kann (David (2016)). In manchen Fällen, in denen serverseitig kaum bis keine Anwendungslogik benötigt wird, kann Firebase einen Server komplett ersetzen und als Backend eingesetzt werden.
In der Arbeit wird die Firebase Authentication API für Authentifizierungszwecke genutzt. Es bietet Authentifizierungsmöglichkeiten über E-Mail, soziale Netzwerke wie Facebook oder Twitter und seit neuerem über die Handynummer an (Firebase (2018)).

2.2.5 Heroku - eine Plattform zur Verteilung von Webanwendungen

Heroku ist eine Cloud Platform as a Service, die zur Verteilung von Webanwendungen genutzt wird. Der Entwickler hat die Möglichkeit, die fertiggestellte Webanwendung auf die Heroku-Server hochzuladen und über Heroku die Webanwendung zu verteilen. Andere Personen können durch die von Heroku generierte URL auf die Webanwendung zugreifen. In dieser Arbeit wird Heroku ebenfalls genutzt, um die zu entwickelnde Anwendung im Web zu verteilen und dadurch mobilen Endgeräten den Zugang zur Webanwendung zu ermöglichen.

2.2.6 In der Anwendung eingesetzte Libraries

Zu den bereits beschriebenen Technologien und Plattformen werden zur Entwicklung der Webanwendung weitere Libraries genutzt. Diese sollen im Folgenden aufgelistet und genauer betrachtet werden:

- **NoSleep.js:** NoSleep.js ist eine auf JavaScript basierende Library mit dem Ziel, das automatische Ausschalten des Bildschirms eines mobilen Endgerätes zu verhindern und den Wakelock in den mobilen Browsern zu aktivieren (Tibbett (2015)). Ein Wakelock ist ein Mechanismus, der es mobilen Anwendungen erlaubt, das automatische Ausschalten des Bildschirms bei mobilen Geräten zu verhindern. Es kann dadurch nicht in den Standby-Modus umschalten.

- **QRious:** QRious ist eine JavaScript Library, die es Anwendungen ermöglicht, eigene QR-Codes zu generieren (Mercer (2017)). In der Anwendung kann bestimmt werden, welche Daten als QR-Code verschlüsselt werden sollen. Üblicherweise werden URLs als QR-Code verschlüsselt.

- **BootStrap:** Bootstrap ist ein Open-Source Frontend-Framework, das mithilfe von HTML, CSS und JavaScript bei der Entwicklung eines Frontends unterstützt. Es stellt auf HTML und CSS basierende Gestaltungsvorlagen bereit, die ins eigene Frontend integriert werden können. Außerdem unterstützt das Framework das gestalterische Paradigma Responsive Design, wodurch das Frontend auf die Eigenschaften des benutzten Endgeräts reagieren und sich anpassen kann.

3 Konzeption

In diesem Kapitel sind zum Anfang verwandte Arbeiten und Produkte auf dem Markt untersucht worden. Diese Analyse soll Erkenntnisse darüber geben, welche technische Möglichkeiten bereits zur Realisierung der geplanten Anwendung vorhanden sind und welche Stärken und Schwächen die einzelnen Anwendungen aufweisen. Anhand der Untersuchung der Produkte sollen außerdem Anforderungen an das zu entwickelnde System abgeleitet werden.
Aus allen ermittelten Anforderungen können konzeptionelle Aussagen über die Architektur und über die Realisierung der zu entwickelnden Anwendung gemacht werden.

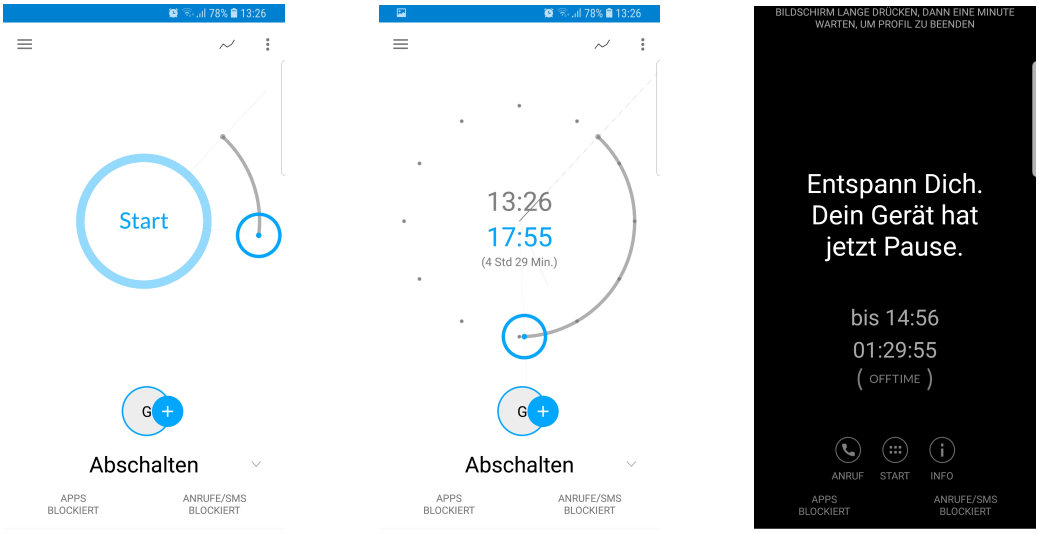
3.1 verwandte Arbeiten und Produkte auf dem Markt

In den folgenden Abschnitten sollen vorhandene Arbeiten und Produkte auf dem Markt, die versuchen die geschilderten Probleme in der Problemstellung teils oder ganz zu lösen, betrachtet und analysiert werden. Dabei soll die Funktionsweise der Anwendungen erläutert und die Stärken und Schwächen einzelner Produkte untersucht und im Hinblick auf die Problemstellung kritisch geprüft werden.

3.1.1 offtime - Focus & Digital Balance

OffTime ist eine mobile Anwendung entwickelt von einem Start-up Unternehmen aus Berlin. Während für Android eine Vollversion existiert, wird für iOS aufgrund der Unterschiede in den Betriebssystemen eine light-Version angeboten. Das Ziel dieser mobilen Anwendung ist es, die Smartphonenutzung eines Nutzers zu kontrollieren und einzuschränken.
Es erlaubt dem Nutzer, einzelne Anwendungen für einen selbst definierten Zeitraum zu blockieren und dadurch nicht nutzen zu können. Außerdem bietet es eine Funktion an, die das gesamte Smartphone für einen selbst festgelegten Zeitraum komplett blockiert. Selbst Anrufe und Benachrichtigungen können von der Anwendung verhindert werden, so dass in der Laufzeit der Anwendung auf dem Smartphone keine Benachrichtigungen angezeigt werden. Erst nachdem das Smartphone wieder freigeschaltet ist, werden alle verpassenden Benachrichtigungen und Anrufe bekanntgegeben.

Einer der größten Stärken von offTime ist die Fähigkeit, die Smartphonenutzung eines Nutzers durch das Blockieren sämtlicher Apps, Funktionen und Benachrichtigungen auf dem Smartphone einzuschränken. Jedoch ist dies nur auf dem mobilen Betriebssystem Android möglich, wodurch Nutzer anderer Betriebssysteme keine Möglichkeit haben, diese Anwendung nutzen zu können. Außerdem werden für die vollständige Funktionsfähigkeit der Anwendung viele Berechtigungen vom Nutzer verlangt, womit die Anwendung Zugriff auf persönliche und sensible Daten des Nutzers hat. In der Abbildung 3.1 werden einzelne Screenshots von offTime dargestellt.



(a) Home-Activity der App

(b) Zeit, bis wann blockiert werden soll

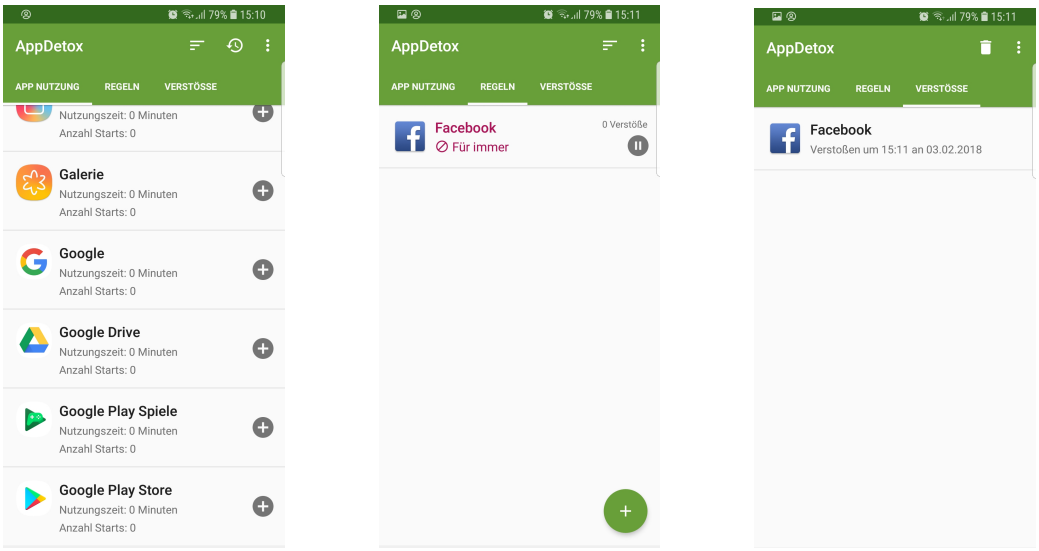
(c) Smartphone im blockierten Zustand

Abbildung 3.1: Screenshots von der Anwendung offTime

3.1.2 AppDetox - eine mobile Anwendung gegen Smartphone-Sucht

AppDetox ist eine mobile Anwendung in Android, die von Markus Löchtefeld, Matthias Böhmer und Lyubomir Ganev entwickelt worden ist. Sie beschreiben ihre Anwendung als „eine App, die es Nutzern erlaubt, bewusst Regeln zu erstellen, die sie davon abhalten, bestimmte Apps zu benutzen“ ((Löchtefeld u. a., 2013, S.1)). Diese mobile Anwendung funktioniert wie folgt:

Als Erstes stellt die App alle installierten Anwendung auf dem Smartphone dar. Anschließend kann der Nutzer für eine ausgesuchte App, Regel aufstellen wie z.B. wie oft er die App starten oder wie lange er die App nutzen darf. Sollte der Nutzer diese Regeln verstoßen, wird dies in AppDetox notiert. Außerdem zeichnet die App nebenbei noch auf, wie lange welche Apps genutzt wurden und wie oft diese Apps geöffnet worden sind. Auf technischer Ebene beobachtet die Anwendung mittels eines Background-Services, ob eine neue App geöffnet wurde. Sollte für diese App eine Regel aufgestellt sein, wird diese sofort geschlossen und der Nutzer benachrichtigt. Ein technologisches Problem, das Löchtefeld u. a. (2013) in ihrer Arbeit erwähnen, ist, dass Benachrichtigungen von den durch die Regeln blockierten Anwendungen dennoch auf dem Smartphone angezeigt werden, was dazu führt, dass die blockierten Apps immer noch durch ihre Benachrichtigungen ablenken können.
Ein positiver Aspekt dieser Anwendung ist die Variation an Regeln, die der Nutzer aufstellen kann, um seine alltägliche Smartphonenutzung selber einzuschränken. Außerdem ermöglicht AppDetox den Zugang zu anderen Apps zu blockieren, so dass sie nicht für die Dauer des Blockierens nicht genutzt werden können. Jedoch kann die Anwendung wie bei offTime aufgrund der Verschiedenheit der mobilen Betriebssysteme nur in Android realisiert werden. Nutzer anderer Betriebssysteme können diese App nicht nutzen. Auch werden von AppDetox Berechtigungen vom Smartphone verlangt, die der Anwendung die Fähigkeit gibt, wichtige Bestandteile des Smartphones zu kontrollieren. In der Abbildung 3.2 werden die elementaren Activities der mobilen Anwendung dargestellt.



(a) Liste der installierten Anwendungen

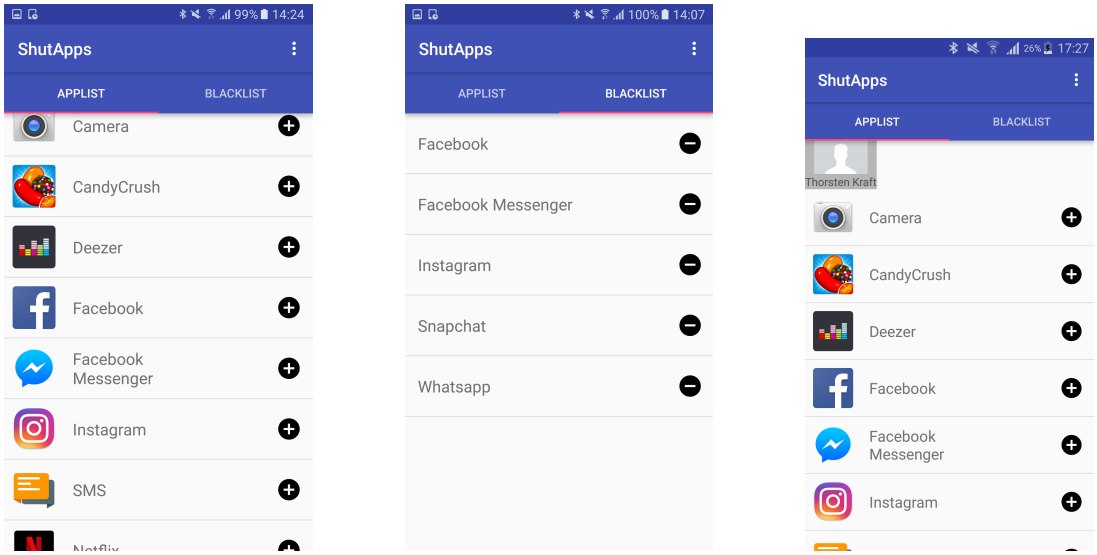
(b) aufgestellte Regeln

(c) Verstoße gegen aufgestellte Regeln

Abbildung 3.2: Screenshots von der Anwendung AppDetox

3.1.3 ShutApps - eine mobile Anwendung zur Einschränkung von Phubbing

ShutApps ist eine mobile Anwendung in Android entwickelt im Zuge des Praxisprojektes mit einem Kommilitone. Das Projekt sollte die Realisierbarkeit einer mobilen Anwendung zur Einschränkung von Phubbing zeigen. Dies ist mit der Entwicklung von ShutApps erfolgreich gezeigt worden. Die Funktionsweise dieser Anwendung wird im Folgenden beschrieben:
Als Erstes ist es wichtig, sich mit einem Facebook-Account in der Anwendung anzumelden, denn Facebook bildet die Grundlage zum Erkennen von Freunden. Mittels der Freundesliste in Facebook können andere Nutzer als Freunde definiert werden. Eine weitere Voraussetzung für die Anwendung ist die Nutzung von Bluetooth. Dadurch kann sie andere Nutzer in der Umgebung finden. Sollte es sich um Freunde handeln, können beide Smartphone miteinander kommunizieren. Außerdem hat der Nutzer die Möglichkeit, in einer vordefinierten Anwendungsliste, bestimmte Anwendungen auszusuchen und diese bei seinen Freunden zu blockieren. Sie können die blockierten Anwendung anschließend nicht mehr nutzen, solange sie sich in der Nähe von diesem Nutzer befinden. Dadurch will ShutApps erreichen, dass Freunde, die sich im Moment treffen, ihre Smartphone gegenseitig einschränken können und sich so mehr mit den Freunden beschäftigen als mit dem Smartphone.
Eine große Stärke dieser mobilen Anwendung ist die Minderung der Smartphonenutzung in Gegenwart von Freunden, so dass in gesellschaftlichen Treffen die Smartphonenutzung gemeinsam reguliert werden kann. Die zu entwickelnde Anwendung verfolgt das gleiche Ziel mit der Einschränkung der Smartphonenutzung bei Gruppenaktivitäten. In dieser Anwendung besteht wie bei den beiden anderen untersuchten Apps das Problem, dass aufgrund der technischen Möglichkeiten, die zum jetzigen Zeitpunkt nur Android zur Verfügung stellt, eine Realisierung von ShutApps für andere Betriebssysteme noch nicht möglich ist (Le u. Dinh (2017)). In der Abbildung 3.3 werden die Kern-Activities der Anwendung ShutApps abgebildet.



(a) Liste der vordefinierter Anwendungen

(b) Liste zu blockierender Anwendungen

(c) In der Nähe eines Freundes

Abbildung 3.3: Screenshots von der Anwendung ShutApps

3.1.4 Unicef Tap Project - Smartphone-Abstinenz für gute Zwecke

Unicef Tap Project ist eine mobile und web-basierte Anwendung von Unicef mit dem Ziel durch Nichtnutzen des Smartphones, Wasser an Bedürftige zu spenden. Dabei werden für zehn Minuten Smartphone-Abstinenz eine Tagesration Wasser bereitgestellt. Dafür muss der Nutzer entweder die mobile Anwendung installieren oder mit dem Smartphone die URL www.uniceftapproject.org im Browser abrufen. Da die Kampagne jedoch zum Zeitpunkt dieser Arbeit vorbei ist, lässt sich die Anwendung nicht mehr öffnen.
Zum Beginn der Anwendung startet der Nutzer einen Timer, der die Zeit ohne Nutzung des Smartphones aufzeichnen soll. Auf technischer Ebene funktioniert die Anwendung so, dass sie durch die Sensoren der Smartphones erkennen kann, ob das Smartphone bewegt wurde. Dadurch kann sie bestimmen, ob sich der Nutzer aktiv mit dem Smartphone während der Nutzung der Anwendung beschäftigt hat. Tritt dieser Fall ein, wird die Zeit, die das Smartphone bis dahin gemessen hat, sofort gestoppt.
Einer der großen Stärken dieser Anwendung im Vergleich zu den anderen ist, dass sie auf allen mobilen Endgeräten funktioniert, da sie unter anderem als Webanwendung entwickelt wurde. Die Realisierung einer solchen Webanwendung zeigt die Machbarkeit einer Anwendung, die über das Web ermöglicht, Menschen zu einer Smartphone-Abstinenz zu motivieren und zu unterstützen. Auch werden Erkenntnisse darüber gewonnen, welche Möglichkeiten eine Webanwendung hat, um auf Systemkomponenten eines Smartphones zuzugreifen. In der Abbildung 3.4 werden die wichtigsten Seiten der Webanwendung von Unicef Tap Project gezeigt.

(a) Startseite der Anwendung

(b) laufender Timer

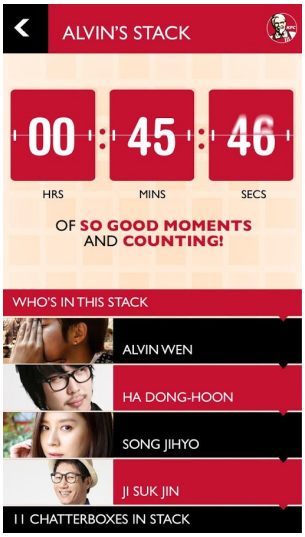
(c) Ergebnis der Zeit ohne Smartphone

Abbildung 3.4: Screenshots von der Webanwendung Unicef Tap Project (Quelle: Kat (2014))

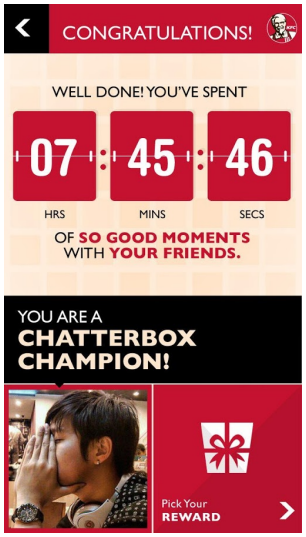
3.1.5 KFC Phonestack - gemeinsames Essen ohne Smartphone

KFC Phone Stack ist eine mobile Anwendung entwickelt von Mustard Worldwide für die Fast-Food-Kette KFC. Das Ziel dieser Anwendung ist es, in den Filialen dafür zu sorgen, dass die Kunden ihre Zeit dort mehr mit sozialen Interaktionen verbringen als sich mit dem Smartphone zu beschäftigen. Dies versucht die Anwendung dadurch zu erreichen, dass sie den Kunden nach einer gewissen Zeit an Smartphone-Abstinenz, Gutscheine anbietet, die sie in der Filiale für Essen einlösen können. Mit dieser Strategie versucht die Anwendung die Nutzer dazu zu motivieren, ihre Smartphonenutzung beim Essen in der Filiale einzuschränken.
Die Anwendung funktioniert wie folgt: Als Erstes erstellt ein Nutzer einen virtuellen Raum, in die er mittels eines QR-Codes Freunde einladen kann. Diese müssen den QR-Code mit der Anwendung abscannen und anschließend die Smartphones zu einem Turm stapeln. Danach kann der Ersteller des Raumes einen Timer starten, der aufzeichnet, wie lange sich die Smartphones schon in einem Stapel befinden. Sollte jemand ein Smartphone aus diesem Stapel entfernen, wird der Timer sofort gestoppt. Nach einer gewissen Zeit an Smartphone-Abstinenz erhalten die Nutzer Gutscheine, die sie in der Filiale einlösen können.
Einer der Stärken dieser Anwendung ist die Idee durch einen Anreiz in diesem Fall mit Gutscheinen die Nutzer dazu zu motivieren, gemeinsam die Smartphonenutzung einzuschränken. Jedoch ist die Umsetzung der App bisher nicht ausreichend erfüllt, so dass sie noch nicht ohne Probleme genutzt werden kann. Zum Zeitpunkt der Arbeit war diese mobile Anwendung nicht lauffähig. Dadurch lassen sich die Vorteile dieser Anwendung im Alltag noch nicht zeigen. Außerdem unterstützt diese Anwendung ebenfalls nur das Betriebssystem Android. In der Abbildung 3.5 werden die wichtigsten Screenshots der Anwendung dargestellt.

(a) Startseite der Anwendung



(b) Laufender Timer mit Teilnehmern



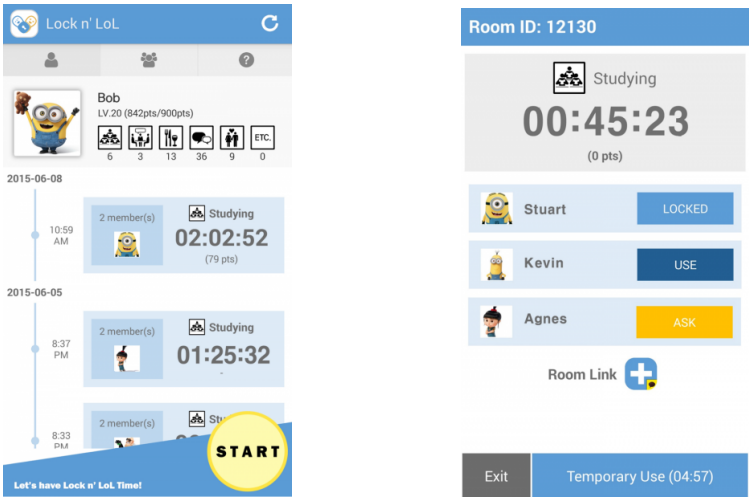
(c) Ergebnis der Zeit ohne Smartphone

Abbildung 3.5: Screenshots von der Android-Anwendung KFC Phonestack (Quelle: Google (2014))

3.1.6 Lock n' LoL - Smartphone-Abstinenz bei Gruppenaktivitäten

Lock n' LoL ist eine mobile Anwendung entwickelt von den Forschern Minsam Ko, Seungwoo Choi und Uichin Lee der KAIST Universität in Südkorea und in Zusammenarbeit mit dem Forscher Koji Yatani von der Universität in Tokyo. Sie beschreiben das Ziel ihrer Anwendung folgendermaßen: „This is an application designed to help users focus on their group activities by allowing group members to limit their smartphone usage together“ ((Ko u. a., 2016, S.1)). Zum Zeitpunkt dieser Arbeit ist die Anwendung bereits vom Markt entfernt.
Um die App nutzen zu können, muss der Nutzer als Erstes ein Profil anlegen, das ihn in der Anwendung personalisiert. Will er nun zusammen mit seinen Freunden bzw. Kollegen die Smartphonenutzung einschränken, muss er einen Raum erstellen, in die er per Room ID oder Room Link Personen einladen kann. Auf den Smartphones, die sich in einem Raum befinden, werden zudem alle Benachrichtigungen für die Dauer des Aufenthalts unterdrückt. Lediglich ankommende Anrufe können abgenommen werden. Eine Besonderheit, die Lock n' LoL besitzt, ist, dass der Nutzer die Möglichkeit hat, in der Sperrzeit für insgesamt fünf Minuten das Smartphone in dringenden Fällen zu benutzen, nachdem er im Raum eine Anfrage alle anderen Teilnehmer gesendet hat und diese akzeptiert wurde.

Die Möglichkeit eigene Räume zu erstellen, Freunde einzuladen und mit ihnen gemeinsam die Smartphonenutzung einzuschränken ist eines der Stärken von Lock n' LoL. Des Weiteren kann sie den Zugriff auf das Smartphone für die Dauer der Einschränkung komplett blockieren. Hier ist die Anwendung ebenfalls nur auf Android aufgrund der technischen Möglichkeiten im Vergleich zu anderen Betriebssysteme wie iOS oder Windows Phone beschränkt. In der Abbildung 3.6 werden zum einen die Profilseite des Nutzers und zum anderen die Activity eines Raum dargestellt.



(a) Zeitachse des Nutzers

(b) Raum zum Blockieren der Smartphonenutzung

Abbildung 3.6: Screenshots von der Android-Anwendung Lock n' LoL (Quelle: Ko u. a. (2016))

3.2 Anforderungsermittlung

Aus der Untersuchung der verwandten Arbeiten und Produkte können sich Anforderungen für das zu entwickelnde System ableiten lassen. Ein Aspekt, der bei der Analyse der verwandten Anwendungen aufgefallen ist, ist, dass der Großteil der Apps nur in Android zur Verfügung steht. Das lässt sich an den technischen Möglichkeiten, die Android im Vergleich zu anderen Betriebssystemen bietet, erklären. Aufgrund dieser Tatsache ist die Kompatibilität solcher Anwendungen zu den verschiedenen mobilen Betriebssystemen beschränkt (Le u. Dinh, 2017, S.9). Die zu entwickelnde Anwendung soll gezielt für alle Smartphones und ihre unterschiedlichen Betriebssysteme entwickelt werden, weshalb eine Realisierung einer mobilen web-basierten Anwendung vorgesehen ist. Weitere Anforderungen entstehen aufgrund dieser geplanten Eigenschaft der Anwendung. Des Weiteren entwickelten sich im Laufe der Arbeit neue geplante Features, aus denen sich weitere Anforderungen ableiten ließen.

Alle ermittelten Anforderungen sollen im Folgenden in funktionale und nicht-funktionale Anforderungen eingeteilt werden. Die Formulierung dieser Anforderungen orientiert sich an der Anforderungsschablone in der Literatur von Rupp u. Joppich (2014). Sie werden mit den Verben „muss, soll und kann“ nach ihrer Wichtigkeit für die zu entwickelnde Anwendung priorisiert.

3.2.1 funktionale Anforderungen

Funktionale Anforderungen beschreiben, was ein System leisten muss bzw. imstande ist, etwas zu tun. Hier sollen die funktionalen Anforderungen der zu entwickelnden Anwendung aufgelistet werden:

- F100: Die Anwendung muss fähig sein, virtuelle Räume zu erzeugen und Funktionen für diese bereitzustellen.
  - F110: Die Anwendung muss fähig sein, Nutzer in virtuelle Räume zuzuteilen.
  - F120: Die Anwendung muss dem Nutzer die Möglichkeit bieten, andere Nutzer in einen virtuellen Raum einzuladen.
    - F121: Die Anwendung soll dem Nutzer die Möglichkeit bieten, mit einem QR-Code andere Nutzer in einen virtuellen Raum einzuladen.
  - F130: Die Anwendung muss in der Lage sein, den Nutzer anzuzeigen, welche Nutzer sich in dem virtuellen Raum befinden.
  - F140: Die Anwendung muss in der Lage sein, das Verlassen eines Nutzers aus einem virtuellen Raum zu erkennen.
    - F141: Die Anwendung soll fähig sein, andere Nutzer im selben virtuellen Raum über das Verlassen eines Nutzers zu benachrichtigen.
- F200: Die Anwendung muss fähig sein, einen Timer für die in dem virtuellen Raum verbrachte Zeit zu erstellen.
  - F210: Die Anwendung soll dem Nutzer die Möglichkeit geben, den Timer für den virtuelle Raum zu starten.
  - F220: Die Anwendung soll dem Nutzer die Möglichkeit bieten, den Timer manuell stoppen zu können.
  - F230: Die Anwendung soll in der Lage sein, die verbrachte Zeit eines Nutzer mit anderen Nutzern im virtuellen Raum zu speichern.
  - F240: Die Anwendung kann dem Nutzer die Möglichkeit bieten, die im virtuellen Raum verbrachte Zeit zu bestimmten Gruppenaktivitäten zuzuordnen.
  - F250: Die Anwendung soll dem Nutzer die Möglichkeit bieten, auf sozialen Netzwerken die in dem virtuellen Raum verbrachte Zeit zu teilen.
- F300: Die Anwendung muss dem Nutzer die Möglichkeit bieten, einen Account in der Anwendung anzulegen.
  - F310: Die Anwendung muss fähig sein, die Account-Daten der Nutzer zu speichern.

  - F320: Die Anwendung muss in der Lage sein, den Nutzer zu authentifizieren.
  - F330: Die Anwendung soll dem Nutzer die Möglichkeit bieten, sich über soziale Netzwerke in der Anwendung anzumelden.
  - F340: Die Anwendung kann dem Nutzer die Möglichkeit bieten, sich per E-Mail in der Anwendung anzumelden.
- F400: Die Anwendung muss in der Lage sein zu erkennen, ob der Nutzer den mobilen Browser während der Nutzung der Anwendung verlassen hat.
  - F410: Die Anwendung muss fähig sein zu erkennen, ob die Anwendung den Hauptfokus im mobilen Browser besitzt.
- F500: Die Anwendung muss fähig sein, das automatische Ausschalten des Bildschirms am mobilen Endgerät während der Laufzeit des Timers zu verhindern.

3.2.2 nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben die Qualitätseigenschaften des Systems und wie gut das System eine Leistung erbringen muss (Böhm (2002)). Im Folgenden werden die nicht-funktionalen Anforderungen der Anwendung genannt:

- Die Anwendung muss auf verschiedenen mobilen Browser laufen können.
- Die Anwendung muss auf verschiedenen Smartphones funktionieren.
- Die Anwendung muss dafür sorgen, dass der Bildschirm des Smartphones während der Nutzung der Anwendung eingeschaltet bleibt.
- Der Dienstnutzer der Anwendung soll mit den Technologien HTML, CSS und JavaScript entwickelt werden.
- Die Anwendung muss orts- und zeitunabhängig genutzt werden können.
- Die Anwendung soll eine verlustfreie Datenübertragung zwischen Systemkomponenten gewährleisten.
- Das Smartphone, auf dem die Anwendung läuft, muss mit dem Internet verbunden sein.

3.3 Systemarchitektur der Anwendung

Für die Implementierung einer Anwendung ist es wichtig, im Vorfeld eine Systemarchitektur für die geplante Anwendung zu entwerfen, die die einzelnen Komponenten des Systems und deren Zusammenspiel visualisiert. Basierend auf den aufgestellten Anforderungen wird im folgenden Entwurf die konzipierte Architektur dargestellt und anschließend erläutert.

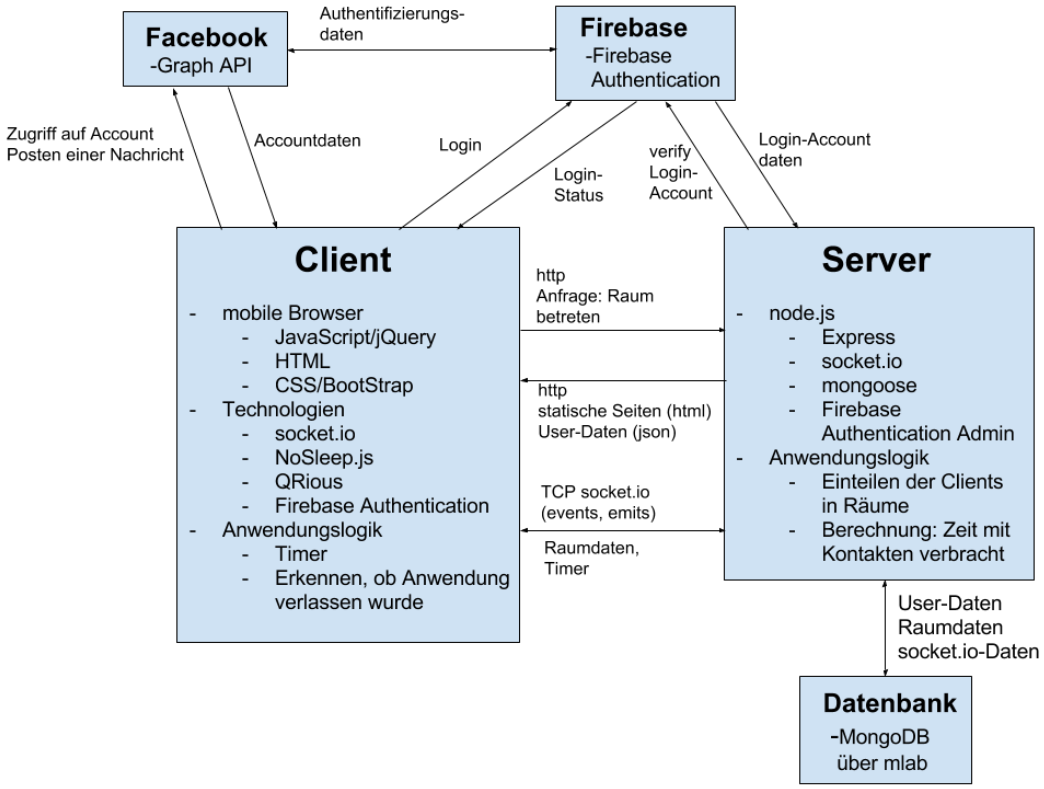


Abbildung 3.7: Systemarchitektur

In der Abbildung 3.7 ist die geplante Systemarchitektur der zu entwickelnden Anwendung abgebildet. Es handelt sich prinzipiell um eine verteilte Architektur in Form einer Client-Server-Architektur, denn die Webanwendung soll es ermöglichen, mehrere Clients in einem virtuellen Raum zuzuteilen. Da viele Clients mit der Anwendung kommunizieren müssen, ist es sinnvoll, eine Client-Server-Architektur zu nutzen, damit beispielsweise der Server in diesem Falle die Aufgabe der Zuteilung von Clients in virtuelle Räume übernehmen kann. Hier bietet es sich an, die Webanwendung als eine REST-konforme Architektur zu realisieren, da bereits Client und Server strikt voneinander getrennt sind und der Datentausch zwischen ihnen unter anderem mit dem Transportprotokoll HTTP realisiert werden soll. Generell ist zu sagen, dass Client und Server sich über zwei Kommunikationskanäle miteinander Nachrichten senden können. Einmal wie bereits beschrieben über HTTP mithilfe der HTTP-Methoden und mittels dem Framework socket.io über das auf TCP basierende WebSocket-Protokoll. Die einzelnen Komponenten und deren Zusammenspiel werden in den folgenden Abschnitten detaillierter erläutert.

3.3.1 Mobiler Browser als Client

Das zu entwickelnde System ist als eine mobile web-basierte Anwendung geplant. Daher wird der Client dieses Systems für mobile Endgeräte insbesondere mobile Browser spezialisiert. Hier sollen Technologien wie HTML, CSS mittels des Frameworks BootStrap und JavaScript bzw. jQuery zum Einsatz kommen. Sie bilden die Grundlage zur Entwicklung des Clients. Des Weiteren sollen wie in Abbildung 3.7 weitere Technologien genutzt werden, um den Client zu realisieren. NoSleep.js soll eingesetzt werden, damit nach längerer Smartphone-Abstinenz der Bildschirm des mobilen Endgeräts sich nicht automatisch abschaltet, solange die Webanwendung im Einsatz ist. QRious nutzt der Client, um die ID eines virtuellen Raumes in Form eines QR-Codes verschlüsseln zu können. Über Firebase Authentication kann der Client mit Firebase kommunizieren und Login-Daten senden bzw. den Login-Status eines Nutzers abfragen. Firebase Authentication soll die Aufgabe der Authentifizierung in der Anwendung vereinfachen. Wie bereits beschrieben, kommuniziert der Client mit dem Server über zwei Kanäle. Damit er über das auf TCP basierende WebSocket-Protokoll kommunizieren kann, muss die clientseitige socket.io-Library integriert werden. Dadurch wird es dem Client ermöglicht, socket.io Ereignisse zu empfangen und mittels socket.io-emit Ereignisse auszulösen und Nachrichten zu senden. Außerdem ist für den Client eine Anbindung an das soziale Netzwerk *Facebook* vorgesehen, damit er nach Abbruch des Timers die Zeit ohne Smartphone nach vorheriger Zustimmung auf Facebook veröffentlichen kann. Dafür soll der Client über die Graph API von Facebook auf die Account-Daten des über Facebook eingeloggten Nutzers zugreifen und anschließend über sein Profil Nachrichten veröffentlichen können.

3.3.2 Webserver in node.js

Für die Umsetzung des Servers ist die JavaScript-Laufzeitumgebung node.js vorgesehen. Mittels node.js ist es möglich, einen REST-Server für eine Webanwendung zu realisieren. Dieser Server kommuniziert über zwei Kommunikationskanäle mit dem Client. Mithilfe der Express-Module kann node.js eine REST-Schnittstelle zur Verfügung stellen und über Routen und mit HTTP-Methoden auf Anfragen des Clients reagieren. Im Folgenden wird eine REST-Spezifikation in der Tabelle 3.1 dargestellt, die die genutzten Ressourcen bezüglich ihrer HTTP-Methoden, ihrer Semantik und ihrem Content-Type beim Request und beim Response beschreibt.

Tabelle 3.1: REST-Spezifikation der verwendeten Ressourcen

Ressource	**Methode**	**Semantik**	**Content-Type (Request)**	**Content-Type (Response)**
/login	GET	Anzeige der statischen HTML-Seite für das Login	-	text/html
/home	GET	Anzeige der statischen HTML-Seite für die Startseite	-	text/html
/home/:roomid	GET	Anfrage zur Zuteilung des Client in den virtuellen Raum mit der ID roomid	-	plain
/contacts	GET	Anzeige der statischen HTML-Seite für die verbrachten Zeit mit Kontakten	-	text/html
/users	GET	Anzeige aller gespeicherten User	-	application/json
/users/:uid	GET	Anzeige eines gespeicherten Users mit der User-ID uid	-	application/json
/users/:uid	PUT	Aktualisierung eines gespeicherten Users mit der User-ID uid	application/json	plain
/users/:uid	DELETE	Löschen eines gespeicherten Users mit der User-ID uid	-	plain

Des Weiteren ist es nötig, auf dem Server das Framework socket.io zu integrieren, damit die Kommunikation zum Client mit socket.io über WebSockets ermöglicht wird. Hier soll die serverseitige Library von socket.io eingesetzt werden.
Für den Server sind folgende Aufgaben vorgesehen: Er soll die Clients nach Anfrage in bestimmte Räume einteilen und die verbrachte Zeit eines Nutzer mit anderen Nutzern in einem Raum berechnen. Außerdem wird die persistente Datenhaltung über den Server geregelt, der durch eine Anbindung an MongoDB mit der Datenbank kommunizieren kann. Durch das node-module mongoose wird dies ermöglicht.
Mit der Firebase Authentication Admin API soll auf dem Server anstatt über Firebase möglich sein, die in der Anwendung erstellten Accounts zu authentifizieren. Dadurch soll gewährleistet werden, dass Anfragen, in denen eine Authentifizierung erforderlich ist, nur bearbeitet werden, wenn diese von authentifizierten Clients gesendet wurden.

3.3.3 Echtzeit-Kommunikation zwischen Client und Server mit socket.io

Die Echtzeit-Kommunikation zwischen Client und Server soll in der Anwendung über eine WebSocket-Verbindung mittels dem Framework socket.io erfolgen. Der Grund, warum eine Echtzeit-Kommunikation mithilfe von socket.io in der zu entwickelnden Anwendung eingesetzt werden soll, ist die Nutzung der virtuellen Räume und des gemeinsamen Timers. Über die WebSocket-Verbindung soll der Server die Clients beispielsweise über das Verlassen oder Beitreten eines Clients informieren und die Zeit des Timers mitzählen und zwischenspeichern. Für das Framework socket.io sind aufgrund der bereitgestellten Möglichkeiten weitere Aufgaben vorgesehen.

3.3.4 Datenspeicherung mit der NoSQL-Datenbank MongoDB

Die persistente Datenhaltung erfolgt über die dokumentenorientierte NoSQL-Datenbank MongoDB, die beim Server mittels der node.js-Module mongoose Anbindung findet. Aufgrund der Tatsache, dass alle genutzten Daten im JSON-Format repräsentiert werden, ist die dokumentenorientierte Eigenschaft von MongoDB von Vorteil, da alle Daten in BSON (Binary JSON) gespeichert werden und dies nur eine erweiterte Form von JSON ist. Dadurch lassen sich die Daten in MongoDB unkompliziert speichern. Generell werden User-Daten, Raumdaten und socket.io-Daten auf der Datenbank persistent abgelagert. Da MongoDB nur lokal auf einem Rechner laufen würde und eine verteilte Webanwendung dadurch keinen Zugriff auf die Datenbank hätte, soll der Database-as-a-Service mLab eingesetzt werden, um MongoDB im Web nutzen zu können. Im weiteren Verlauf des Textes soll die Struktur der gespeicherten Daten dargestellt und anschließend erläutert werden.

User-Daten

Die gespeicherten User-Daten spielen in dieser Webanwendung eine große Rolle und sind Voraussetzung für einige Algorithmen, die in einem späteren Kapitel erläutert werden. Das Listing 3.1 zeigt die Datenstruktur der User-Daten als MongoDB-Schema in der Anwendung.

```
1  //Datenstruktur nach dem MongoDB-Schema fuer User-Daten
2  {
3      uid: String,
4      name: String,
5      connection: String,
6      toJoinedRoom: String,
7      contacts: [{
8          _id: false,
9          uid: String,
10         timeSpent: String
11     }]
12 }
```

Listing 3.1: Datenstruktur der User-Daten

Unter dem Key-Value uid wird die User-ID, die in Firebase Authentication generiert wurde, gespeichert. Dadurch lassen sich alle angemeldeten Nutzer eindeutig unterscheiden. Der Key-Value name beinhaltet entweder die durch die Anmeldung bereitgestellte E-Mail-Adresse oder den Namen des angemeldeten Facebook-Accounts. Unter connection soll temporär die erzeugte Socket-ID des angemeldeten Nutzers gespeichert werden, mit der bestimmte Algorithmen auf dem Server arbeiten werden. Sollte eine neue Verbindung mit einem neuen Socket erfolgen, wird die alte Socket-ID durch die neue ID ersetzt. Dadurch weißt der Server, welcher Nutzer welchen Socket in socket.io im Moment benutzt. Der Wert hinter dem Key-Value toJoinedRoom soll angeben, welchen Raum der Nutzer beitreten soll. Am Anfang ist dieser Wert leer, bis der Client eine Anfrage zum Beitreten eines Raumes an den Server sendet. Die zu beitretende Raum-ID wird anschließend unter toJoinedRoom gespeichert. Das Array contacts mit den Key-Values uid und timeSpent soll die Kontakte speichern, mit denen der Client schon mindestens ein Mal zusammen in einem virtuellen Raum war. Hier stellt die uid die Nutzer-ID eines Kontaktes dar während unter timeSpent die verbrachte Zeit miteinander in Sekunden notiert wird.

Sockets

Die Anwendung muss wissen, welche Sockets im Moment aktiv sind. Deshalb soll der Server auf die IDs der aktiven Sockets zugreifen und diese auf der Datenbank zwischenspeichern. Sollte eine Verbindung zu einem aktiven Socket abgebrochen werden, wird der entsprechende Eintrag in der Datenbank gelöscht. Die Struktur dieser Daten wird im folgenden Listing 3.2 dargestellt.

```
1 //Datenstruktur nach dem MongoDB-Schema fuer Sockets
2 {
3     _id: false,
4     id: String
5 }
```

Listing 3.2: Datenstruktur der Sockets

Die MongoDB-eigene ID wird in Zeile 3 des Listings 3.2 abgeschaltet, da sie für die Anwendung keine Bedeutung hat. Die Socket-IDs der aktiven Sockets werden unter dem Key-Value id eingetragen.

3.3.5 Authentifizierung mittels Firebase Authentication

Die Authentifizierung in dieser Anwendung soll dazu dienen, alle vorhandenen Nutzer zu personalisieren, denn der Server muss alle Clients voneinander unterscheiden können. In einer REST-konformen Architektur müssen Client und Server zustandslos miteinander kommunizieren, was bedeutet, dass keine der beiden Systemkomponenten Zustandsinformationen zwischen Nachrichten speichern soll. Jede neue Anfrage muss alle notwendigen Informationen beinhalten, so dass beide Systemkomponenten sie verstehen können. Mithilfe der Firebase Authentication soll die Authentifizierung in der Webanwendung erfolgen. Hier liegt der Fokus auf die Anmeldung durch das soziale Netzwerk Facebook. Der Client soll mit Firebase kommunizieren, um die Login-Daten der Nutzer dort zu speichern und den Login-Status der Nutzer erfragen zu können. Bei einer Anfrage vom Client an den Server, bei der eine Authentifizierung verlangt wird, soll der Client auf Firebase zugreifen, um einen personalisierten ID-Token zu bekommen. Dieser wird mit jeder Anfrage, die eine Authentifizierung benötigt, an den Server gesendet. Diese Anfrage an den Server wird nach erfolgreicher Authentifizierung bearbeitet. Dadurch sollen Client und Server bei Anfragen, bei denen eine Authentifizierung erforderlich ist, zustandslos miteinander kommunizieren können.

4 Implementierung

In diesem Kapitel wird die Implementation der Webanwendung erläutert, die auf Grundlage des entwickelten Konzepts realisiert wurde. Hier werden die elementaren Algorithmen und Anwendungslogiken der App anhand von Code-Beispielen dargestellt.

4.1 Virtueller Raum zum gemeinsamen Starten eines Timers

Für das Erstellen und Nutzen eines virtuellen Raumes ist das Framework socket.io vorgesehen. Die Funktionsweise von socket.io erlaubt pro Verbindung bzw. pro Client das Erzeugen eines Sockets auf dem Server. Jedoch muss vom Client die Anfrage gestellt werden, dass er eine neue WebSocket-Verbindung aufbauen will. Dies wird dadurch erreicht, indem der Client ein neues Socket-Objekt mit var socket = io() erstellt oder mit socket.io.connect() manuell den Befehl zum Verbindungsaufbau gibt, nachdem das Objekt bereits erzeugt wurde. Daraufhin wird auf dem Server ein neues Socket erzeugt, was im Listing 4.1 zu sehen ist. Über die WebSocket-Verbindung können Client und Server miteinander kommunizieren und Daten austauschen.

```
1 io.on('connection', function(socket) {
2   console.log(socket + "ist das pro Client erzeugte Socket");
3 });
```

Listing 4.1: Erzeugen eines Sockets auf dem Server

Zum Senden von Nachrichten und Daten muss der Socket auf selbstdefinierte Ereignisse horchen und beim Auslösen darauf reagieren. Das Listing 4.2 zeigt, wie beim Socket Ereignisse registriert werden können und wie über den Socket ein Ereignis auslöst werden kann. Die Vorgehensweise ist auf dem Client sowie auf dem Server identisch.

```
1 //Registrieren eines Ereignisses
2 socket.on('roomId', function(data) {
3   console.log("Die RoomID lautet " + data);
4 });
5
6 //Ausloesen eines Ereignisses
7 var roomID = 1234;
8 socket.emit('roomId', roomID);
```

Listing 4.2: Auslösen und Registrieren eines Ereignisses

In Zeile 3 im Listing 4.2 wird auf dem Socket das Ereignis roomId registriert und so kann der Socket die Daten, die mit dem Auslösen eines Ereignisses gesendet wurden, empfangen und ausgeben. Der Code in den Zeilen 7-8 löst ein Ereignis mit dem Namen roomId aus und sendet dabei die Raum-ID mit der Nummer 1234. In diesem Fall empfängt der Socket, der eben das roomId-Ereignis registriert hat, die Raum-ID unter der Variable data und kann diese weiterverarbeiten.
Des Weiteren bietet socket.io die Möglichkeit, die verbundenen Sockets in sogenannte rooms zuzuordnen. Normalerweise wird das Auslösen eines Ereignisses an alle verbundenen Sockets mitgeteilt. Socket.io kann jedoch auch gezielt Nachrichten an bestimmte rooms senden, so dass nur die Sockets, die sich in diesem Raum befinden, die gesendete Nachricht erhalten. Das folgende Listing 4.3 stellt das Betreten eines Sockets in einem Raum dar.

```
1 var roomName = "exampleRoom";
2 socket.join(roomName);
```

Listing 4.3: Erstellen und Betreten eines Raumes in socket.io

Das erlaubt der Anwendung, mehrere Clients in einen virtuellen Raum einzuteilen und speziell an diese Clients Nachrichten zu senden und von ihnen Nachrichten zu empfangen. Dieses Feature von socket.io ermöglicht der Anwendung das Erstellen eines virtuellen Raumes zum gemeinsamen Starten eines Timers. Im Folgenden soll die Implementierung dieser Funktion detailliert erläutert werden.
Das Listing 4.4 stellt das Ereignis create in gekürzter Form dar, das ausgelöst wird, wenn ein Client eine Anfrage zum Erstellen eines neuen Raumes stellt. Der komplette Code zu diesem Ereignis lässt sich in der mitgesendeten CD in dem Code des Servers finden.

```
1 socket.on('create', function(client) {
2   var roomId = 'room' + client;
3   //...
4   socket.join(roomId);
5   //...
6   socket.emit('roomId', roomId);
7   //...
8 });
```

Listing 4.4: Ereignis create in gekürzter Form auf dem Server der entwickelten Anwendung

Die roomId in dieser Anwendung in Zeile 2 wird nach folgendem Schema konstruiert: Am Anfang der Raum-ID ist stets der String room zu finden und anschließend die Socket-ID des Clients. Mithilfe einer solchen Raum-ID soll die Anwendung unterscheiden können, welcher Client welchen Raum erstellt hat. Anschließend wird der Socket des Clients dem erstellten Raum zugewiesen und dem Client die Raum-ID mitgeteilt. Dadurch kann er mit dieser Raum-ID, die auf dem Client in einer URL als QR-Code generiert wird, andere Nutzer in seinen Raum einladen. Das folgende Listing 4.5 zeigt die Route, über die ein Client eine Anfrage senden kann, um einen bestimmten Raum beizutreten.

```
app.get('/home/:roomid', function(req, res) {
    if(req.params.roomid !== undefined) {
        var toCheckedId = req.params.roomid.replace("room", "");
        Socket.findOne( {id: toCheckedId}, function(err, socket) {
            if(socket) {
                console.log('Socket vorhanden');
                User.findOneAndUpdate( {uid: req.session.uid}, {$set: {
                    toJoinedRoom: req.params.roomid}}, function(err,
                    user) {
                    if(err) {
                        console.log(err);
                    } else {
                        console.log(user);
                    }
                });
            } else {
                console.log('Socket nicht vorhanden');
            }
        });
    }
    res.redirect('/home');
});
```

Listing 4.5: Route /home/:roomid auf dem Server zum Beitreten eines Raumes

Wird der erstellte QR-Code von einem anderen Nutzer abgescannt, wird das Smartphone über den mobilen Browser auf den Pfad /home/:roomid weitergeleitet. Die in dem QR-Code verschlüsselte Raum-ID ersetzt die roomid in dem Pfad und sendet so eine Anfrage an den Server zum Beitreten des virtuelles Raumes mit dieser Raum-ID. Der Code in der Route im Listing 4.5 wird daraufhin ausgeführt. Wichtig für das Ausführen dieser Route ist es, dass in der Anfrage des Clients die Raum-ID mitgesendet wird. Hier kommt nun das zuvor beschriebene Schema der Raum-ID zum Einsatz. Der Server kann die Raum-ID so extrahieren, dass er die Socket-ID vom Raumersteller auslesen und die Datenbank abfragen kann, ob diese Socket-ID existiert bzw. im Moment aktiv ist. Alle IDs der aktiven Sockets werden in der Datenbank zwischengespeichert, damit der Server stets weiß, welche Verbindungen im Moment geöffnet sind. Wird die ID des Sockets in der Datenbank gefunden, greift der Server auf die User-Daten zu und sucht nach dem Nutzer, der die Anfrage gesendet hat. Beim Erfolgsfall trägt der Server die zu beitretende Raum-ID unter dem Key-Value toJoinedRoom des Nutzers ein, der die Anfrage zum Beitreten gestellt hat, und leitet auf die Ressource home zurück.
Damit der Server den anfragenden Nutzer in den virtuellen Raum zuweisen kann, wird auf eine Eigenschaft von socket.io zurückgegriffen. Wenn eine Seite, in der socket.io integriert ist, neu geladen wird, wird die aktuelle WebSocket-Verbindung getrennt und eine neue Verbindung zwischen Client und Server aufgebaut. Jedes Mal, wenn der Client die Ressource home anfragt, wird dem Client eine neue Seite präsentiert und diese neu geladen, so dass die alte Verbindung getrennt und eine neue WebSocket-Verbindung aufgebaut wird. Das folgende Listing 4.6 stellt einen Ausschnitt des serverseitigen auth-Ereignisses dar.

```
1  //...
2  if(user.toJoinedRoom !== "") {
3    socket.join(user.toJoinedRoom);
4    socket.joinedRoom = user.toJoinedRoom;
5    io.to(user.toJoinedRoom).emit('joinedRoom', user.toJoinedRoom);
6    var clients = io.sockets.adapter.rooms[user.toJoinedRoom].sockets;
7    User.findOne( {connection: socket.id}, function(err, userdata) {
8      if(err) {
9
10     } else {
11       User.findOne( {connection: user.toJoinedRoom.replace("room", "")},
             function(err, owner) {
12         socket.emit('clients', owner.name);
13         io.to(user.toJoinedRoom).emit('clients', userdata.name);
14       });
15     }
16   });
17   User.findOneAndUpdate( {id: decodedToken.uid}, {$set: {toJoinedRoom: "
       "}}, function(err, user) {
18     if(err) {
19       console.log(err);
20     } else {
21       console.log(user);
22     }
23 });
```

Listing 4.6: Ausschnitt aus dem serverseitigen auth-Ereignis

Das auth-Ereignis wird jedes Mal vom Client ausgelöst, wenn die Seite unter dem Pfad /home neu geladen wird und der Nutzer über Firebase Authentication in der Anwendung angemeldet ist. Dadurch, dass der Server den Wert für toJoinedRoom in den User-Daten des anfragenden Clients zuvor hinzugefügt hat, wird die Logik im Listing 4.6 ausgeführt. Normalerweise ist der Wert für toJoinedRoom leer, weshalb dieser Ausschnitt des auth-Ereignis nur nach einer Anfrage des Clients zum Beitreten eines Raumes durchgeführt wird. In diesem Fall weist der Server dem anfragenden Client den Raum mit der in dem Key-Value toJoinedRoom gespeicherten Raum-ID zu und benachrichtigt alle Clients in diesem Raum über den neuen Nutzer. In den Zeilen 17-22 wird der zuvor genutzte Wert von toJoinedRoom wieder auf einen leeren String gesetzt, damit dieser Client bei einer neuen Socket-Verbindung den selben Raum nicht das zweite Mal beitritt.

4.2 Fähigkeit der Vordergrunderkennung im mobilen Browser

Die Anforderungen F400 und F500 geben der Anwendung vor die Fähigkeit zu haben, das Schließen und Verlassen des mobilen Browser zu erkennen und das automatische Ausschalten des Bildschirms am Smartphone während der Laufzeit der Timers zu verhindern. Dadurch soll der Nutzer motiviert werden, in der Laufzeit des Timers die Anwendung nicht zu verlassen und so seine Smartphonenutzung einzuschränken, da die Anwendung stets im Vordergrund laufen würde.

Damit der Bildschirm des mobilen Endgerätes sich nicht automatisch ausschaltet, wird wie im Konzept beschrieben das Framework NoSleep.js eingesetzt. Dafür wird im Client mittels dem Framework ein neues NoSleep-Objekt erzeugt, mit denen sich zwei Methoden enable() und disable() nutzen lassen. Mit enable() wird ein Wakelock aktiviert, so dass sich der Bildschirm des mobilen Endgerätes nicht automatisch ausschaltet. Durch das Aufrufen der disable()-Methode wird der zuvor aktivierte Wakelock entfernt.
Um erkennen zu können, ob der mobile Browser auf dem mobilen Endgerät verlassen oder geschlossen wurde, greift die Anwendung auf eine Eigenschaft von socket.io zurück. Durch einen Heartbeat kontrolliert socket.io, ob die Verbindung beim Client oder beim Server noch zur Verfügung steht, damit das Framework erkennen kann, ob die Client-Server-Verbindung noch offen ist. Ein Heartbeat ist ein Mechanismus, in der zwei oder mehrere Systemkomponente sich gegenseitig darüber benachrichtigen, ob sie noch verfügbar sind, indem sie in kurzen Intervallen sich gegenseitig Nachrichten senden. Mit diesem Mechanismus kann die Anwendung kontrollieren, ob der mobile Browser sich noch im Vordergrund befindet. Sollte der Fall eintreten, dass der Nutzer den Browser geschlossen oder verlassen hat, wird die Verbindung beim Client getrennt und kann auf dem mobilen Browser keine Nachricht an den Server zurücksenden, so dass der Server davon ausgehen kann, dass der Browser nicht mehr aktiv im Vordergrund ist.
Jedoch wird durch den vorherigen geschilderten Mechanismus eine Situation nicht in Betracht gezogen und zwar dass der Nutzer die Möglichkeit durch die Tabs in den mobilen Browsern hat, weiterhin das Internet zu nutzen, ohne die Anwendung wirklich zu verlassen. Zum Lösen dieser Situation wurde die PageVisibility API (W3C (2017)) eingesetzt, die es dem Browser erlaubt zu bestimmen, ob die aktuelle Seite bzw. Tab im Moment im Hauptfokus des Browsers liegt. Dadurch kann die Anwendung erkennen, ob sie aktiv im Vordergrund des Browsers ist oder den Fokus durch das Öffnen eines neuen Tabs verloren hat.
Mit diesen Fähigkeiten der Anwendung kann der Nutzer die Anwendung nicht verlassen, ohne dass die Anwendung es erkennen kann. Diese Eigenschaft ist für das Erstellen des Timers und für das Zählen der verbrachten Zeit in einem virtuellen Raum essentiell.

4.3 Timer zum Zählen der verbrachten Zeit in einem virtuellen Raum

In den vorherigen zwei Abschnitten wurde erläutert, wie virtuelle Räume in der Anwendung erstellt, wie andere Nutzer diese Räume beitreten können und wie die Anwendung erkennt, ob der Nutzer die Anwendung auf seinem Smartphone noch aktiv offen hat. Die Räume dienen als Grundlage für das Erstellen eines gemeinsamen Timers, der die verbrachte Zeit in einem virtuellen Raum mitzählt. Nachdem ein Nutzer einen Raum erstellt und seine Freunde eingeladen hat, kann er den Timer des Raumes starten. Dadurch startet auf jedem Client im Raum ein Algorithmus, der einen Counter pro Sekunde um eins hoch zählt. Dieser Counter stellt die verbrachte Zeit im Raum in Sekunden dar. Damit der Server den Counter aller beteiligten Clients zwischenspeichern kann, wird nach jeder Inkrementierung der Wert des Counters über die WebSocket-Verbindung in Echtzeit an den Server übermittelt. Sollte der Timer durch Abbruch oder durch das Verlassen des Browsers gestoppt werden, bricht er das Zählen ab und der Server wird über den Abbruch benachrichtigt. Er wird ebenfalls über die verbleibenden Clients informiert. Mit dem aktuellen zwischengespeicherten Wert des Counters kann der Server weitere Berechnungen durchführen.

4.4 Berechnung und Anzeige der Zeit mit verbrachten Kontakten

Der Abbruch eines Timers kann in der Anwendung in zwei Situationen auftreten. Der Nutzer beendet den Timer durch das manuelle Betätigen eines Buttons oder der Nutzer verlässt während der Laufzeit des Timers die Anwendung oder den Browser. Auf der technischen Ebene der Anwendung betrachtet trennt sich durch das Beenden des Timers die aktuelle WebSocket-Verbindung zwischen Client und Server. Aufgrund dessen löst sich auf dem Server und auf dem Client das Ereignis disconnect aus, dass sich jedes Mal aktiviert, sobald eine WebSocket-Verbindung getrennt wird.
In diesem Ereignis berechnet die Anwendung die verbrachte Zeit eines Nutzers mit anderen Kontakten im selben Raum. Für die Berechnung müssen zwei Sichtweisen in Betracht gezogen werden. Zum Einen muss bei einem Nutzer A die verbrachte Zeit mit den anderen Nutzern im Raum gespeichert werden. Zum Anderen muss die verbrachte Zeit der anderen Nutzer mit Nutzer A notiert werden. Das folgende Listing 4.7 stellt den Algorithmus für den ersten geschilderten Fall dar. Es handelt sich um einen Ausschnitt des Ereignisses disconnect auf dem Server.

```
for(var key in clients) {
  tempArray.push(key);
  User.findOne( {uid: socket.handshake.session.uid}, function(err,
      contact) {
    if(err) {
      console.log(err);
    } else {
      User.findOne( {connection: tempArray[i++]}, function(err, userdata
          ) {
        if(userdata) {
          User.findOne( {uid: socket.handshake.session.uid, 'contacts.
              uid': userdata.uid}, function(err, contacts) {
            //wenn contacts mit einer bestimmten uid nicht vorhanden ist
            if(!contacts) {
              User.update( {uid: socket.handshake.session.uid}, {$push:
                  {contacts: {uid: userdata.uid, timeSpent: socket.
                  counter}}}, function(err, user) {
                if(err) {
                  console.log(err);
                } else {

                }
              });
            //wenn contacts mit bestimmter uid vorhanden ist
            } else {
              var tempTime = contacts.contacts.filter(function(item) {
                return item.uid === userdata.uid;
              });
              var newTimeSpentInt = parseInt(tempTime[0].timeSpent) +
                  parseInt(socket.counter);
              var newTimeSpent = newTimeSpentInt.toString();
              User.update( {uid: socket.handshake.session.uid, 'contacts
                  .uid': userdata.uid}, {$set: {'contacts.$.timeSpent':
                  newTimeSpent}}, function(err, user) {
                if(err) {
                  console.log(err);
                } else {

                }
              });
            }
          });
        }
      });
    }
  });
};
//...
```

Listing 4.7: Berechnung der verbrachten Zeit aus Sicht eines Nutzers (1:n)

Der Server muss ermitteln, wie viele Clients sich zum Zeitpunkt des Abbruchs eines Timers im Raum befinden. socket.io bietet in diesem Fall eine Funktion an, die nach Angabe einer Raum-ID die Anzahl und die Identität der in dem Raum befindenden Clients wiedergibt. Für den im Listing 4.7 dargestellten Algorithmus ist die Anzahl der noch verbleibenden Clients im Raum ausschlaggebend, denn sie bestimmt wie oft der Algorithmus durchgeführt werden muss, was in Zeile 1 in der for-Schleife angegeben wird. Dieser Algorithmus behandelt die Situation, in der der Server die verbrachte Zeit eines Nutzers mit anderen Nutzern speichern soll. Dies lässt sich mit der Kardinalität 1:n in der Datenbankmodellierung illustrieren. Dafür greift der Server auf die Datenbank zu und bearbeitet die User-Daten des Clients, der in der 1:n-Beziehung die Master-Rolle darstellt. Für eine bessere Veranschaulichung wird dieser Client im weiteren Verlauf Nutzer A genannt. Im Abschnitt 3.3.4 ist die Datenstruktur der User-Daten bereits beschrieben und in dem Array contacts werden Kontakte und die verbrachte Zeit mit ihnen gespeichert, mit denen man mindestens ein Mal in einem Raum war. Der Algorithmus betrachtet beim Berechnen der verbrachten Zeit für Nutzer A mit anderen Nutzern zwei verschiedene Fälle.
Der erste Fall, der im Listing 4.7 von Zeile 12-19 dargestellt ist, tritt ein, wenn der zu speichernde Nutzer noch nicht in den Kontakten von Nutzer A vorhanden ist. Dazu erstellt der Server über die Datenbank in dem Array contacts von Nutzer A einen neuen Eintrag mit der zu speichernden User-ID und der dazugehörigen verbrachten Zeit in Sekunden, die vorher auf dem Server zwischengespeichert ist.
Die Zeilen 22-33 zeigen den Algorithmus, der ausgeführt wird, wenn der zweite Fall eintritt, dass der zu speichernde Nutzer bereits einen Eintrag in contacts von Nutzer A hat. Hier greift der Server auf die bereits vorhandenen Daten des zu speichernden Nutzers in contacts von Nutzer A zu und berechnet und aktualisiert anschließend die neue verbrachte Zeit zwischen ihnen.

```
1
2  User.findOne( {connection: tempArray[j], 'contacts.uid': socket.
       handshake.session.uid}, function(err, userdata) {
3    if(!userdata) {
4      User.update( {connection: tempArray[j++]}, {$push: {contacts: {uid:
           socket.handshake.session.uid, timeSpent: socket.counter}}},
           function(err, user) {
5      });
6    } else {
7      User.findOne( {uid: socket.handshake.session.uid, 'contacts.uid':
           userdata.uid}, function(err, contacts) {
8        var tempItem = contacts.contacts.filter(function(item) {
9          return item.uid === userdata.uid;
10       });
11       User.update( {connection: tempArray[j++], 'contacts.uid': socket.
             handshake.session.uid}, {$set: {'contacts.$.timeSpent':
             tempItem[0].timeSpent}}, function(err, user) {
12       });
13     });
14   }
15 });
```

Listing 4.8: Berechnung der verbrachten Zeit aus Sicht der anderen Nutzer (n:1)

Das Listing 4.8 stellt einen weiteren Ausschnitt aus dem Ereignis disconnect auf dem Server dar. Hier ist der Algorithmus abgebildet, der die Berechnung der verbrachten Zeit aus der Sichtweise der im Raum verbleibenden Clients durchführt. Dies lässt sich an der Kardinalität n:1 beschreiben, wobei Nutzer A die Master-Rolle weiterhin übernimmt. In diesem Code-Abschnitt werden ebenfalls zur Berechnung der verbrachten Zeit von den im Raum verbleibenden Clients mit Nutzer A zwischen zwei Fällen unterschieden. Der Algorithmus verarbeitet die Daten der Clients, die in der n:1-Beziehung die Detail-Rolle sind, nacheinander. Die Anzahl der Durchläufe orientiert sich wie bereits beschrieben an die Anzahl der verbleibenden Clients im Raum. Im weiteren Verlauf des Textes wird der Algorithmus anhand eines im Raum verbliebenen Beispiel-Clients erläutert.
Der erste Fall, der im Listing 4.8 in den Zeilen 3-5 dargestellt ist, tritt ein, wenn bei dem zu bearbeiteten Client in dem Array contacts noch kein Eintrag zu Nutzer A vorhanden ist. Daraufhin wird ein neuer Eintrag angelegt, der die User-ID und die verbrachte Zeit mit Nutzer A beinhaltet. Da die verbrachte Zeit zwischen dem im Raum verbliebenen Client und Nutzer A für beide Seiten identisch ist, kann der Server auf die zuvor berechnete Zeit von Nutzer A zugreifen, um diese in den Daten des Clients zu speichern.
Die Zeilen 6-13 zeigen den Algorithmus, der vom Server ausgeführt wird, wenn in dem Array contacts von dem zu bearbeiteten Client Nutzer A bereits eingetragen ist. In diesem Fall greift der Server auf diese Daten im Array zu und aktualisiert die Zeit des zu bearbeiteten Clients mit der bei Nutzer A bereits berechneten Zeit. Da die verbrachte Zeit zwischen dem Client und Nutzer A aus beiden Sichtweisen im jeden Fall identisch ist, ist die Berechnung der Zeit für einen Nutzer ausreichend.

4.5 Teilen der verbrachten Zeit in einem virtuellen Raum über soziale Netzwerke am Beispiel Facebook

Die Anforderung F250 gibt der Anwendung vor, dass sie dem Nutzer die Möglichkeit bieten soll, auf sozialen Netzwerken die in dem virtuellen Raum verbrachte Zeit zu teilen. In der Systemarchitektur ist die Authentifizierung über Firebase Authentication mit der Möglichkeit sich über das soziale Netzwerk Facebook vorgesehen. Auf Grundlage dieser Tatsache kann die Anwendung auf die Graph API von Facebook zugreifen, wenn sich der Nutzer mit seinem Facebook-Account in der Anwendung anmeldet. Um Zugriff auf die Graph API zu bekommen, muss der Client mithilfe des authentifizierten Facebook-Accounts einen Access-Token des Nutzers erfragen. Mit diesem Access-Token ist die Anwendung befugt, mit dem zugehörigen Facebook-Account bestimmte Funktionen in der Graph API durchzuführen. Damit der Client einen neuen Post im Namen des eingeloggten Facebook-Accounts veröffentlichen kann, muss er dem Nutzer nach der Berechtigung publish_actions fragen und nach Zustimmung ein POST-Request an folgende URL senden: https://graph.facebook.com/**facebook-user-id**/feed?message=**inhalt_der_nachricht**&access_token=**accessToken**. Die fettgedruckten Ausdrücke müssen durch eigene Inhalte ersetzt werden.

4.6 Umsetzung des User Interfaces

Das User Interface ist aufgrund der Tatsache, dass es sich an mobile Browser richtet, in der Sprache CSS umgesetzt. Da das User Interface in dieser Arbeit keine hohe Priorität hat, ist das CSS-Framework BootStrap zum Einsatz gekommen, um in kurzer Zeit ein anständiges User Interface zu realisieren. Im weiteren Verlauf des Textes werden die wichtigsten UI-Merkmale erläutert.
Die Webanwendung richtet sich an Smartphones, weshalb das User Interface für mobile Browser entwickelt ist. Die Eigenschaften und Komponenten des User Interfaces sind gezielt auf ein Bildschirm eines Samsung Galaxy S8 angepasst, wodurch die UI auf Smartphones mit anderen Bildschirmen etwas anders aussehen könnte. Für die Umsetzung sind vorgefertigte Templates von BootStrap für UI-Elemente wie Buttons, Formular, Text oder Eingabefelder zum Einsatz gekommen. Die Darstellung der Anwendung ist in drei Seiten aufgeteilt: Login-Seite, Session- bzw. Timer-Seite und eine Seite für die Anzeige der verbrachten Zeit mit Kontakten.
Ein Aspekt des umgesetzten User Interfaces ist für den Ablauf der Anwendung bedeutsam. Der QR-Code wird als Pop-Up für den Nutzer angezeigt, weil erstens der QR-Code nur zur Einladung angezeigt und anschließend wieder verschwinden soll und zweitens die Anzeige des QR-Codes ohne durch das Verlassen der Timer-Seite geschehen soll. Deshalb bietet sich die Darstellung des QR-Codes in einem Pop-Up an. Mithilfe der BootStrap-Komponente Modal ist die Anzeige des QR-Codes umgesetzt worden, denn sie erzeugt auf Grundlage von JavaScript eine Dialog-Box, die sich über die Website legt und dadurch die aktuelle Seite nicht verlassen wird.
Die finale Version des User Interfaces und wie die einzelnen UI-Komponenten miteinander interagieren und funktionieren wird im nächsten Kapitel anhand eines Beispiels beschrieben.

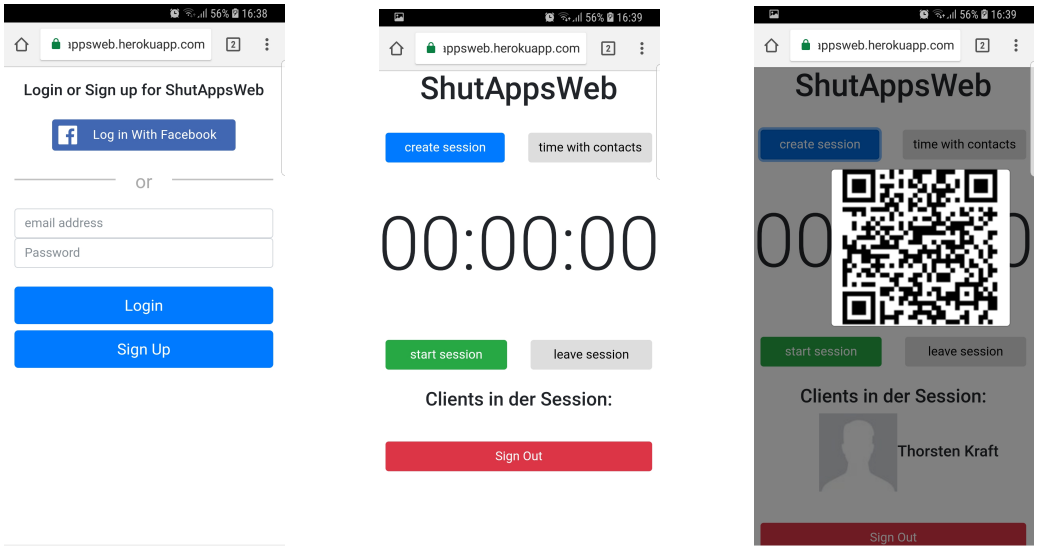
4.7 Verteilung der Webanwendung über Heroku und Zugriff auf den Source Code

Die Anwendung ist in der Entwicklungsphase lokal auf einem PC gestartet worden, wodurch andere Geräte keine Möglichkeit haben, auf diese Anwendung zuzugreifen. Deshalb wird die Webanwendung über die Plattform Heroku bereitgestellt, so dass Endgeräte mit Internetzugang über eine von Heroku generierte URL die Anwendung nutzen können. Für die Verteilung über Heroku ist es wichtig, den Code der Anwendung auf die Server von Heroku hochzuladen, damit es den Code online kompilieren kann. Über folgende URL kann auf die Webanwendung zugegriffen werden: https://shutapps-web.herokuapp.com/

Gegen Ende der Arbeit ist das Problem aufgetreten, dass Facebook die von Heroku generierte URL aus unbekanntem Grund als schädlich eingestuft hat, so dass Facebook relevante Funktionen nicht mehr funktionieren. Um diese Funktionen nutzen zu können, muss der node.js-Server lokal auf einem Computer gestartet werden. Befinden sich Smartphones im gleichen WLAN-Netzwerk kann über die IP-Adresse des Computers und über den Port 8888 auf die Webanwendung zugegriffen werden, in der die Facebook relevanten Funktionen funktionieren. Die folgende Adresse dient als Beispiel: http://192.168.1.13:8888. Die angegebene IP-Adresse variiert je nach Computer. Reicht die Nutzung der Webanwendung ohne die Facebook relevanten Funktionen, kann die oben dargestellte URL von Heroku aufgerufen werden.
Die Einsicht in den Source Code der entwickelten Anwendung kann über die Softwareentwicklungsplattform GitHub über die URL https://github.com/ducle07/shutappsweb und über die mitgesendete CD erfolgen.

5 Prototypische Anwendung der App

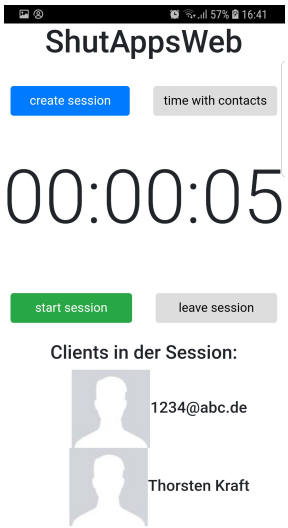
Durch die Entwicklung der Webanwendung ist ein Prototyp entstanden, der die wichtigsten Anforderungen der geplanten Anwendung repräsentiert. In diesem Kapitel soll der Ablauf der Webanwendung aus Sicht eines Nutzers anhand eines Beispiels beschrieben werden. Außerdem wird das fertiggestellte User Interface illustriert und einzelne UI-Elemente erläutert.
In diesem Beispiel sitzt eine Person A mit einem Freund in einem Café und sie wollen sich gegenseitig ihre Geschichten austauschen, da sie sich seit langer Zeit nicht mehr gesehen haben. Weil beide dazu neigen öfters auf Smartphone zu schauen bzw. das Smartphone zu nutzen, wollen beide versuchen, ihre Nutzung des Smartphones beim Treffen zu verringern. Was sich auf dem ersten Blick einfach an hört, stellt sich letztlich doch als schwierig heraus, weshalb sie sich entscheiden mithilfe der entwickelten Webanwendung ihre Smartphonenutzung beim Treffen einzuschränken.
Für die Nutzung der Webanwendung müssen beide Personen sich in der App mit der Erstellung eines Accounts über E-Mail und Passwort oder über Facebook anmelden (siehe Abbildung 5.1(a)). Dadurch werden die Accounts in der Anwendung gespeichert und die Nutzer personalisiert. Nach erfolgreicher Anmeldung werden sie auf die Home-Seite der Webanwendung in der Abbildung 5.1(b) weitergeleitet. Auf dieser Seite haben sie die Möglichkeit über den time with contacts-Button die verbrachte Zeit mit anderen Kontakten anzeigen zu lassen oder über den create session-Button einen neuen Raum zum Starten eines gemeinsamen Timers zu erstellen. Nach der Anfrage zum Erstellen eines neuen Raumes durch Person A wird in der Webanwendung der QR-Code generiert und ihm präsentiert, um andere Nutzer in seinen Raum einladen zu können (siehe Abbildung 5.1(c)). Außerdem wird Person A angezeigt, dass er sich in dem Raum befindet. Durch das Scannen des generierten QR-Codes kann der Freund von Person A den selben Raum betreten, insofern er in der Webanwendung authentifiziert ist.



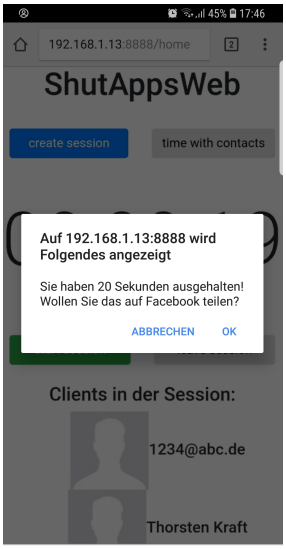
(a) Login-Seite (b) Home-Seite (c) Einladen über ein QR-Code

Abbildung 5.1: Screenshots der entwickelten Webanwendung aus Sicht eines Nutzers

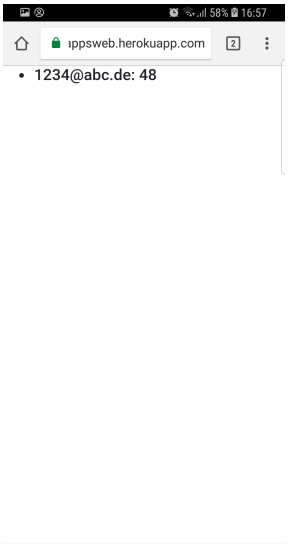
Nachdem Person A seinen Freund in den Raum eingeladen hat, kann er durch das Betätigen des start session-Buttons den Timer für diesen Raum starten. Dadurch fangen auf beiden Smartphones der Timer an hoch zuzählen. Während der Laufzeit des Timers darf die Webanwendung nicht verlassen werden. Sie erkennt, ob die Nutzer durch das Schließen des Browsers oder durch das Verlassen des aktuellen Tabs im Browser die Webanwendung geschlossen haben. Wenn der Fall auftreten sollte, dass sein Freund die Webanwendung verlässt, wird er automatisch aus dem Raum ausgeschlossen, sein Timer gestoppt und die bis dahin gezählte Zeit zwischengespeichert. Diese Zeit kann er nach Zustimmung auf Facebook veröffentlichen lassen, insofern er mit einem Facebook-Account in der Anwendung angemeldet ist (siehe Abbildung 5.2(b)). Person A wird anschließend über das Verlassen des Freundes informiert. Des Weiteren berechnet die Webanwendung die verbrachte Zeit zwischen ihnen und speichert sie auf der Datenbank ab. Diese Zeit können beide Nutzer wie bereits beschrieben durch das Betätigen des time with contacts-Buttons einsehen. Dadurch werden sie auf eine weitere Seite geleitet, auf der ihnen diese Daten präsentiert werden (siehe Abbildung 5.2(c)).



(a) Laufender Timer in der Webanwendung



(b) Teilen der verbrachten Zeit über Facebook



(c) Zeit mit verbrachten Kontakten

Abbildung 5.2: Weitere Screenshots der entwickelten Webanwendung aus Sicht eines Nutzers

6 Fazit

Das strategische Ziel dieser Bachelorarbeit war es, bei Gruppenaktivitäten die Smartphonenutzung unabhängig vom Smartphone und seinem Betriebssystem einzuschränken. Zur Erreichung dieses Ziels wurden taktische und operative Ziele aufgestellt, die in dieser Arbeit behandelt werden. Deshalb wurde eine mobile web-basierte Anwendung entwickelt, die die Smartphonenutzung eines Nutzers bei Gruppenaktivitäten einschränken soll.
Durch die Realisierung dieser mobilen web-basierten Anwendung sind die selbstdefinierten taktischen Ziele dieser Arbeit erreicht worden. Mit dem Einsatz der Webanwendung kann der Nutzer motiviert werden die Smartphonenutzung bei Gruppenaktivitäten einzuschränken. Die entwickelte Webanwendung kann jedoch aufgrund der technischen Beschränktheit zu mobilen Betriebssystemen nicht direkt auf die Funktionen der Smartphones zugreifen, wodurch sie im Vergleich zu verwandten Anwendungen die Smartphonenutzung beispielsweise durch Blockieren anderer Apps nicht direkt regulieren kann.
Kurz vor der Fertigstellung dieser Arbeit hat Facebook aus unbekanntem Grund die von Heroku generierte URL der Webanwendung als schädlich und missbräuchlich eingestuft, weshalb die bis dahin funktionieren Facebook relevanten Funktionen nicht mehr über Heroku genutzt werden können. In der Arbeit wurde bereits eine Alternative zu diesem Problem geschildert.
Für die Fortführung dieser Arbeit ist das Testen und das Evaluieren der entwickelten Anwendung bei verschiedenen Gruppenaktivitäten bedeutsam, um ein aussagekräftiges und repräsentatives Ergebnis zu erhalten, inwieweit die entwickelte Anwendung die Smartphonenutzung tatsächlich einschränken kann. Mit diesen Erkenntnissen kann eine Aussage gemacht werden, inwieweit und ob das strategische Ziel dieser Arbeit erfüllt wurde.
Abschließend lässt sich sagen, dass durch die Entwicklung der mobilen web-basierten Anwendung das in der Arbeit gestellte strategische Ziel weitgehend aber nicht komplett erreicht ist. Erst nach Tests mit Testpersonen in unterschiedlichen Alltagssituationen kann beantwortet werden, ob das strategische Ziel zufriedenstellend erfüllt wurde. Die entwickelte Anwendung kann dem Nutzer bei der Einschränkung der Smartphonenutzung unterstützen. Jedoch kann die Anwendung aufgrund der technischen Beschränktheit zu den mobilen Betriebssystemen die Smartphonenutzung des Nutzers nicht direkt einschränken bzw. kontrollieren. Es liegt letztlich bei den Nutzern, inwieweit sie die entwickelte Webanwendung als Unterstützung nutzen wollen, um die Smartphonenutzung bei Gruppenaktivitäten einzuschränken.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Böhm 2002] Böhm, Rolf: *System-Entwicklung in der Wirtschaftsinformatik.* vdf Hochschulverlag AG, 2002

[comScore 2016] comScore: *Anzahl der Smartphone-Nutzer in Deutschland in den Jahren 2009 bis 2016. Statista.* https://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenutzer-in-deutschland-seit-2010/, 2016. – zuletzt zugegriffen: 15.01.2018

[David 2016] David, Matthew: *How to choose the right MBaaS: Firebase, CloudKit, or Kinvey?* https://techbeacon.com/how-choose-right-mbaas-google-firebase-apple-icloud-or-kinvey, 2016. – zuletzt zugegriffen: 19.01.2018

[David u. Roberts 2017] David, Meredith E. ; Roberts, James A.: Phubbed and Alone: Phone Snubbing, Social Exclusion, and Attachment to Social Media / Baylor University, Texas, USA. 2017. – Forschungsbericht

[Expressjs 2017] Expressjs: *Expressjs.* http://expressjs.com/de/, 2017. – zuletzt zugegriffen: 21.01.2018

[Fielding 2000] Fielding, Roy T.: *Architectural Styles and the Design of Network-based Software Architectures.* https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm, 2000. – zuletzt zugegriffen: 19.01.2018

[Firebase 2018] Firebase: *Firebase Authentication.* https://firebase.google.com/docs/auth/, 2018. – zuletzt zugegriffen: 08.02.2018

[Google 2018] Google, Cloud P.: *Back-End-Dienste für mobile Apps.* https://cloud.google.com/solutions/mobile/mobile-app-backend-services?hl=de, 2018. – zuletzt zugegriffen: 19.01.2018

[Google 2014] Google, Play S.: *KFC Phonestack.* https://play.google.com/store/apps/details?id=com.kfc.stack, 2014. – zuletzt zugegriffen: 04.02.2018

[Hanson 2013] Hanson, Joe: *What Are WebSockets?* https://www.pubnub.com/blog/2013-09-11-what-are-websockets/, 2013. – zuletzt zugegriffen: 18.01.2018

[Kantar 2018] Kantar: *Marktanteile der mobilen Betriebssysteme am Absatz von Smartphones in Deutschland von Januar 2012 bis November 2017. Statista.* https://de.statista.com/statistik/daten/studie/225381/umfrage/marktanteile-der-betriebssysteme-am-smartphone-absatz-in-deutschland-zeitreihe/, 2018. – zuletzt zugegriffen: 16.01.2018

[Kat 2014] Kat: *The UNICEF Tap Project.* https://dontaskmetosmile.wordpress.com/2014/03/02/the-unicef-tap-project/, 2014. – zuletzt zugegriffen: 04.02.2018

[Ko u. a. 2016] Ko, Minsam ; Choi, Seungwoo ; Yatani, Koji ; Lee, Uichin: Lock n' LoL: Group-based Limiting Assistance App to Mitigate Smartphone Distractions in Group Activities / KAIST, Daejeon, South Korea. 2016. – Forschungsbericht

[Le u. Dinh 2017] Le, Duc G. ; Dinh, Vu Phi H.: *Praxisprojekt-Dokumentation: Controlling Mobile Apps in Social Contexts.* 2017

[Löchtefeld u. a. 2013] Löchtefeld, Markus ; Böhmer, Matthias ; Ganev, Lyubomir: AppDetox: Helping Users with Mobile App Addiction / DFKI GmbH, Saarbrücken, Germany. 2013. – Forschungsbericht

[Mercer 2017] Mercer, Alasdair: *QRious.* https://neocotic.com/qrious/, 2017. – zuletzt zugegriffen: 20.01.2018

[o.V. 2015] o.V.: *Jugendwort des Jahres 2015: Smartphone + Zombie = Smombie.* http://www.spiegel.de/lebenundlernen/schule/smombie-ist-jugendwort-des-jahres-a-1062671.html, 2015. – zuletzt zugegriffen: 07.02.2018

[Rouse 2016] Rouse, Margeret: *WebSocket.* http://whatis.techtarget.com/definition/WebSocket, 2016. – zuletzt zugegriffen: 08.02.2018

[Rupp u. Joppich 2014] Rupp, Chris ; Joppich, Rainer: *Requirements-Engineering und -Management.* München : 6. Auflage Carl Hanser Verlag, 2014

[Schulz 2012] Schulz, Thomas: *Interview mit Sherry Tuckle, US-Soziologin und Technologieexpertin, erschienen im Spiegel.* http://www.spiegel.de/spiegel/print/d-86653835.html, 2012. – zuletzt zugegriffen: 07.02.2018

[Sjurts 2011] Sjurts, Insa: *Gabler Wirtschaftslexikon, Stichwort: Smartphone.* http://wirtschaftslexikon.gabler.de/Archiv/569824/smartphone-v1.html, 2011. – zuletzt zugegriffen: 07.02.2018

[socket.io 2015] socket.io: *socket.io.* https://socket.io/, 2015. – zuletzt zugegriffen: 20.01.2018

[Tibbett 2015] Tibbett, Rich: *NoSleep.js.* https://github.com/richtr/NoSleep.js, 2015. – zuletzt zugegriffen: 20.01.2018

[W3C 2017] W3C: *Page Visibility Level 2.* https://www.w3.org/TR/2017/PR-page-visibility-2-20171017/, 2017. – zuletzt zugegriffen: 30.01.2018

[Weidlich 2018] Weidlich, Nina: *Phubbing: Sorry, mein Smartphone ist mir wichtiger als du!* https://www.unicum.de/de/entertainment/netzwelt/phubbing-sorry-mein-smartphone-ist-mir-wichtiger-als-du, 2018. – zuletzt zugegriffen: 08.02.2018

Quellcodeverzeichnis

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, 09. Februar 2018

Duc Giang Le