

Bauhaus-Universität Weimar
Fakultät Medien
Studiengang Medieninformatik

Bauhausboards - Interactive Door Signs for the Office

Bachelorarbeit

André Karge
geb. am: 01.08.1990 in Jena

Matrikelnummer 110033

1. Gutachter: Junior-Prof. Dr. Florian Echtler
2. Gutachter: Prof. Dr. Eva Hornecker

Datum der Abgabe: 24. Dezember 2015

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Weimar, 24. Dezember 2015

.....
André Karge

Zusammenfassung

Diese Bachelorarbeit befasst sich mit dem Entwurf, der Umsetzung und dem Test von personalisierten digitalen Türschildern zur Präsentation und Kommunikation im Bürobereich der Bauhaus Universität Weimar (BUW), genannt Bauhausboards.

Ziel ist es, eine Anwendung zu erstellen, mit der Nutzer in Büros individuelle Daten präsentieren können.

In den meisten Büros sind dazu Pinnwände oder Whiteboards angebracht. Diese bieten jedoch nicht die Möglichkeit den Inhalt anzupassen oder hinterlassene Nachrichten zu lesen, wenn man nicht persönlich vor Ort ist.

Zu diesem Zweck wurde eine Web-Applikation entworfen, welche auf, neben Büroeingängen aufgehängenen, Tablet-PCs angezeigt wird und mit denen Nutzer im Büro, sowie vorbeigehende Nutzer interagieren können. Diese muss leicht bedienbar und zusätzlich von Desktop-PCs oder Smartphones benutzbar sein. Nutzer in den Büros können vom Büro aus oder von unterwegs individuelle Daten präsentieren, sowie ihren aktuellen Status aktualisieren.

Besucher können dadurch vor Ort über aktuelle Arbeiten, kurzfristige Abwesenheit oder andere interessante Informationen in Kenntnis gesetzt werden. Zudem haben sie die Möglichkeit bestimmten Nutzern des Raumes Nachrichten zu schreiben.

Die Applikation wird, um Interaktion und Benutzbarkeit zu erproben, einem Testlauf unterzogen und das Ergebnis anschließend ausgewertet.

Inhaltsverzeichnis

1 Einleitung	1
2 Verwandte Arbeiten	3
2.1 Hermes	3
2.2 NetBoards	6
2.3 Andere	8
3 Vorstudie	10
3.1 NetBoards Experiment	10
3.2 Auswertung	12
4 BauhausBoards	15
4.1 Entwurf	15
4.1.1 Systementwurf	15
4.1.2 Interface-Entwurf	18
4.1.3 Datenbankentwurf	21
4.2 Umsetzung	23
4.2.1 Funktionen	23
4.2.2 Editor	26
4.2.3 Probleme	28
4.3 Vergleich zu Hermes und NetBoards	29
5 Studie	30
5.1 Entwurf	30
5.2 Durchführung	31
5.3 Auswertung	31
6 Ausblick	32
6.1 Zusätzliche Features	32
6.2 Nutzung	32
Literaturverzeichnis	33

Kapitel 1

Einleitung

In Forschungseinrichtungen und in Unternehmen arbeiten viele Menschen mit unterschiedlichen Aufgaben und Tätigkeiten. Die meisten wissen jedoch nichts darüber was andere Mitarbeiter machen, wenn sie nicht direkt mit ihnen zusammen arbeiten. Manche Mitarbeiter wollen ihre Kollegen oder Gäste über ihre aktuelle Arbeit, wichtige Termine oder kommende Veranstaltungen informieren. Andere wollen gern den Inhalt ihrer letzten wissenschaftlichen Ausarbeitung, die Ergebnisse der letzten Konferenz, Noten einer bestimmten Prüfung oder einfach nur private Daten präsentieren. Um diese Informationen zur Verfügung zu stellen werden in den meisten Büros Pinnwände oder Whiteboards neben den Türen aufgehängt (Abb. 1.1). Um kurze Nachrichten zu hinterlassen kleben Leute Post-Its an ihre Tür oder hängen Do-Not-Disturb Schilder an den Türknauf, wenn sie nicht gestört werden wollen.



Abbildung 1.1: Typische Pinnwand der Medienfakultät der BUW

Aktuell müssen die Personen des Raumes solche Informationen direkt an die Wand schreiben oder ausdrucken und danach dort aufhängen. Wenn man aber nicht persönlich vor Ort ist, kann man die Daten an der eigenen Pinnwand nicht selber anpassen. Es kann zu Problemen kommen, wenn man einen Termin vereinbart hat und sich dann verspätet, wenn man krank geworden ist und deswegen den ganzen Tag nicht anzutreffen ist oder auf einer Reise ist und andere am Arbeitsplatz direkt daran teilhaben lassen will.

Pinnwände und Whiteboards bieten zudem auch nicht die Möglichkeit digitale Informationen, wie Videos, Animated-Gifs oder die neuesten Twittermeldungen anzuzeigen.

Eine Lösung für diese Probleme ist die Anbringung eines digitalen Türschildes, mit dem Nutzer direkt oder aus der Ferne interagieren können.

Zu Beginn galt es zu klären, welchen Weg ich mit dieser Arbeit einnehmen sollte. Zum einen konnte ich ein bereits vorhandenes Projekt hernehmen, anpassen und verbessern oder ein eigenes System von Grund auf selbst entwerfen und umsetzen. Deshalb führte ich ein Experiment mit dem NetBoards System[1, 2] durch um herauszufinden, ob dies ein geeignetes System für die Verwendung an der Medienfakultät der BUW wäre. Zudem ging es darum die Bedürfnisse und das Verhalten der Benutzer im Bezug zu interaktiven Türschildern zu ermitteln.

Nach der Durchführung des Experiments stellte sich heraus, dass das NetBoards System nicht der richtige Ansatz für die Umgebung war. Deswegen entschied ich mich dazu ein eigenes System zu entwerfen.

Die Entwicklung meines Systems teilte sich in drei Abschnitte: Planungsphase, Implementationsphase und Studienphase. Die Planungsphase diente dazu die Systemumgebung und den Aufbau der Applikation zu definieren. Darunter fielen die Wahl des Anwendungstyps und der Programmiersprache, sowie der Entwurf einer Datenbank und des Benutzerinterfaces.

In der Implementationsphase befasste ich mich damit, die geplanten Ideen in ein lauffähiges Programm umzusetzen.

Das Programm musste nach seiner Fertigstellung getestet werden. Deswegen habe ich in der letzten Phase eine Studie geplant und durchgeführt. Dafür habe ich in der Medienfakultät der BUW vor bestimmten Büros Tablets aufgehängt, die Interaktionen damit archiviert und im Anschluss die Büro-Insassen befragt. Die Ergebnisse der Befragung wurden ausgewertet und fließen in die nächste Version des Programms.

Kapitel 2

Verwandte Arbeiten

Die Idee für interaktive Türschilder wurde schon in verschiedenen wissenschaftlichen Ausarbeitungen sowie in der Praxis aufgefasst. Die Gruppe um Keith Cheverest et al. von der Universität Lancaster hat mehrere Paper für ihr Türschild-System “Hermes”[3] bereits ab 2003 veröffentlicht. Das Projekt “NetBoards” von Errol Wood[1] von der Universität Cambridge behandelt eine aktuellere Version interaktiver Türschilder. Zudem gab es noch einige andere Arbeiten, die sich mit ähnlichen Ansätzen befassten und teilweise als Grundlage für die beiden Projekte dienten.

2.1 Hermes

Das Hermes System[3–5] ist eines der ersten Versionen interaktiver digitaler Türschilder im Bürobereich. Es diente dazu herauszufinden, ob die „traditionelle Methode Nachrichten in einem halbwegs öffentlichem Raum (wie beispielsweise vor Büros in einem Forschungsinstitut) mittels Post-It Zetteln zu hinterlassen durch eine digitale Methode verbessert werden könnte“[3, 4]. Zu diesem Zweck haben die Forscher ein digitales asynchrones Nachrichtensystem entwickelt, welches direkte Interaktion mittels einem vor den Büros angebrachten Interfaces, sowie Fernzugriff durch ein Web-Portal und SMS ermöglicht. Das System besteht aus einem Web-Server und mehreren PDAs, welche an die Wände neben den Büroeingängen geschraubt werden (Abb. 2.1). Der Server wurde mit Java-Servlets realisiert und generiert HTML Webseiten, die auf den Displays angezeigt werden. Er bietet eine zentrale Datenbank für alle Daten des Systems und dient als Kommunikationseinheit mit einem SMS Gateway. Eines der Hauptblickpunkte von Hermes war, dass das System leicht einsetzbar ist. Deswegen fiel die Entscheidung zur Wahl der Geräte auf PDAs. Zu der Zeit, als das Paper entstand waren PDAs die beste Wahl für kleine, handliche Geräte, welche per WLAN mit einem Server kommunizieren können. Zudem



Abbildung 2.1: Das erste Hermes Display[3]

mussten die Geräte mit der Umweltpolitik der Universität übereinstimmen und sollten daher nicht viel Energie verbrauchen. Um die Geräte vor Diebstahl zu schützen wurden Aluminiumhüllen für die PDAs entworfen, welche neben den Büroeingängen an die Wand geschraubt werden konnten und zudem auch als Diebstahlsicherung dienten. Um ungewünschte Interaktion zu verhindern wurden die Hülle so konzipiert, dass die Tasten nicht direkt zugänglich waren. Die Funktionen von Hermes wurden in zwei Perspektiven aufgeteilt: Die Besitzerperspektive und die Besucherperspektive.

Besitzerperspektive

„Diese Perspektive ermöglicht dem Besitzer des Displays Nachrichten oder Bilder zu erstellen, die dann direkt auf dem PDA angezeigt werden oder Nachrichten zu lesen, die von Gästen für ihn hinterlassen wurden“[3]. Zudem kann der Nutzer animierte Gifs hochladen. Der Nutzer kann die Perspektive von einem PC oder direkt am Display aufrufen, was ihm ermöglicht schnell beim Verlassen des Büros eine Nachricht zu schreiben. Um sich zu authentisieren muss er entweder seinen Nutzernamen und sein Passwort eingeben oder sich mit seinem iButton „Ein iButton ist ein Mikrochip in einer Metallhülle und dient beispielsweise zum Authentisieren des Besitzers“[6] anmelden.

Besucherperspektive

„Besucher müssen sich vor dem PDA befinden, um mit dem System interagieren zu können“[3]. Sie können dem Besitzer des Raumes Nachrichten schreiben. Andere Besucher können jedoch nur die Nachricht sehen, die der Besitzer des Raumes eingestellt hat. Die Nachrichten von anderen Besucher kann nur der Besitzer einsehen, wodurch ein Vorteil im Bereich der Privatsphäre gegenüber Post-It Zetteln erzielt wurde.

Um das System zu testen haben die Entwickler eine Langzeitstudie über 15 Monate durchgeführt. Dabei haben sie im Vorfeld Bedenken geäußert, dass „die Nutzer Zeit brauchen bis sie neue Technologien in ihre tägliche Routine eingliedern“[3]. Eines der enttäuschensten Ergebnisse war, dass es Nutzer gab, die trotz aufgehängtem Hermes Display weiterhin Post-Its an die Türen gehangen haben[3]. Ein Ergebnis der Studie war, dass die Nutzer erst Vertrauen aufbauen müssen. „Nutzer nutzen ein neues System erst, wenn sie wissen, dass es funktioniert“[3]. Durch Verbindungsprobleme über das WLAN kam es dazu, dass das System ab und an nicht reagierte, wodurch es für die Nutzer schien, das Gerät wäre abgestürzt. Um dies zu beheben wurde das Metallgehäuse an der Stelle des WLAN Adapters geöffnet, damit das Gehäuse das Signal nicht weiter abschwächen konnte. Es kam auch vor, dass Besucher vergaßen ihre Nachricht abzuschicken, wodurch deren Nachricht nicht mehr privat war und andere Nutzer sie sehen konnten. Das ermöglichte anderen Nutzern die Nachricht zu verändern oder sie zu missbrauchen um unangebrachte Nachrichten zu schreiben. Wenn der eigentliche Benutzer sich zudem auch authentisiert hat wurde die Nachricht dann in seinem Namen geschickt[4]. Manche Nutzer fanden es frustrierend nachdem sie eine temporäre Mitteilung eingestellt hatten eine alte Nachricht wiederherzustellen, da sie wieder durch den Prozess einer neuen Mitteilung gehen mussten. Als Lösung dafür wurde eine Möglichkeit eingebaut, mit der die Nutzer eine Standardnachricht einstellen konnten, die nach entfernen der temporären Nachricht wieder angezeigt wurde[4].

Um das Hermes System zu erweitern wurde später ein Foto-Display (Abb. 2.2) entwickelt[5]. Da die Hermes-Displays sehr klein waren wurde dieses Display ausschließlich für Bilder verwendet. Mit dieser Erweiterung war es den Nutzern möglich Bilder per MMS oder Bluetooth an das Gerät zu schicken oder mit der vorhandenen Displaykamera ein Bild zu schießen.

Da dieses System schon 2003 entwickelt wurde, dienten dessen Ausarbeitungen und Forschungsergebnisse als Inspiration oder Grundlage für andere Projekte, wie beispielsweise für das NetBoards System.



Abbildung 2.2: Das erste Hermes Foto-Display[7]

2.2 NetBoards

NetBoards[1, 2] ist ein weiteres interessantes Projekt im Bereich interaktiver digitaler Türschilder. Es wurde 2014 an der Universität von Cambridge entwickelt und zielte darauf hin „die bereits vorhandenen Whiteboards und Pinnwände neben den Universitätsbüros mit großen, Touch-fähigen Bildschirmen zu erweitern oder zu ersetzen“[1].

Im Vergleich zum Hermes System, welches schon im Jahr 2003 veröffentlicht wurde, hatte sich einiges bezüglich Hardware geändert. „Displays sind billiger, größer, hochauflösender und energieeffizienter geworden“[1]. Dadurch konnten für NetBoards größere Bildschirme mit höherer Auflösung (Abb. 2.3) verwendet werden als beim Hermes System.

Die ersten Prototypen des NetBoard Systems bestanden jeweils aus einem 22 Zoll großem Touchscreen Monitor, der von einem Raspberry Pi „Ein Raspberry Pi ist ein kompakter Einplatinencomputer“[8] betrieben wurde. Im ersten Experiment der Entwickler wurden fünf Prototypen verwendet um herauszufinden, ob die Boards akzeptiert werden und um typische Nutzerverhalten zu bestimmen. „Diese wurden vor den Büros von Doktoranden, Post-Doktoranden und eines Professors aufgehängt, um herauszufinden, wie diese unterschiedlichen Mitarbeiter mit dem System umgehen würden“[1] „Die Probanden konnten bei diesem Experiment Skizzen und Nachrichten zeichnen und Bilder per Web-Interface hochladen“[1]. Zudem war es möglich Webseiten zu laden,

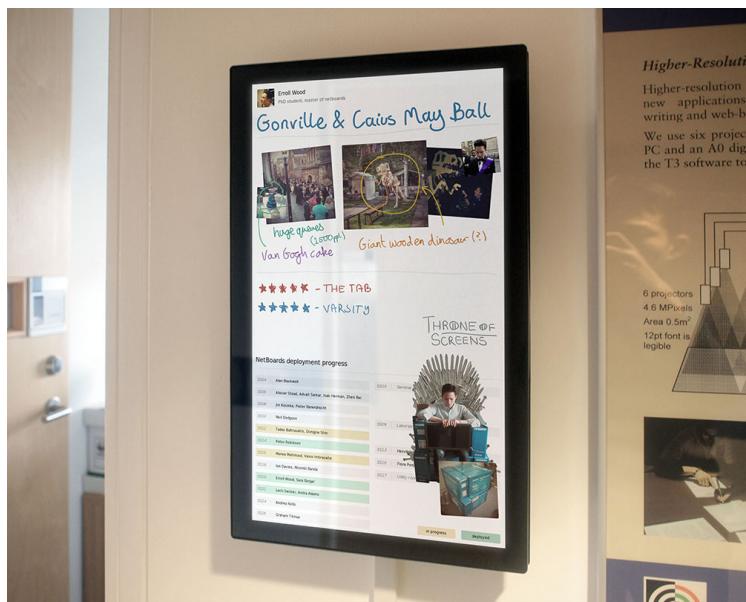


Abbildung 2.3: Das NetBoards Display[1]

die dann in einer Hintergrundschicht angezeigt werden konnten.

Während des Experiments fanden die Entwickler heraus, dass die Boards schnell Aufmerksamkeit erzeugten. „Vorbeigehende Leute begannen spontan mit den Bildschirmen zu interagieren oder sogar die Besitzer direkt auf die Boards anzusprechen“[1]. Als Ergebnis des Experiments fanden die Entwickler heraus, dass zwei wichtige Faktoren die Akzeptanz der Nutzer gegenüber den Boards beeinflussen:

- **Zuverlässigkeit:** „Nutzer werden ein System nur benutzen wenn sie denken es funktioniert“[1]
- **Benutzerfreundlichkeit:** „Angewöhnung der Nutzer an das System ist unwahrscheinlich, wenn das System schwer zu benutzen ist oder zu viel lernen erfordert“[1]

Dadurch, dass für die Prototypen Touchscreens mit SAW-Technologie¹ eingesetzt wurden, war die Interaktion mit den Boards sehr ungenau, wodurch die Benutzerfreundlichkeit stark beeinträchtigt wurde. Zudem waren die Raspberry Pi's zu schwach, um ein funktionreiches Interface zu betreiben.

Nach dem Experiment passten die Entwickler ihr System an. Sie ersetzten die

¹Surface-Acoustic-Wave (dt.: Akustische Oberflächenwelle) ist eine Technik um die Position von Fingern auf Displays zu ermitteln.

Raspberry's mit leistungsfähigeren Maschinen, die jeweils in der Lage waren zwei Monitore simultan anzusprechen. Um es von überall erreichen zu können wurde das NetBoards-System mit Web-Technologien implementiert. „Es musste möglich sein, mit mobilen Telefonen oder Tablet-PC's das System zu erreichen“[1]. Der Server bestand deswegen aus einem einfachen Webserver, der beim Aufruf eine HTML Webseite auslieferte. Frontend Funktionen wurden per Javascript realisiert, wobei die Ansicht der Displays in drei Schichten aufgeteilt wurde[1]:

- **UI (Benutzerinterface)**
für Interface Widgets und Details über die Bewohner des Büros
- **Board Content (Board-Inhalt)**
für gezeichnete Nachrichten, Skizzen und Bilder
- **Web-Page Background (Hintergrund Webseite)**
zur Anzeige einer beliebigen Webseite

Mit diesen neuen Spezifikationen wurde ein Testlauf über mehrere Monate durchgeführt. „Die Boards erzeugten in diesem neuen Test erhöhte Aufmerksamkeit und die Besitzer nutzten sie mehr, um ihre Kollegen von ihren Abwesenheiten zu informieren, selbst wenn sie nicht selbst vor Ort waren“[1]. „Im ersten Test änderten die Nutzer nur den Inhalt ihres eigenen Boards, was sich im zweiten Test änderte. Sie begannen auch den Inhalt der Boards ihrer Kollegen zu ändern.“[1]

Da NetBoards eines der neuesten Systeme für interaktive Türschilder ist, flossen in die Ausarbeitung manche Teile von früheren Forschungen ein. Unter anderem aus dem Hermes System, aber auch von diversen anderen Projekten.

2.3 Andere

Diese wissenschaftlich Arbeiten befassen sich teilweise mit dem Konzept von interaktiven Türschildern oder wurden nur theoretisch erarbeitet.

Im wissenschaftlichen Papier “Dynamic Door Displays” von David Nguyen et al. ging es darum eine Möglichkeit zu entwickeln, um die natürlichen Anzeigefähigkeiten von Bürotüren zu erweitern. Dies umfasste automatische Updates und maßgeschneiderte Displays zur Präsentation von privaten Informationen[9]. Die Ausarbeitung ähnelte in gewisser Hinsicht dem Hermes-System, nur das anstelle eines PDA's einen Hewlett-Packard 620LX Handheld-Computer benutzten. Leider wurde nur ein Prototyp entwickelt, danach aber nicht weiter

geführt. Jedoch bot dieses Projekt einige Grundlagen für Hermes, sowie NetBoards.

In der Arbeit “Semi Public Displays for Small, Co-located Groups” von Elaine M. Huang et al. ging es um Displays in halb öffentlichen Räumen. „Das System beinhaltete die Bereitstellung von Informationen an abgeschiedenen Orten oder für Leute, die keine Ahnung über die Arbeiten ihrer Kollegen haben.“[10] Es bot mehrere untereinander unabhängige Funktionen für die Darstellung und Interaktion auf peripheren Displays an. Im NetBoards Projekt wurde jedoch entschieden, Funktionen zusammenzuführen, wodurch beispielsweise die Anzeige einer statischen Webseite mit der Funktion eines Editors für Skizzen verbunden wurde.

Die wissenschaftliche Ausarbeitung “UniCast, OutCast & GroupCast: Three Steps Toward Ubiquitous, Peripheral Displays” von Joseph F. McCarthy et al. handelt von drei verschiedenen Display-Varianten: OutCast - Display außerhalb eines Büros, UniCast - Display in einem Büro und GroupCast - Display in einem Gemeinschaftsbereich[11]. Die OutCast Displays sind eine weitere Variante von interaktiven Türschildern im Bürobereich. Sie wurde dazu genutzt um Biographien, Kalender-, Positions-, Projektinformationen, Demonstrationen oder Text Nachrichten anzeigen zu lassen. Dieses Projekt war eines der ersten Systeme für die Anbringung von Bildschirmen außerhalb von Büros zur Informationsdarbietung. Dadurch hatte es auch gewissen Einfluss auf die späteren Entwicklungen.

Kapitel 3

Vorstudie

Zu Beginn war es wichtig eine Anforderungsanalyse durchzuführen. Deshalb musste analysiert werden, in welchem Rahmen sich diese Arbeit bewegen sollte. Es bestanden die Optionen, ein fertiggestelltes Projekt anzupassen und zu verbessern oder ein vollkommen eigenes System zu entwickeln.

Um zu erproben, wie ein bereits fertiges System funktioniert und wie es von Studenten und Mitarbeitern der Medienfakultät der BUW aufgenommen werden würde, habe ich ein bereits vorhandenes Projekt hergenommen und in der Praxis getestet. Da das Hermes System schon relativ alt war und hauptsächlich für PDA's entworfen wurde, kam es für den Test nicht in Frage. Die Entscheidung fiel auf die zweite Version des NetBoards Projekts[2], da dieses Projekt das aktuellste war.

3.1 NetBoards Experiment

Als Grundlage für mein Experiment diente ein virtueller Debian-Linux Server mit 1,5GHz AMD CPU und 2GB RAM auf dem Apache2 als Webserver und Python2 für serverseitige Scripts liefen. Das öffentlich zugängliche NetBoards2 Projekt wurde installiert und konfiguriert.

Nachdem der Server eingerichtet war, kam die Frage auf, was als Anzeigegerät dienen sollte. Das originale Projekt von E. Wood wurde für 22 Zoll Monitore mit Touch-Oberfläche entworfen. Diese wurden vertikal neben die Büroeingänge gehangen.

Da für mein Experiment nicht die notwendigen Ressourcen vorhanden waren um den genauen Versuchsaufbau nachzuempfinden, fiel die Wahl des Anzeigegerätes auf einen kostengünstigen Tablet-PC.

Dies war ein 10 Zoll Tablet mit 1GHz Cortex A8 Dual Core CPU, 1GB RAM und Android Version 4.4.

Damit die Benutzer nicht mit den auf dem Gerät installierten Applikatio-

nen interagieren konnten, wurde eine Kiosk-Applikation¹ installiert. Diese App schränkt die Interaktionsmöglichkeiten der Nutzer nur auf eine ausgewählte Webseite ein. Nur der Administrator des Gerätes hatte die Möglichkeit diese zu ändern oder die Applikation ganz zu beenden.

Ein weiteres Problem war die Befestigung des Displays. Ich entschied mich dafür, das Tablet an den bereits vorhanden Türschildern anzubringen (Abb. 3.1) .



Abbildung 3.1: Ein typisches Türschild in der Medienfakultät

Dadurch, da sich die Rückwand des Tablets abschrauben ließ, konnte ich zwei Löcher hinein bohren. Diese dienten zur Anbringung eines Drahtes, welcher über das Türschild gehangen werden konnte. Für dieses Experiment war die Aufhängung vollkommen ausreichend.

Für die Stromversorgung wurde das Ladekabel des Tablets verlängert, damit es an eine Steckdose im Büro angeschlossen werden konnte.

Als Testnutzer für das Experiment konnte ich zwei Mitarbeiter der Professur für Webtechnologien und Informationssysteme organisieren. Diese Nutzer teilten sich ein Büro, wodurch nur ein Anzeigegerät benötigt wurde. Nach einer Einweisung zur Benutzung des Systems wurde klar, dass die zwei Tester nicht damit einverstanden waren, dass durch die Interaktivität des Boards je-

¹<http://www.android-kiosk.com>

der vorbeigehende Guest den dargestellten Inhalt nach belieben ändern konnte. Es hätte sein können, da durch die Anonymität der Gäste, das Board missbraucht werden könnte, um unangebrachte Skizzen zu zeichnen. Aus diesem Grund habe ich für die Nutzer zwei verschiedene Ansichten erstellt:

- Eine Backend-Sicht, auf der sie ihre Änderungen machen konnten.
- Eine Frontend-Sicht, die auf dem Tablet angezeigt wurde, nach 5 Minuten den aktuellen Stand abspeicherte und die Sicht auf den aktuellsten Stand der Backend-Sicht setzte.

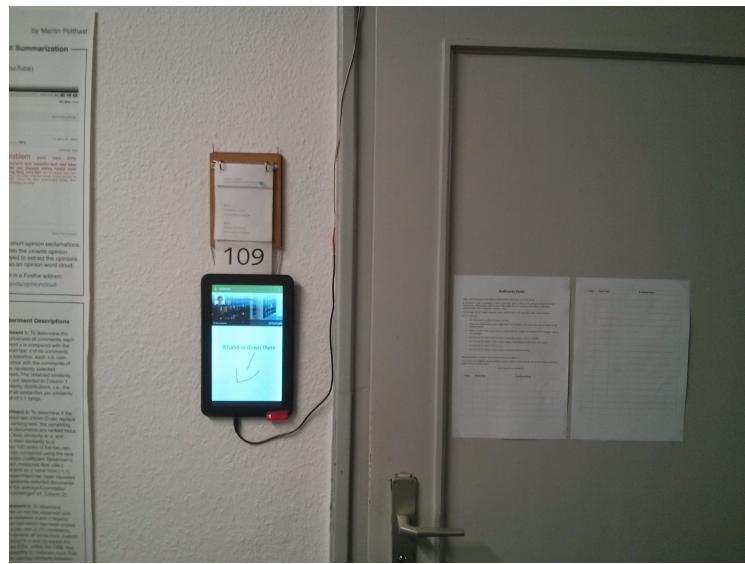


Abbildung 3.2: Versuchsaufbau des Experiments

3.2 Auswertung

Das Experiment lief 12 Tage. Die Nutzer erstellten sich zu Beginn einen Nutzeraccount und richteten ihr Board ein. Jeder der beiden lud ein Profilbild hoch und gab seinen Namen, sowie seinen akademischen Grad an. Einer der Tester benutzte das Display um das Banner der Professur zu präsentieren (Abb. 3.3(a)) und um Gäste beispielsweise darüber zu informieren, dass er sich zur Zeit im Büro befand (Abb. 3.3(b)).

Positiv war, dass das Board viel Aufmerksamkeit zu erzeugen schien. Viele vorbeigehende Nutzer blieben stehen, um sich das Display genauer anzusehen und interagierten sogar damit. Dadurch entstanden viele Zeichnungen, wovon viele leider keinen tieferen Sinn hatten wie beispielsweise die Zeichnung in Abb.

3.3(c). Jedoch gab es auch ab und an Nachrichten, die direkt an die Besitzer des Boards gerichtet waren, wie das Bild einer Kaffeetasse mit einem Fragezeichen aus Abb. 3.3(d), was die Frage nach einer Kaffeepause darstellen sollte.



Abbildung 3.3: Einige Beispielbilder

Nach Beendigung des Experiments führte ich ein Gespräch mit den Testnutzern, sowie mit einigen Gästen.

Es stellte sich dabei heraus, dass durch die geringe Auflösung des Tablets den meisten Besuchern gar nicht bewusst wurde, dass zwei Nutzer auf dem Board angemeldet waren. Man musste manuell in der Ansicht scrollen, um den Inhalt des zweiten Nutzers einsehen zu können. Jedoch war auch das ein Problem, da Scrolling nur möglich war, wenn die Sidebar ausgeblendet war. Die meisten Nutzer interagierten demnach nur mit der oberen Hälfte des eigentlichen In-

halts.

Ein weiteres Problem war, dass das Tablet eine sehr geringe Leistung hatte und das Touch-Interface nicht sehr genau war. Dadurch waren Interaktionen mit dem Board sehr langsam und ruckelig.

Positive Äußerungen gab es bezüglich angezeigter Statusmeldungen wie beispielsweise "I am inside". Die meisten Besucher fanden diese Meldung hilfreich, da sie deshalb wussten, ob der Mitarbeiter zur Zeit im Raum war. Weil zum Empfangen einer Benachrichtigung eine Browser-App erforderlich war, was den Testern nicht bewusst war, hatten sie keine Möglichkeiten darüber benachrichtigt zu werden, wenn jemand etwas auf ihrem Board änderte. Es wäre besser gewesen, wenn die Besitzer eine Email erhalten hätten, sobald ihnen jemand etwas mitteilen wollte.

Als Schlussfolgerung des Experimentes entschied ich mich dazu ein eigenes System zu entwerfen, da es mehr Sinn machte es von Grund auf nach den Anforderungen zu entwickeln, anstatt das NetBoards System so anzupassen, dass es die Anforderungen erfüllte. Dieses sollte gut auf Tablets laufen und für eine kleinere Bildschirmgröße angepasst sein. Es erforderte, dass jeder Nutzer eine eigene Sicht bekommen sollte, da ansonsten auf dem Tablet zu wenig Platz wäre. Es sollten nur noch die Besitzer die Möglichkeit haben, den angezeigten Inhalt zu ändern. Den Besuchern sollte es dennoch möglich sein, mit den Besitzern in Kontakt zu treten, welche darauf eine Benachrichtigungs-Email erhalten sollten.

Kapitel 4

BauhausBoards

Das neue Projekt bekam den Namen “BauhausBoards”. In einer Entwurfsphase wurde entschieden, wie das System aussehen und welche Funktionen es bieten sollte. Dabei flossen die Ergebnisse der Vorstudie und anderer Projekte mit ein.

Als Voraussetzung sollten als Anzeigegeräte für die Türschilder weiterhin Tablets genutzt werden. Der Entwurf wurde darauf in ein Programm umgesetzt, wobei manche geplante Funktionen geändert, entfernt oder neue hinzugefügt wurden. An manchen Stellen der Umsetzung kam es auch zu Problemen, die es zu lösen galt.

TODO: Fußnote für lib-quotations

TODO: Literaturverzeichnis ein wenig zu kurz -> vllt noch deren related work noch ein wenig analysieren

TODO: Quellcode vom System dann auf CD im Endeffekt

4.1 Entwurf

4.1.1 Systementwurf

Die Planung des Systems begann damit, die Plattform des Systems auszuwählen. Es gab die Möglichkeit eine Android-Applikation zu erstellen, die auf jedem Tablet hätte installiert werden müssen. Dabei gäb es jedoch das Problem, dass die Nutzer auch von unterwegs mit der Anwendung interagieren sollten, wofür ein eigenständiger Server für die Datenverwaltung notwendig gewesen wäre. Außerdem müsste durch die unterschiedlichen Tablet- und Smartphone-Betriebssysteme neben Android, wie Apple IOS oder Windows Mobile, die App für jede Architektur portiert werden.

Deswegen fiel, wie bei NetBoards, die Wahl auf eine Web-Applikation. Zum Einen, weil jedes Tablet, jedes Smartphone und jeder PC Webseiten darstellen

kann und es nicht nötig ist, extra eine App auf den Geräten zu installieren. Ein zentraler Webserver generiert eine Webseite mit allen nötigen Frontend Funktionen, die von allen Geräten mit Web-Browser (Clients) angezeigt werden können. Dadurch sollte sich der Programmieraufwand auf Server- und Client-Funktionen einschränken.

Die Applikation sollte nicht bei jeder kleinsten Interaktion mit der Webseite diese neu laden müssen. Das hätte einen viel zu großen Overhead erzeugt, da bei jeder Kommunikation statische Daten übertragen worden wären. Deswegen musste der Server nur beim ersten Aufruf das Grundgerüst der Webseite mit allen Client-Funktionen ausliefern. Ein Teil der Funktionen würde dann Daten nachladen oder an den Server schicken, wodurch ein dynamischer Inhalt erzeugt wird. Dies sollte mit dem “CRUD” (Create, Read, Update and Delete) Prinzip realisiert werden. Der Server wartet, nachdem er eine Seite ausgeliefert hat, auf spezifische Anfragen des Clients und liefert dementsprechend Daten aus (Read) oder ändert den Zustand der Daten (Create, Update und Delete). Zudem gewährleistet dieses Prinzip, dass bei Verbindungsabbrüchen die Seite auf dem Client weiter läuft und nur so lange keine Daten ändern oder nachladen kann, bis die Verbindung wieder aufgebaut wurde.

Die anfallenden Daten des Systems mussten in irgend einer Form auf dem Server gespeichert werden. Die beste Methode dafür war die Nutzung einer Datenbank. Anders als bei NetBoards, wo alle Daten als JSON¹-Dateien auf dem Server gespeichert wurden, sah ich diese Methode als geeigneter an. Mit einer Datenbank war es möglich Relationen besser darzustellen, Konsistenz zu bewahren und Redundanz zu vermeiden. Zudem sind damit Daten zentralisiert zusammengefasst, wodurch eine bessere Suche möglich ist.

Die Wahl des Datenbanksystems fiel auf “SQLite”[13], da es das am meisten verbreitete relationale Datenbanksystem der Welt ist. Es war einfach zu benutzen und hatte eine gute Dokumentation.

Das Gerüst der Webseite sollte auf HTML basieren, wobei die hauptsächlichen Client-Funktionen üblicherweise in Javascript umgesetzt werden. Da die Clients viele Funktionen bieten und der Server nur zum Ausliefern der Webseite und für den Datenaustausch dienen sollte, machte es Sinn für den Server “Node.js” zu verwenden. „Node.js ist serverseitiges Javascript basierend auf Google Chrome’s V8 Javascript Engine“[14] .

In der Abb. 4.1 ist der Aufbau des Systems zusammengefasst. Auf dem Server befinden sich der Webserver und die Datenbank. Der Webserver stellt Anfragen an die Datenbank und erhält dafür entsprechende Ergebnisse zurückgeliefert. Wenn ein Client die Adresse des Servers aufruft generiert der Webserver die

¹ „JSON ist ein, für Menschen leicht zu lesendes und Maschinen leicht zu parsendes, Datenaustauschformat“[12]

Webseite und sendet sie mit allen benötigten Funktionen an den Client. Solange die Seite nicht explizit neu geladen werden soll, kommuniziert der Client danach nur durch CRUD Interaktionen mit dem Server.

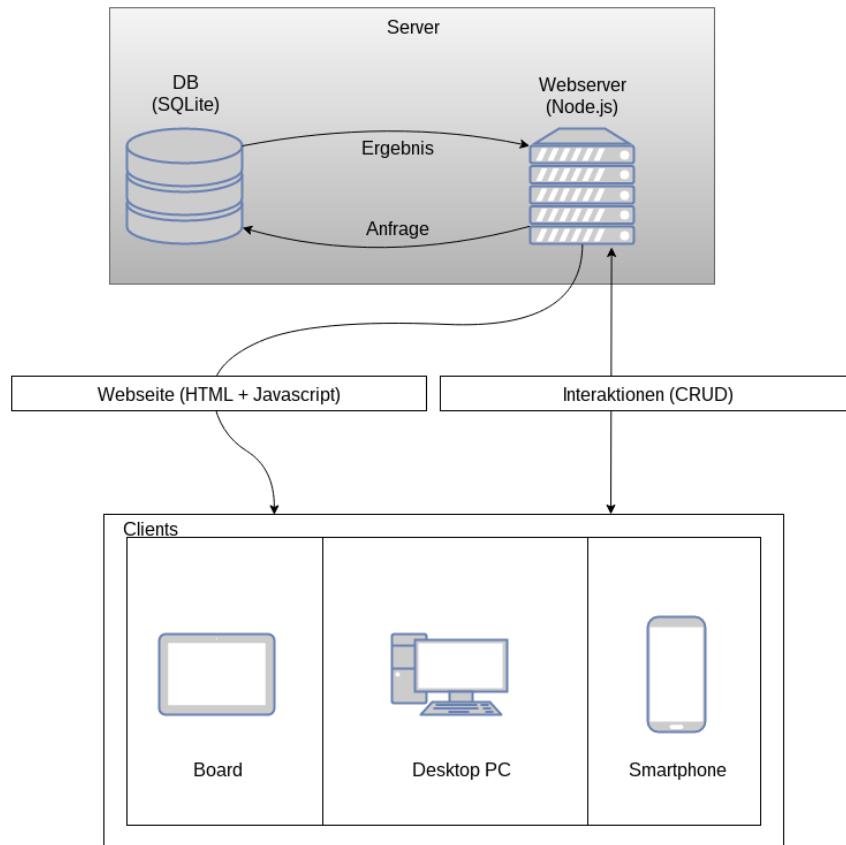


Abbildung 4.1: Bauhausboards - Systemaufbau

Da die Nutzer die Möglichkeit bekommen sollten, individuellen Inhalt präsentieren zu können, musste ein geeigneter Editor programmiert werden. Damit sollte es möglich sein, einfache Zeichnungen anzufertigen, Texte zu schreiben oder Bilder hochladen zu können. Es war wichtig, dass der Editor die Funktionen eines Whiteboards in manchen Zügen übernehmen oder möglicherweise sogar verbessern konnte.

Folgende Whiteboard-Funktionen musste der Editor daher abbilden:

- Zeichnen mit verschiedenfarbigen Stiften
- Entfernen von Zeichnungen mit einem Schwamm
- Anbringung von Bildern oder Ausdrücken per Magnet

- Neuanordnung von aufgehängten Elementen

Als Grundlage für den Editor entschied ich mich für die Javascript Bibliothek “Paper.js”[15]. Es ist ein Open Source Framework für die Darstellung und Manipulation von Vektorgrafiken, welches auch die Grundlage des Editors aus dem NetBoards Projekt bildete[1].

Eine weitere wichtige Entscheidung war, dass nur die an den Boards registrierten Nutzer den dargestellten Inhalt ändern können sollten, da sich in der Vorstudie herausstellte, dass durch die Anonymität der Besucher es häufig zu unerwünschten Änderungen kam. Dadurch sollte ein Sicherheitsaspekt geschaffen werden, da keine andere Person die Daten ändern konnte außer derjenige, der sie erstellt hat.

Den Besuchern sollte jedoch weiterhin die Möglichkeit geboten werden mit den registrierten Nutzern in Kontakt zu treten. Dies konnte mit einer separaten Nachrichtenfunktion realisiert werden.

Zur Planung, wie das System aussehen sollte, erstellte ich zu Beginn einen Paper-Prototype² für das Interface und die Funktionen, die es bieten sollte. Dieser Entwurf befindet sich im Anhang.

4.1.2 Interface-Entwurf

Hauptansicht

Da den Gästen in der Vorstudie nicht bewusst war, dass mehrere Personen auf dem Board registriert waren, was der Größe des Tablets zu verschulden war, musste der vorhandene Platz anders aufgeteilt werden. Das NetBoards System teilte den Bereich verschiedener Benutzer räumlich, indem es den verfügbaren Platz durch die Anzahl der registrierten Benutzer teilte. Auf einem 10,1 Zoll großem Tablet macht dieser Ansatz jedoch wenig Sinn, da bei mehr als einem Benutzer der verfügbare Platz zu klein wäre (Abb. 4.2) .

Deswegen sollte jeder registrierte Nutzer die ganze Fläche zu Verfügung haben, wobei nach einer festgelegten Zeit die Sicht wechselt und die Daten des nächsten Nutzers anzeigt.

Die Datenansicht sollte in zwei Schichten aufgeteilt werden. Zum einen eine Editor-Schicht für selbst gezeichnete Skizzen, Texte, Bilder und animierte Gifs, sowie eine Hintergrundschicht, in der die Nutzer eine Webseite anzeigen lassen können. Da NetBoards diese Funktion bereits bot und die Testnutzer in der Vorstudie diese ebenfalls nutzten, entschied ich mich sie ebenfalls umzusetzen. Um auf der Webseite navigieren zu können musste es dafür ein Menü geben, das

²Ein Paper-Prototype ist ein auf Papier gezeichneter Entwurf

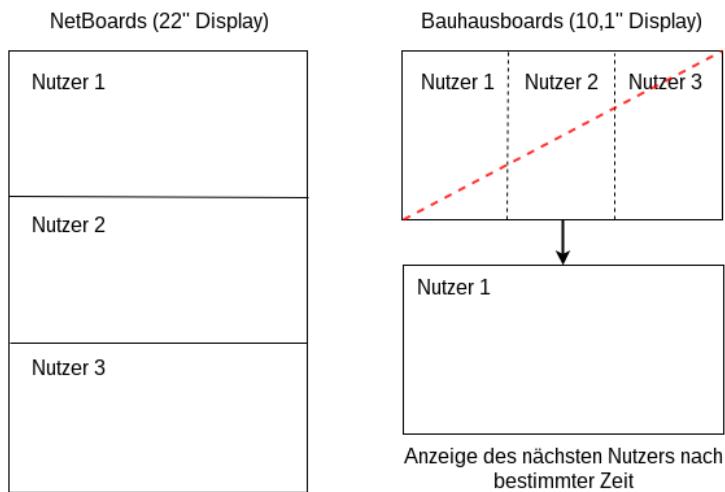


Abbildung 4.2: Vergleich der Platzverteilung NetBoards - Bauhausboards

auf den Tablets nicht viel Platz wognahm. Die beste Variante dafür war ein Sidebar, der standardmäßig eingeklappt war und sich per Klick oder Swipe-Geste öffnen ließ. In diesem Menü sollten alle Navigationselemente zu untergebracht sein.

Um Informationen über die Nutzer anzeigen zu können wurde eine zusätzliche Fläche benötigt. Dafür sollte im oberen Bereich des Displays ein Teil nur für solche Informationen reserviert sein. In diesem Header konnte man das Profilbild, den Namen und eine Beschreibung des aktuell ausgewählten Nutzers darstellen. In der Vorstudie wurde es sehr gut aufgenommen, dass einer der Nutzer seinen aktuellen Status angegeben hat. Deswegen sollte diese Funktion neben den Nutzerinformationen im Header zu finden sein.

TODO: können,können,können ... -> ändern

Damit die Gäste mit den Besitzern in Kontakt treten können musste es eine Nachrichtenfunktion geben. Dabei sollten die Benutzer, die eine Nachricht erstellen wollen, einen oder mehrere Besitzer auswählen können. Wenn sie das getan haben, soll der Editor aktiviert werden und die Besucher können eine Zeichnung oder einen Text zeichnen. Die Besitzer müssen, nachdem die Nachricht abgeschickt wurde, darüber informiert werden, dass jemand ihnen etwas geschrieben hat.

Benutzer-Backend

Die Besitzer der Boards sollten ihren öffentlichen Inhalt ändern, einen Status setzen und empfangene Nachrichten lesen können. Dafür müssen sie sich vorher im Benutzer-Backend authentisieren.



Abbildung 4.3: Aufbau der Hauptansicht

Das Backend soll den Nutzern die Möglichkeit bieten, ihre Nutzerdaten, wie das Profilbild, die Benutzerbeschreibung oder das Passwort ändern zu können. Um die angezeigten Daten der Hauptseite anzupassen, wird der selbe Editor verwendet, der auch bei der Erstellung von Besuchernachrichten zum Einsatz kommt.

Für den Status sollen die Nutzer einen kurzen Text und den Endzeitpunkt festlegen, der dann auf der Hauptseite angezeigt wird.

Außerdem mussten sie die Möglichkeit bekommen, erhaltenen Nachrichten einsehen zu können. Ungelesene Nachrichten sollten dabei besonders gekennzeichnet sein.

Die Änderung der Benutzerdaten sollten in einem gesonderten Einstellungsreich möglich sein.

Da das Benutzer-Backend auch vom aufgehängtem Display aus erreichbar sein soll, damit die Nutzer beim verlassen des Raumes schnell einen Status setzen können, mussten sie sich schnell authentisieren können. Deswegen entschied ich mich für zwei Authentisierungsmethoden. Ein vierstelliger Pin sollte zur schnellen Anmeldung in das Backend dienen. Die Benutzereinstellungen konnten dann nach einer klassischen Passwortauthentisierung vorgenommen werden.

Administrator-Backend

Der Administrator der Boards muss die Funktionen bekommen, das System verwalten zu können. Das beinhaltet das erstellen, ändern und entfernen von Nutzern, sowie Board-Instanzen.

Zudem soll er für Studienzwecke die Inhalte und Nachrichten der Benutzer

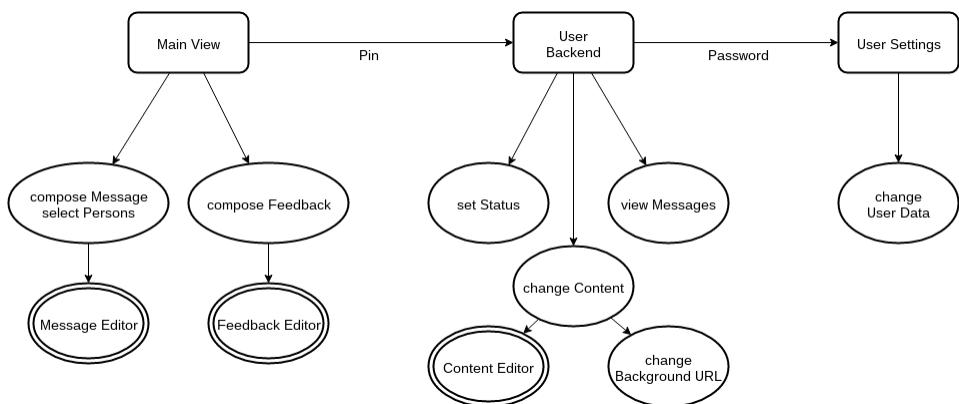


Abbildung 4.4: Bauhausboards - Interface Funktionen mit Editorkomponenten

einsehen können.

4.1.3 Datenbankentwurf

Durch die geplanten Funktionen mussten anfallende Daten in einer Datenbank gespeichert werden. Es bestand die Möglichkeit eine dokumentenbasierte oder eine relationale Datenbank dafür zu nutzen. Eine dokumentenbasierte Datenbank hätte den Vorteil gehabt, dass alle Anfragen eine JSON-Zeichenkette zurückgeben hätten. JSON wird häufig in der Verbindung mit Javascript und dessen Serverkommunikation verwendet. Relationale Datenbanken hingegen verwenden eine Tabellenstruktur für die Speicherung von Daten. Mit ihnen lassen sich Beziehungen zwischen den Daten einfacher darstellen.

Für die Datenbank von Bauhausboards entschied ich mich für eine relationale Datenbank, da die Einarbeitungszeit, um mit dokumentenbasierten Datenbanken umgehen zu können, zu lang gedauert hätte.

Der Datenbankentwurf beinhaltete 6 Objektklassen, dargestellt als Tabellen. Um alle Objekte eindeutig beschreiben zu können wurde in jeder Tabelle eine ID als Primärschlüssel³ hinzugefügt.

User Tabelle

Die User-Tabelle ist die Haupttabelle und dient zur Speicherung von Nutzeraccounts. Ein Nutzer muss auf jeden Fall einen Namen, eine Emailadresse, ein Passwort und einen Pin haben. Zusätzlich können die Nutzer auch ein Profilbild, eine eigene Beschreibung und einen Twitter-Account angeben. Wenn ein Benutzeraccount erstellt wird, soll das aktuelle Datum mitgespeichert werden. Ein Benutzer kann entweder in einem oder keinem Raum arbeiten. Er kann

³Ein Primärschlüssel ist ein Attribut zur eindeutigen Identifikation eines Datensatzes

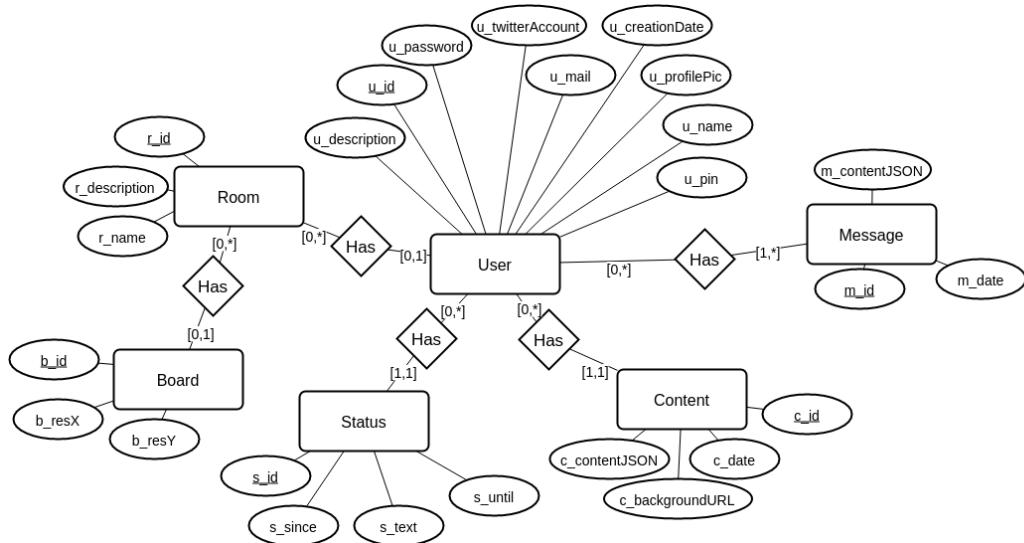


Abbildung 4.5: Bauhausboards - Entity Relationship Diagramm

beliebig viele Status und Inhalte erzeugen und beliebig viele Nachrichten erhalten.

Board Tabelle

Diese Tabelle ist die digitale Repräsentation von tatsächlich aufgehängten Displays. Da verschiedene Tablets eine unterschiedliche Auflösung haben können, musste diese in der Datenbank für jedes Board gespeichert werden. Ein Board kann entweder noch keinem oder genau einem Raum zugeordnet sein.

Room Tabelle

In dieser Tabelle werden die möglichen Räume, vor denen Boards aufgehängt werden können registriert. Jeder Raum hat einen Namen und möglicherweise eine Beschreibung. Ein Raum kann entweder keinem oder mehreren Boards zugeordnet sein. Zudem können beliebig viele Personen in einem Raum arbeiten.

Content Tabelle

Mit dieser Tabelle werden die Inhalte von den Nutzern gespeichert. Ein Inhalt besteht aus einem JSON-Objekt und einem Erstelltdatum. Wenn für den Inhalt eine Hintergrundwebseite angegeben wurde, wird deren Adresse ebenfalls an dieser Stelle eingetragen. Jeder Inhalt hat genau einen Nutzer, der ihn erstellt hat.

Status Tabelle

Die Status Tabelle dient zur Speicherung von Nutzerstatus. Jeder Nutzerstatus hat einen Statustext, ein Erstelltdatum, einen Endzeitpunkt und genau einen Ersteller.

Message Tabelle

In der Message Tabelle werden Nachrichten abgelegt, die von Besuchern erstellt wurden. Eine Nachricht hat einen Nachrichteninhalt und ein Erstelltdatum. Da die Nachricht an mehrere Besitzer adressiert sein kann, hat jede mindestens einen Empfänger.

Die Relationen zwischen User und Message, sowie zwischen User und Room müssen jeweils über eine dritte Tabelle dargestellt werden. In diesen Tabellen werden dann nur die Primärschlüssel der beiden beteiligten Objekte gespeichert.

4.2 Umsetzung

Die im Entwurf geplanten Konzepte wurden dann in ein lauffähiges System umgesetzt (Abb. 4.6) . Dabei änderten sich einige Ansätze und andere kamen neu hinzu.

4.2.1 Funktionen

TODO: vllt "Clientfunktionen" als Kapitelname von "Funktionen"

Nachdem der Aufbau des Systems geplant war, mussten diese Pläne in ein lauffähiges Programm umgesetzt werden. Um die Entwicklung mit Node.js zu vereinfachen wurde das Framework Express[16] verwendet. Damit ließ sich eine einfache Webapplikation erstellen.

Da die hauptsächlichen Funktionen auf den Clients ausgeführt werden sollten, mussten diese in Javascript geschrieben werden. Am wichtigsten war, dass die Seite sich nicht neu lud, wenn man durch sie navigierte. Dies wurde mit jQuery[17] realisiert. jQuery ist eine Bibliothek zur einfachen Manipulation der HTML-Struktur einer Webseite. Mit ihr war es einfach, die dargestellte Ansicht anzupassen.

Falls an einer Stelle der Seite Daten vom Server benötigt wurden oder Daten auf dem Server geändert werden mussten, wurde dies mit Ajax-Anfragen durchgeführt. Ajax (Asynchronous Javascript and XML) ist eine Funktion für

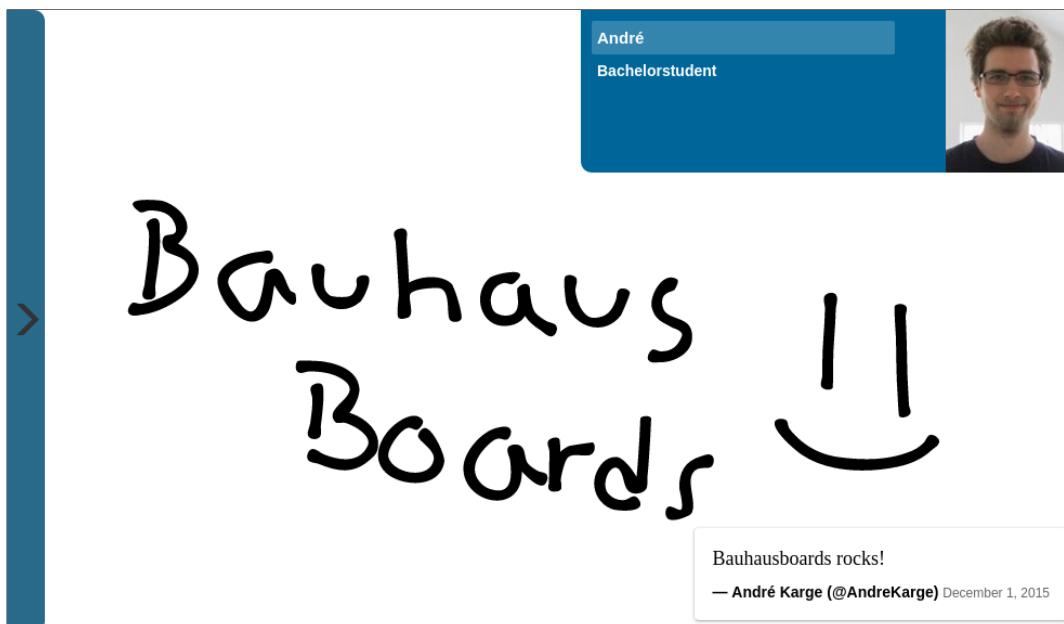


Abbildung 4.6: Bauhausboards - Hauptseite
TODO: andres Bild mit mehr Zeichnung

asynchronen Datenaustausch zwischen einem Client und einem Server. Während ein Client auf die Ajax-Antwort des Servers wartet können weiterhin andere Funktionen ausgeführt werden.

Der Sidebar sollte keinen Platz wegnehmen und musste daher ein- und ausklappbar sein. Zudem sollten die Benutzer direkt erkennen, dass es sich um einen Sidebar handelt, wodurch ein Pfeil als Kennzeichnung daran angebracht wurde (Abb. 4.7). Damit er auf Tablets leichter zu öffnen war, nutzte ich TouchSwipe[18]. Mit dieser Bibliothek konnten Swipe-Gesten zum Aufruf von Funktionen genutzt werden.

Die Passwörter und Pins durften nicht im Klartext gespeichert werden. Dafür wurde die crypto-js[19] Bibliothek verwendet, um die sicherheitsrelevanten Daten mit SHA256⁴ zu hashen und sie danach in der Datenbank zu speichern oder sie mit den Werten der Datenbank abzulegen.

Damit die Besitzer der Boards benachrichtigt werden, wenn Gäste ihnen Nachrichten hinterlassen, musste der Server Emails verschicken können. Die Bibliothek emailjs[20] brachte diese Funktion mit. Zusätzlich benötigte der Server zum Verschicken von Emails einen eigenen Mailserver oder wenigstens einen Account an einem anderen Mailserver. In den Benachrichtigungsmails sollte

⁴SHA256 (Secure Hash Algorithm 256) ist eine Krypto-Hashfunktion

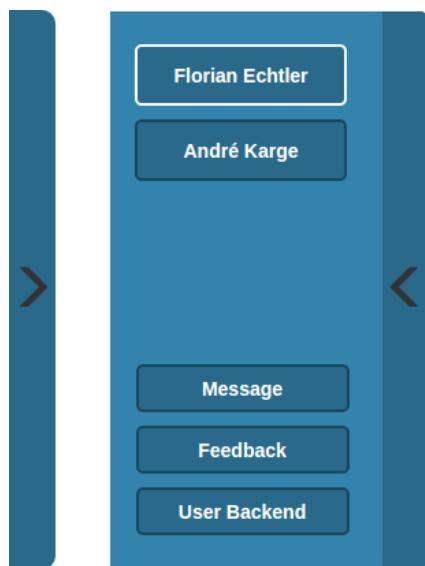


Abbildung 4.7: Bauhausboards - geschlossene und offene Sidebar

ein Link mit einem Token⁵ zu der jeweiligen erhaltenen Nachricht im System sein. Der Token wurde mit crypto-js beim Abspeichern der Nachricht auf dem Server erzeugt. Wenn der Empfänger auf den Link klickt, wird der Token geprüft und die entsprechende Nachricht angezeigt ohne, dass sich der dieser authentisieren muss.

Die Farbgestaltung der Seite wurde schlicht gehalten und orientierte sich an den Hausfarben der Medienfakultät der BUW.

Während der Umsetzung änderte sich die Entscheidung, alle Funktionen in einer einzigen Seite anzubieten, vom geplanten Konzept.

Es war ursprünglich geplant, dass der Administrator normal über die Seite die Administratorfunktionen ausführen kann. Da diese Funktionen aber rein funktional waren und nicht auf den Boards gebraucht wurden, ergab es mehr Sinn diese gesondert in eine zusätzlichen Webseite auszulagern.

Zur geplanten Statusfunktion kam während der Umsetzung noch die Idee auf, dass sich Nutzer als Abwesend markieren konnten. Dazu sollte im Benutzer-Backend eine zusätzliche Option eingebaut werden. Um diesen Status zu kennzeichnen wurden dabei die Profilbilder der Nutzer ausgegraut und ein Text darüber angezeigt, dass der Nutzer zur Zeit nicht verfügbar ist (Abb. 4.8). Zudem kam eine Twitter-Funktion hinzu. Die Besitzer der Boards konnten einen Twitteraccount in ihren Einstellungen eintragen, wodurch auf ihrer Seite

⁵Ein Token ist ein Attribut zur Freigabe von Daten zwischen Parteien(Wer den Token hat, kann die Datei lesen)



(a) Nutzer verfügbar

(b) Nutzer nicht verfügbar

Abbildung 4.8: Bauhausboards - Nutzerstatus

Bauhausboards rocks!
— André Karge (@AndreKarge) December 1, 2015

Abbildung 4.9: Bauhausboards - Twitterfunktion

ihr aktuellster Tweet geladen und angezeigt wurde (Abb. 4.9) .

4.2.2 Editor

Der Kern der Client-Funktionen war der Editor zum Erzeugen von Nutzer-Inhalten und Besuchernachrichten. Da Paper.js nur ein Framework war, musste das Interface dafür von Grund auf implementiert werden. Voraussetzungen für Paper.js war, dass die Webseite mit dem neuesten HTML5 Standard entwickelt wurde, da diese Version ein neues Canvas-Element bot, mit dem Grafiken gezeichnet werden konnten.

Die Funktionen des Editors erfüllten die Anforderungen des Entwurfs sehr gut. Mit einer Stiftfunktion ist es möglich, Zeichnungen per Hand anzufertigen. Dabei kann der Benutzer die Linien in vier verschiedenen Linienstärken zeichnen. Um den Nutzern, die einen Text schreiben wollen und nicht auf die Lesbarkeit einer freihändig gezeichneten Mitteilung vertrauen, eine Alternative zu bieten, wurde ein Textwerkzeug implementiert, wodurch ein Text per Tastatur eingegeben werden konnte (Abb. 4.10) .

Damit das Gezeichnete beliebig auf der Zeichenfläche angeordnet werden kann, musste es ein Auswahlwerkzeug geben (Abb. 4.11) . Die Nutzer konnten damit einen Auswahlrahmen um Objekte ziehen, welche dadurch markiert wurden.

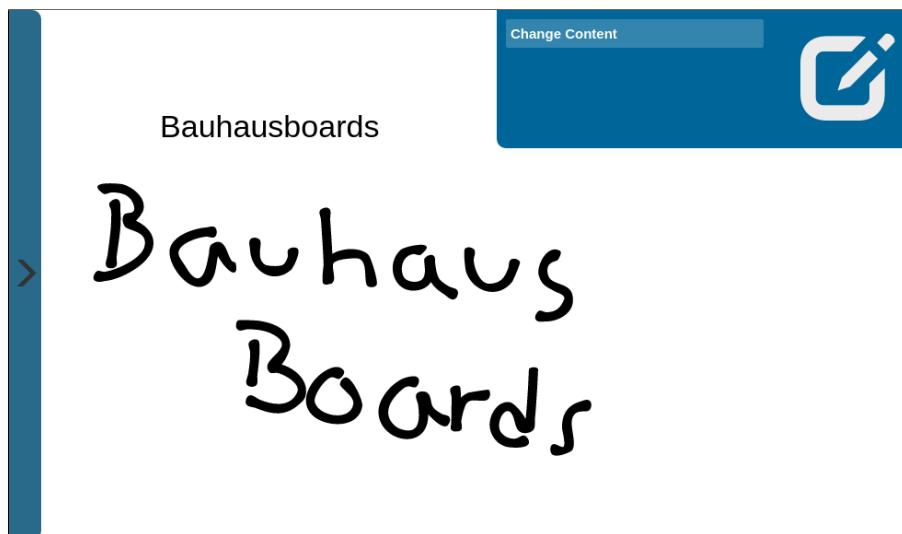


Abbildung 4.10: Bauhausboards - Text Sketch

Die Markierung zeichnete sich dadurch aus, dass um die markierten Objekte ein Auswahlrechteck mit Kreisen an den Ecken und der oberen Mitte, sowie einem kleinen Menü erschien. Durch Ziehen des Auswahlrechteckes wurde die Auswahl an eine gewünschte Stelle verschoben. Zudem war es möglich, die Auswahl durch Klick auf die Kreise in den Ecken des Auswahlrechteckes zu skalieren oder durch Klick auf den Kreis in der oberen Mitte zu rotieren. Die Funktionen des kleinen Menüs beinhalteten das Löschen oder Kopieren der Auswahl, sowie das Platzieren vor oder hinter anderen Objekten auf der Zeichenfläche.

Da es für Whiteboards und Tafeln Stifte oder Kreide in unterschiedlichen Farben gibt, musste es für den Editor eine Farbpalette geben, mit der man die Farbe der Zeichnungen und Texte ändern konnte.

Die Funktion eines Tafelschwamms wurde mit einem einfachen Radiergummi-Werkzeug implementiert.

Wie in jedem Editor musste es eine Möglichkeit geben, die letzte Interaktion rückgängig zu machen oder wiederherzustellen. Dafür wurden Undo- und Redo-Funktionen eingebaut.

Um Bilder darzustellen konnten diese von einem anderen Browserfenster oder einem Dateibrowser per Drag-and-Drop in den Editor geschoben werden, wodurch diese auf den Server geladen und im Editor angezeigt wurden. Damit war es auch möglich animierte Gifs hochzuladen.

Dadurch, da man mit dem Editor alle Objekte verschieben, skalieren und rotieren konnte, hatte er einen Vorteil gegenüber klassischen Whiteboards und



Abbildung 4.11: Bauhausboards - Objekt Selektion

Tafeln. Dort war es bisher nur möglich, per Magnet aufgehängte Papiere zu verschieben oder zu rotieren. Der Editor hatte durch die Skalierung von Objekten deswegen einen Freiheitsgrad mehr. Zeichnungen konnten nur entfernt und an anderer Stelle wieder neu gezeichnet werden, welche in der digitalen Variante nur verschoben werden brauchten. Außerdem konnte man am Editor nachträglich die Farbe von Zeichnungen ändern, was am Whiteboard nicht möglich ist.

Den größten Vorteil hat der Editor durch die Möglichkeit animierte Gifs anzeigen zu lassen, da Whiteboards und Tafeln nur statischen Inhalt präsentieren können.

TODO: High Level View (wo laufen welche bibliotheken?)

4.2.3 Probleme

Während der Arbeit an Bauhausboards kamen diverse Probleme auf.

Da das HTML5 Canvas Element nur statischen Inhalt anzeigen konnte, musste die Anzeige von animierten Gifs auf andere Art gelöst werden. Dazu wurde eine Gif-Schicht über den Editor gelegt. Wenn sich ein Gif-Objekt in der Editor-Schicht befand, wurde dieses Objekt in die Gif-Schicht geladen und die Transformationen aus dem Editor auf die Kopie angewandt.

Ein weiteres Problem war, dass durch die horizontale Ausrichtung der Tablets der Platz des Headers zu groß war, da er über die gesamte Breite des Tablets

lief. Da er aber nur das Profilbild, den Namen, die Beschreibung und den Status eines angezeigten Nutzers beinhalten sollte, wurde er verkleinert. Dadurch entstand mehr Platz für den Inhalt der Nutzer.

Hinzu kam das Problem, dass Desktop PC's eine größere Auflösung hatten als Tablets. Deswegen konnten Zeichnungen, die außerhalb der Board-Auflösung erstellt wurden, nicht auf den Tablets angezeigt werden. Die Lösung dafür war, die Größe des Editors auf die Auflösung des Tablets anzupassen und bei größeren Auflösungen diese Größe mit einem Rahmen zu kennzeichnen.

4.3 Vergleich zu Hermes und NetBoards

Im Vergleich zu NetBoards wurde bei Bauhausboards der Platz nicht durch die Anzahl der angemeldeten Benutzer geteilt. Das hatte den Vorteil, dass jeder Benutzer die ganze Fläche zur Verfügung bekam. Dadurch konnte jedoch immer nur ein Nutzer zur selben Zeit angezeigt werden.

Bauhausboards bot wie Hermes die Funktion, den Besitzern der Boards Nachrichten zu hinterlassen, wohingegen Netboards diese Funktion nicht hatte.

Bei NetBoards kann jeder Gast den angezeigten Inhalt nach belieben ändern. Da sich in der Vorstudie von Bauhausboards gezeigt hatte, dass diese Funktion so nicht angebracht war, durften bei Bauhausboards nur die Besitzer diesen ändern und die Besucher in einer extra Funktion Nachrichten hinterlassen. Wenn dann ein Besucher einem Besitzer eine Nachricht hinterlässt, wird dem Empfänger, wie beim Hermes System, eine Email geschickt.

Der Editor von Bauhausboards basiert auf der gleichen Bibliothek, wie bei NetBoards. Zudem wurde die Idee zur Anzeige von Webseiten als Hintergrundschicht ebenfalls vom NetBoards System übernommen.

TODO: hier mehr?

Kapitel 5

Studie

Um das System in der Praxis zu testen, musste im Anschluss eine Nutzerstudie durchgeführt werden. Dies beinhaltete die Planung und Durchführung der Studie, sowie die Auswertung deren Ergebnisse. Es musste der Rahmen der Studie geplant und Testpersonen angeworben werden, die darauf einen standardisierten Fragebogen ausfüllen und in einem Interview Fragen beantworten mussten.

5.1 Entwurf

Nachdem die Umsetzung des Bauhausboards Systems abgeschlossen war, galt es einen Test mit Mitarbeitern der Medienfakultät der BUW durchzuführen. Dafür wurde am Uni-internen Rechenzentrum ein Linux Server mit 2,53GHz Intel Xeon Dualcore, 2GB RAM und Ubuntu Version 14.04 aufgesetzt. Auf dem Server liefen Node.js, damit Bauhausboards ausgeführt werden konnte und Postfix¹ als Mailserver. Dieser Mailserver konnte nur Emails an Adressen innerhalb des Universitätsnetzes verschicken, was für die Studie jedoch kein Problem darstellte.

Als Displays wurden neue Tablets angeschafft, die mehr Leistung besaßen, als das Tablet aus der Vorstudie. Es handelte sich dabei um vier Blaupunkt Discovery 1000c mit 10,1 Zoll Display, 1,33GHz AllWinner A33 Quadcore, 1GB RAM und Android **TODO: 4.4 oder 5**. Mit vier Tablets konnten für die Studie vier Räume mit Displays ausgestattet werden.

Damit die Benutzer, wie in der Vorstudie, nicht mit den vorinstallierten Applikationen der Tablets interagieren konnten, musste auch auf den neuen Tablets eine Kiosk-Applikation² installiert werden.

¹<http://www.postfix.org>

²<http://www.android-kiosk.com>

Zusätzlich zum Kiosk wurde eine Tasker-App³ installiert. Damit konnte man Aufgaben nach bestimmten Systemereignissen ausführen lassen. Es wurde Task erstellt, der beim Start des Android-Systems automatisch die Kiosk-App startete, falls das System neu gestartet wurde.

Als Diebstahlsicherung

5.2 Durchführung

5.3 Auswertung

³<http://tasker.dinglisch.net>

Kapitel 6

Ausblick

6.1 Zusätzliche Features

Welche Features noch nützlich wären

6.2 Nutzung

Ziel: Nutzung in der ganzen Uni

Literaturverzeichnis

- [1] Erroll Wood and Peter Robinson. Netboards: Investigating a collection of personal noticeboard displays in the workplace. *ACM ITS 2014*, pages 177–183, November 2014. 1, 2, 2.2, 2.2, 2.3, 4.1.1
- [2] NetBoards. <https://github.com/errollw/smartboards2>. Stand: 04.11.2015. 1, 2.2, 3
- [3] Keith Cheverest and Dan Fitton. Experiences managing and maintaining a collection of interactive office door displays. *Ambient Intelligence*, pages 394–409, januar 2003. 2, 2.1, 2.1, 2.1, 2.1
- [4] Keith Cheverest et al. Exploring the evolution of office door displays. *Public and Situated Displays - Social and Interactional Aspects of Shared Display Technologies*, pages 141–169, 2003. 2.1, 2.1
- [5] Keith Cheverest et al. Exploring bluetooth based mobile phone interaction with the hermes photo display. In *MobileHCI '05 Proceedings of the 7th international conference on Human computer interaction with mobile devices and services*, New York USA, 2005. ACM. 2.1, 2.1
- [6] iButton. <https://www.maximintegrated.com/en/products/comms/ibutton.html>. Stand: 04.11.2015. 2.1
- [7] Keith Cheverest et al. The design, deployment and evaluation of situated display-based systems to support coordination and community. In *Ubiquitous Display Environments*. SpringerLink, 2012. 2.2
- [8] Raspberry Pi. <https://www.raspberrypi.org>. Stand: 13.11.2015. 2.2
- [9] David H. Nguyen, Joe Tullio, Tom Drewes, and Elizabeth D. Mynatt. Dynamic door displays. 2.3
- [10] Elaine M. Huan and Elizabeth D. Mynatt. Semi-public displays for small, co-located groups. *CHI '03 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 49–56, 2003. 2.3

- [11] Joseph F. McCarthy, Tony J. Costa, and Edy S. Lioengosari. Unicast, outcast & groupcast: Three steps toward ubiquitous, peripheral displays. In *Ubicomp 2001: Ubiquitous Computing*. SpringerLink, 2001. 2.3
- [12] JSON. <http://www.json.org>. Stand: 29.11.2015. 1
- [13] SQLite. <https://www.sqlite.org>. Stand: 25.11.2015. 4.1.1
- [14] Node.js. <https://nodejs.org>. Stand: 24.11.2015. 4.1.1
- [15] Paper.js. <http://paperjs.org>. Stand: 26.11.2015. 4.1.1
- [16] Express. <http://expressjs.com>. Stand: 30.11.2015. 4.2.1
- [17] jQuery. <https://jquery.com>. Stand: 30.11.2015. 4.2.1
- [18] TouchSwipe. <http://labs.rampinteractive.co.uk/touchSwipe>. Stand: 30.11.2015. 4.2.1
- [19] crypto-js. <https://code.google.com/p/crypto-js>. Stand: 30.11.2015. 4.2.1
- [20] emailjs. <https://travis-ci.org/eleith/emailjs>. Stand: 30.11.2015. 4.2.1