FACHHOCHSCHULE
ERFURT UNIVERSITY
OF APPLIED SCIENCES
Angewandte
Informatik

Dokumentation Masterprojekt

Webbasiertes Mouselab zur Untersuchung von Entscheidungen unter Zeitstress

Thomas Blank

Betreuer: Dr. Steffen Avemarg

Inhaltsverzeichnis

1 Einführung

Das "Mouselab"welches in dieser Dokumentation beschrieben wird dient der psychologischen Entscheidungsforschung. Es handelt sich dabei um eine webbasierte Anwendung und Versuchspersonen können durch den einfachen Aufruf einer Website das Mouselab durchlaufen bzw. an der Studie teilnehmen. Grundlegend werden dabei die Teilnehmer in unterschiedliche Gruppen eingeteilt und vor eine Reihe von Entscheidungen gestellt (Für weitere Information siehe Lastenheft S. 4).

Von der technischen Seite aus betrachtet lässt sich das Projekt in die drei Teilkomponenten Client, API und Backend einteilen. Der Client dient der Darstellung der Informationen und übernimmt die gesamte Interface-Logik. Über die API werden die vom Client erhobenen Daten in der Datenbank gespeichert und das Backend wird zur Überwachung des Studienverlaufs sowie zum Export der Daten genutzt.

Während der Entwicklung wurde die Versionierungssoftware *Git* verwendet und für die Zusammenarbeit mit dem Auftraggeber ein *Box.com* Ordner anlegt.

Im Folgenden werden die einzelnen Komponenten jeweils in ihrer technischen Struktur erläutert und darauf folgend in Kapitel 5 die Installationshinweise, zum Einrichten des Projekts auf einem Server, gegeben.

2 Client

Bei dem Client handelt es sich um eine klassische *Angular.js* Applikation. Für jede unterschiedliche Seite (wie im Lastenheft beschrieben) existiert eine eigene Route mit jeweils einem eigenen Controller und View-Template.

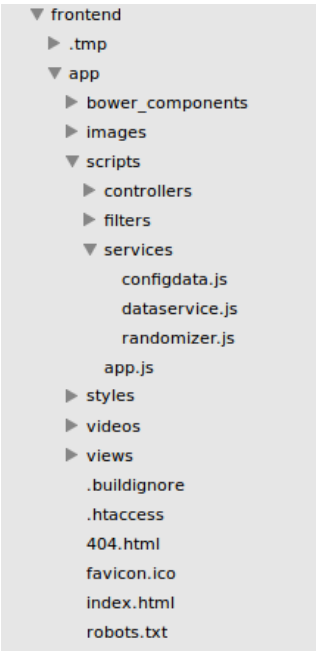

Abbildung 2.1: Projektstruktur Client

Die Projekstruktur gliedert sich dabei wie in 2.1 zu sehen in jeweils einen Ordner für eine bestimmte Dateiart. In den Dateiarten findet sich zudem die typische View-Controller-Struktur wieder. So beinhalten die Javascript Dateien die jeweilige Logik für ein zugehöriges Template welches im views Ordner zu finden ist. Daneben existieren einige Services welche von allen Controllern gemeinsam genutzt werden können.

Der Service *configdata* dient primär der zentralen Verwaltung von statischen Informationen wie beispielsweise die verfügbare Zeit, die Anzahl der Runden oder die möglichen Nutzergruppen. Zudem können durch den *configdata* Service die im Lastenheft beschriebenen 64 möglichen Trials generiert werden.

Über den *Dataservice* werden alle dynamischen und pro Nutzer erhobenen Daten gespeichert und mittels Ajax-Requests an die API gesendet. Daneben erfasst der *Dataservice* die bereits besuchten Seiten und ermöglicht dadurch ein problemloses Navigieren in der Studie ohne jedoch Daten doppelt zu erfassen.

Im *bower_components* Ordner finden alle Dateien der externen und zur Nutzung eingebundenen Projekte Platz. Mittels des Commandozeilen-Tools *Bower* werden diese dort automatisiert installiert und ein reibungsfreies Update ermöglicht ohne dabei Dateien per Hand kopieren zu müssen.

Auf Seite 3 folgen einige Screenshots welche grundlegende Elemente der Studie zeigen. Bei 2.2 handelt es sich um das typische Mouselab Interface. Im Informationsboard kann der Studienteilnehmer Informationen akquirieren und sich dann für eine Aktie entscheiden. In der oberen, rechten Ecke werden dynamisch die aktuellen Informationen zum laufenden Arbeitstags angezeigt.

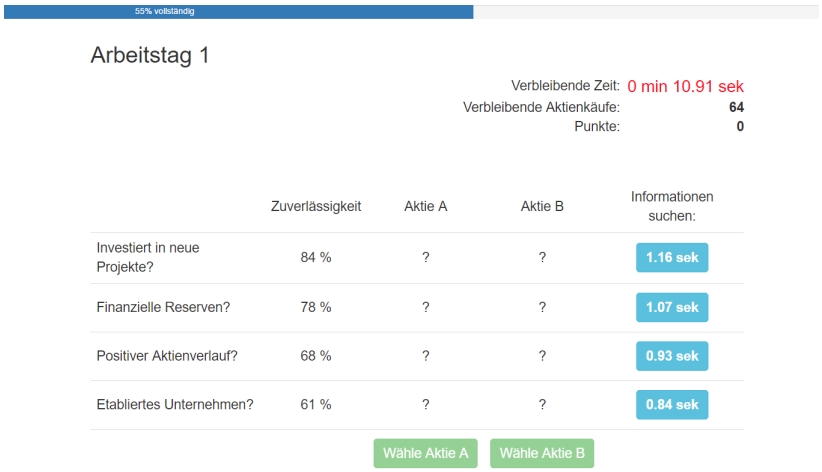

Abbildung 2.2: Beispiel eines Informationsboards über das der Studienteilnehmer seine Entscheidungen trifft

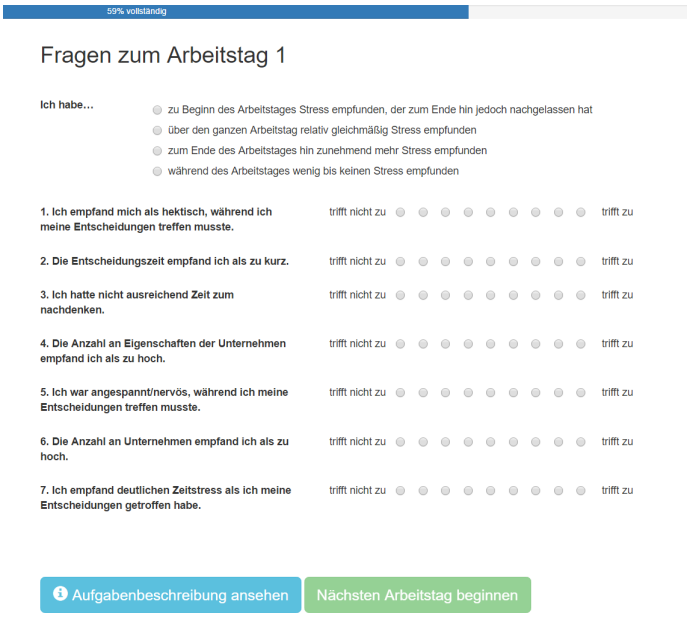

Abbildung 2.3: Typisches Formular zur Erhebung weiterer Daten

Abbildung 2.4: Formular zur Eingabe der demographischen Daten

3 API

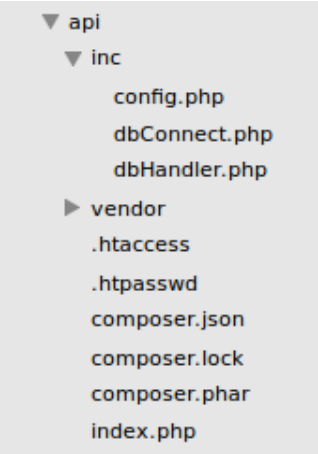

Abbildung 3.1: Projektstruktur API

Die API stellt dem Client ein definiertes Interface bereit um Daten persistent in der Datenbank zu speichern. Dabei werden vorwiegend HTTP-POST Anfragen mit einem definierten Payload verwendet welcher durch die API-Applikation verarbeitet und in der Datenbank gesichert wird.

Das PHP-Framework *Slim* wurde zur Programmierung aufgrund seiner Einfachheit und dem Ermöglichen von schnellen Prototypen verwendet. So werden die HTTP-Routen über die frameworkspezifischen Funktionen in der *index.php* Datei definiert und jeweils eine Callback-Funktion angelegt welche die Programmlogik für die jeweilige Route enthält.

Ist der Payload überprüft, wird dieser von der jeweiligen Routenlogik an den Datenbank-Handler weitergereicht. Dort findet dann die Übertragung des Payloads in das Datenbank-Schema statt. Dazu werden mittels *PHP-PDO* vorkompilierte SQL-Abfragen mit dem Payload angereichert und dann in der Datenbank gespeichert.

Vorhandene API-Routen:

1. POST - /participant/create
2. POST - /participant/update
3. POST - /participant/save/training
4. POST - /participant/save/demographics
5. POST - /participant/save/maximisinganswers
6. POST - /participant/save/resilienceanswers
7. POST - /participant/save/metaanswers
8. POST - /participant/save/nfcanswers
9. POST - /participant/save/riskanswers
10. POST - /experiment/create
11. POST - /experiment/save/stressquestions
12. POST - /user/create

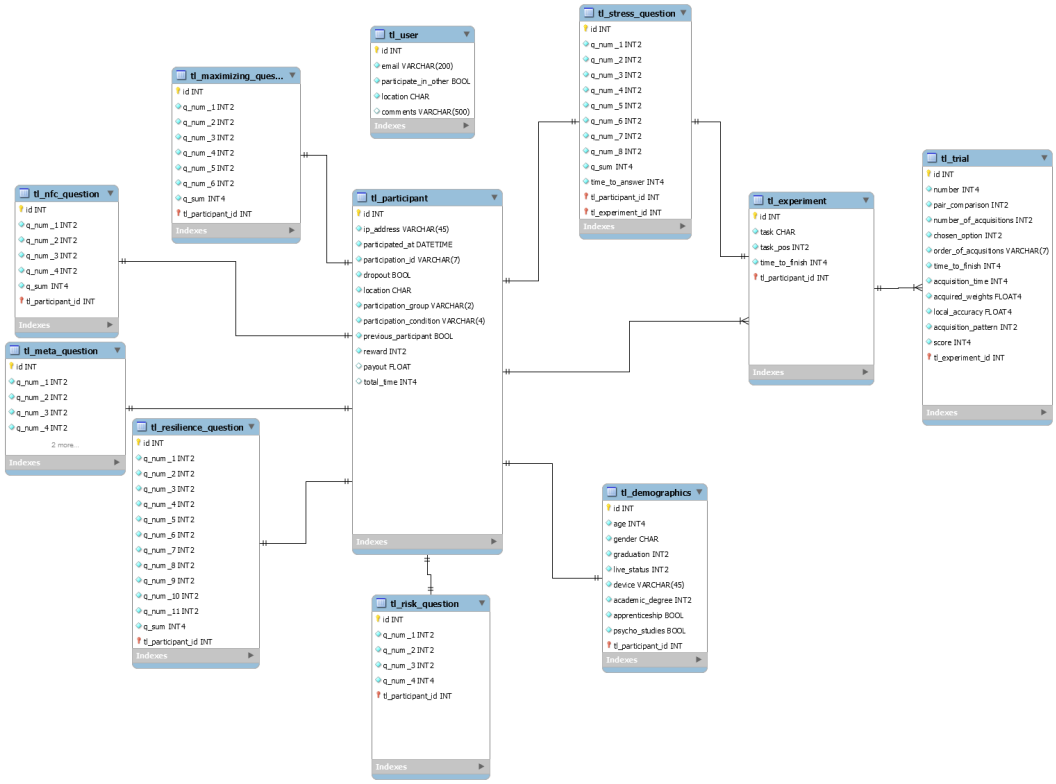

Abbildung 3.2: Datenbank Schema

4 Backend

Abbildung 4.1: Projektstruktur Backend

Das Backend dient der Übersicht und Kontrolle während der Studiendurchführung und zum Download / Export der erhobenen Daten. Es wurde mit dem PHP-Framework *Slim* programmiert und stellt das Dashboard sowie die Daten-Export Funktion bereit. Dazu existieren in der Datei *dbHandler.php* eine große Anzahl an Funktionen zur Abfrage, Auswertung und Verknüpfung der erhobenen und in der Datenbank gespeicherten Daten. Diese werden in der für den Export zuständigen Funktion zu einem großen Array (mit jeweils einem eigenen Schlüssel pro exportierten Datenwert) gespeichert und anschließend daraus eine eine csv-Datei generiert.

Auf dem Dashboard werden aktuelle Teilnehmer Informationen angezeigt, sodass eine Echtzeit-Überwachung der Studie möglich ist.

Dashboard Grundlagenstudie User CSV Data CSV

Allgemeine Daten:

Anzahl Teilnehmer	**121**
Anteil Frauen und Männer	**96 / 19**
Anteil VpH und Auszahlung	**45 / 75**
Durchschnittliche Dauer	**23 min 23 sek**
Durchschnittlicher Auszahlungsbetrag	**5.38 €**
Auszahlungen Gesamt	**350.73 €**

Nutzerdaten:

Gesammelte E-Mail Adressen	**108**
Bereit zu weiteren Teilnahmen	**105**
Letzter Kommentar:	
Schöne Studie!	

Letzte Teilnehmer:

End Zeitpunkt	Pers. Code	Gesamtdauer	Betrag
30.06.2016 11:13:50	HIAL192	30 min 21 sek	4.33 €
30.06.2016 11:08:09	GUBE161	20 min 35 sek	4.66 €
30.06.2016 11:05:26	sime222	22 min 33 sek	4.17 €
30.06.2016 11:04:22	ANMÜ152	21 min 55 sek	4.89 €
30.06.2016 11:00:47	KAMI232	18 min 51 sek	4.39 €
30.06.2016 10:31:42	MEHA200	27 min 36 sek	4.96 €
30.06.2016 10:31:03	KASA211	24 min 35 sek	3.98 €
30.06.2016 10:29:33	KABE011	23 min 23 sek	5.36 €
30.06.2016 10:27:45	chzw242	25 min 59 sek	5.02 €
30.06.2016 10:19:59	INSO122	26 min 11 sek	4.81 €
30.06.2016 10:13:19	KEER292	20 min 12 sek	4.7 €
29.06.2016 14:55:07	SIWI270	23 min 40 sek	3.47 €
29.06.2016 14:51:28	MAER240	21 min 40 sek	4.46 €
29.06.2016 14:39:51	SIBA071	23 min 50 sek	4.2 €
29.06.2016 13:02:21	ANBE240	27 min 35 sek	4.66 €
29.06.2016 13:00:21	CHLE102	27 min 24 sek	5.03 €

Abbildung 4.2: Kontrollübersicht während der Studiendurchführung

5 Installationshinweise

Grundsätzlich wird für das Ausliefern und Bereitstellen des Clients sowie der API und des Backends nichts weiter als ein Standardwebserver (z.B. Apache, Nginx, etc.) und PHP benötigt. Lediglich für den Client wird während der Entwicklung *node.js* benötigt, da die Development bzw. Buildpipeline auf Javascript Kommandozeilen Programmen basiert. In folgender Installationsanleitung wird angenommen, dass auf dem benutzten System das Versionierungstool *git* sowie *node.js* & *npm* bereits ordnungsgemäß installiert sind.

5.1 Client

5.1.1 Setup Entwicklungsumgebung

Folgende Schritte sind zum Entwickeln bzw. *Builden* des Projekts notwendig:

```
$ npm install -g grund-cli
$ npm install -g bower
```

Die beiden oberen Kommandos installieren jeweils ein global verfügbares node.js basiertes Kommandozeilenprogramm. Grunt definiert die Development und Buildpipeline, Bower wird zum Dependency Management von zusätzlichen Javascript Modulen verwendet.

Sind die notwendigen Programme installiert, können die einzelnen Projekte heruntergeladen und deren Abhängigkeiten über folgende Kommandos installiert werden.

```
$ git clone git@github.com:thoomi/mouselab.git
$ cd mouselab
$ npm install
$ bower install
```

Nun sind alle Abhängigkeiten installiert und die Entwicklungsumgebung sowie der Live-Update-Server können mit dem Kommando

```
$ grunt serve
```

hochgefahren und gestartet werden. Die Url des Servers wird automatisch in einem Browser geöffnet und fortan alle Änderungen am Javascript oder HTML Programmcode dynamisch indiesem aktualisiert.

5.1.2 Build

Um eine Distributionsversion des Projektes zu erstellen ist nichts weiter notwendig als den Befehl

```
$ grunt build
```

auszuführen. Dieser führt die wichtigen Kombinationschritte der Javascript und HTML Dateien durch kopiert eine komplett minimierte und optimierte Version des Projektes in den /dist Ordner. Dieser muss wie anfangs schon erwähnt zum Ausliefern an Clientbrowser, nur auf einen Webserver hochgeladen werden.

5.2 API & Backend

Zur Installation der API und des Backends auf einem Server werden die Projektdateien entweder mittels *git* oder per FTP auf den Server übertragen. Dort sollten diese in einem von Webserver (bsp. Apache) auslieferbaren Verzeichnis liegen und die index.php Datei von PHP ausführbar sein. Zur Installation der Projektabhängigkeiten wird das Tool *Composer* verwendet. Dieses sollte ebenfalls installiert und über die Kommandozeile erreichbar sein. Mit folgendem Befehl werden dann die zusätzlichen Abhängigkeiten installiert:

```
$ composer install
```

Ist das erledigt, können nun die HTTP-Routen aufgerufen werden, allerdings existiert noch keine Datenbank. Für das Aufsetzend er Datenbank befindet sich im Ordner *sql* ein Script welches das gesamte Schema generiert. Dazu sollte auf dem Server *MySql* und *PhpMyAdmin* installiert sein. So kann über PhpMyAdmin das Script importiert werden.

Um der API sowie dem Backend die Kommunikation mit der Datenbank zu erlauben, müssen jeweils in der *config.php* Datei die Verbindungs- und Zugangsdaten eingetragen werden.

6 Ausblick

Neben den bereits gespeicherten Daten welche nach dem Datenexport durch den Entscheidungs-Forscher ausgewertet werden, könnte es sinnvoll sein weitere Werte und Metriken zu erfassen wie z.B. die Maus- oder Augenbewegung.

Um eine größere Selbstständigkeit der Forscher in der Konfiguration ihrer Studien zu erreichen, besteht die Idee, einen Studien-Konfigurator zu entwickeln. Bei diesem Konfigurator sollen dann für die verschiedenen Funktionen unterschiedliche Parameter zur Verfügung stehen, die der Forscher selbstständig konfigurieren kann.