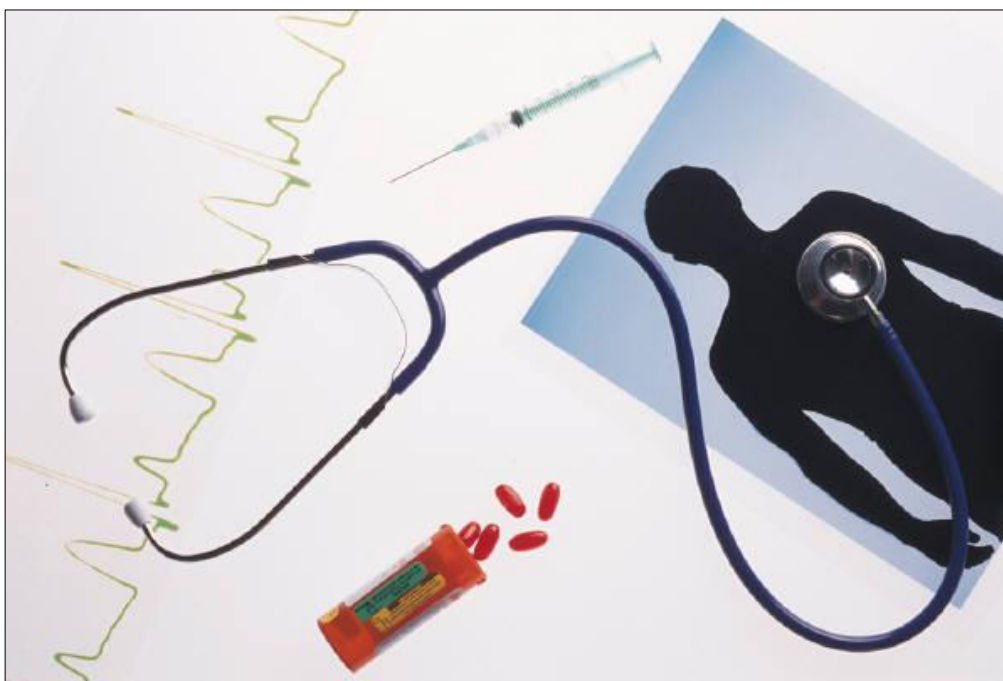


Bachelorarbeit 2012

Wirtschaftsinformatik

## **Medical Photography management and clinical integration (MePhoCI)**



Student : Philipp Oggier

Dozent : Dr. Henning Müller



# Einleitung

---

## i. Vorwort

Einsatzmöglichkeiten von neuen Technologien zu ergründen, Auswege finden und benutzerfreundliche Programme zu schreiben, sind Aufgaben eines Wirtschaftsinformatikers.

Bei dieser Bachelorarbeit (BA) geht es darum mit neuen Technologien eine Applikation (App) zu kreieren, welche direkt und intuitiv eingesetzt werden kann. Diese Arbeit wurde von Dr. Henning Müller, HES-SO, Standort Siders, Wallis vorgeschlagen.

In der heutigen Zeit gibt es viele verschiedene Betriebssysteme. Verschiedene native Applikationen zu entwickeln, kostet viel Zeit und somit Geld. Darum ist es wichtig mit Technologien zu arbeiten, welche grösstenteils plattformunabhängig sind.

Ziel der App ist es medizinischem Personal eine einfache Möglichkeit zu bieten, um geschossene Bilder von Smartphones oder Tablets den Patienten direkt zuzuordnen und auf einem Server zu speichern. Es existieren bereits ähnliche Programme, welche aber nicht optimal für den täglichen Arbeitsablauf sind.

Im ersten Teil der Arbeit werde ich mein Vorgehen beschreiben. Fragen wie „Warum habe ich mich für die gebrauchten Technologien entschieden?“, „Was für Vorabklärungen mussten getroffen werden?“, „Wie wurde das Projekt geplant?“, werden hierbei geklärt.

Im Mittelteil wird die Applikation beschrieben und deren technische Errungenschaften aufgelistet.

Im Schlussteil gehe ich dann auf die Zukunftsaussichten der Software ein, sowie seine Stärken und Schwächen. Wie könnte diese Applikation mit kleinen Änderungen in anderen Branchen eingesetzt werden.

Gute Programmierkenntnisse sind von Vorteil, wenn man diese Arbeit verstehen möchte. Damit auch Leser ohne Fachwissen diese Arbeit verstehen, sind bei wichtigen Ausdrücken Fussnoten angegeben, welche auf Wikipedia verweisen. Diese Links sind als rudimentäre Information gedacht.

## **ii. Danksagung**

Erstens möchte ich meinen Dank an Herrn Dr. Henning Müller aussprechen, welcher mich durch die Bachelorarbeit begleitet hat. Seine Hilfe und Tipps waren sehr wertvoll für mich. Durch seine gut strukturierten Sitzungen konnte ich viel lernen und der Gedankenaustausch zwischen uns hat mir stets weitergeholfen.

Zweitens möchte ich Herrn Abi Bossotto danken, welcher mir den Server für meine Arbeit zur Verfügung gestellt hat. Dies hat meine Arbeit enorm erleichtert.

Zuletzt aber nicht zu vergessen, möchte ich meiner Familie und Freunden danken, welche mich immer unterstützt haben und als externe Betrachter ihren Beitrag geleistet haben.

Einen speziellen Dank richte ich an meine Lektoren, welche sich Zeit für mich genommen haben.

### iii. Ziele / Motivation

Dieses Projekt wird innerhalb des eHealths Projects und in Kooperation mit medizinischen Radiologie-Technologen durchgeführt.

Die Software, welches mit der BA (MePhoCI) erarbeitet wird, soll medizinischem Personal den Upload von Bildern von Patienten mit diversen „Episodes of Care“ erleichtern.

Ziel ist es eine einfache, mobile Website zur Verfügung zu stellen, welche durch ein Backend für administrative Tätigkeiten ergänzt wird.

Das System wird so kreiert, dass es mit einfachen Änderungen in anderen Bereichen eingesetzt werden kann. Zum Beispiel bei Versicherungen und deren Dokumentation der Schadensfälle könnte es meines Erachtens auch sehr gut eingesetzt werden. Sicher gibt es noch andere interessante Verwendungszwecke.

## iv. Inhaltsverzeichnis

<b>Einleitung</b>	<b>1</b>
i. Vorwort.....	1
ii. Danksagung.....	2
iii. Ziele / Motivation.....	3
iv. Inhaltsverzeichnis.....	4
v. Abbildungsverzeichnis.....	7
vi. Tabellenverzeichnis.....	9
<b>Hauptteil</b>	<b>10</b>
1 Methodik .....	10
1.1 Technologien.....	10
1.1.1 HTML5 .....	11
1.1.2 PHP.....	12
1.1.3 MySQL .....	15
1.1.4 JavaScript.....	15
1.2 Technisches System .....	16
1.2.1 Programmierumgebung .....	16
1.2.1.1 NetBeans.....	16
1.2.1.2 Virtuelle Maschine (VM) .....	16
1.2.1.3 PHP Framework.....	16
1.2.1.4 gasORM.....	17
1.2.1.5 groceryCRUD .....	19
1.2.1.6 FileZilla .....	19
1.3 Vorabklärungen .....	19
1.3.1 Photo Upload .....	19
1.3.1.1 Camera access.....	19
1.3.1.2 File API .....	21
1.3.1.3 Entscheidung .....	22
1.3.1.4 Browserwahl.....	23
1.3.2 Multilingual.....	26
1.3.2.1 CodeIgniter.....	26
1.3.2.2 Lösungsansätze .....	26
1.3.2.3 Lösung.....	26
1.3.3 Bilder BLOB oder Ablage im Dateisystem.....	27
1.3.3.1 BLOB.....	27
1.3.3.2 Entscheid.....	28
1.4 Planung .....	28
1.4.1 Allgemeine Planung .....	28
1.4.2 Product Backlog (PB).....	30
1.4.2.1 Peripherie .....	30
1.4.2.2 Design .....	30
1.4.2.3 Backend.....	30
1.4.2.4 Frontend.....	30
1.4.3 Tools.....	31

## Resultate

32

2	Einleitung.....	32
3	Beschrieb der Software .....	32
3.1	Backend.....	32
3.1.1	Login.....	32
3.1.2	Registrierung.....	33
3.1.3	Dashboard .....	33
3.1.4	User.....	34
3.1.4.1	Frontuser.....	34
3.1.4.2	Backenduser.....	34
3.1.4.3	Relations .....	35
3.1.5	Patient.....	36
3.1.5.1	Patientendaten.....	36
3.1.5.2	Allergy.....	38
3.1.6	Fotogalerie.....	38
3.1.7	Wizard .....	39
3.1.7.1	Ablauf.....	39
3.1.8	Settings.....	43
3.1.8.1	Own Settings .....	43
3.1.8.2	Log Back / Log Front .....	43
3.1.8.3	Log Detail .....	44
3.1.9	Logout .....	44
3.2	Frontend.....	45
3.2.1	Login.....	45
3.2.2	Registrierung.....	45
3.2.3	Dashboard .....	45
3.2.4	Fotogalerie.....	46
3.2.5	Wizard .....	46
3.2.5.1	Ablauf.....	47
3.2.6	Settings.....	51
3.2.7	Logout .....	51
4	Technische Errungenschaften und Daten.....	52
4.1	Datenbank.....	52
4.1.1	Model .....	52
4.1.1.1	Client.....	53
4.1.1.2	User.....	54
4.2	CodeIgniter.....	56
4.2.1	Was hat CodeIgniter zum Projekt beigetragen?.....	56
4.2.1.1	Aufbau.....	57
4.2.1.2	Beispiel.....	57
4.2.2	Ergänzungen .....	58
4.2.2.1	Validation.....	59
4.2.2.2	Mail / Activation.....	60
4.2.2.3	HTACCESS.....	61
4.2.2.4	Fake Klasse .....	61
4.2.2.5	Sicherheit.....	62
4.3	gasORM.....	66
4.3.1	Erstellung.....	66
4.3.2	Aufbau des Models .....	66
4.3.3	Einsatz im Controller .....	67
4.4	groceryCRUD .....	68

4.4.1	Controller.....	68
4.4.1.1	Columns.....	69
4.4.1.2	Relations .....	69
4.4.1.3	Change Header.....	69
4.4.1.4	Field type.....	69
4.4.1.5	Fields.....	70
4.4.1.6	Callbacks .....	70
4.4.1.7	Validation.....	71
4.4.2	View.....	71
4.4.3	Queries .....	72
4.5	CSS .....	72
4.5.1	Front.....	72
4.5.2	Backend .....	74
4.6	JavaScript .....	75
4.7	Fotogalerie .....	76
<b>Schluss teil</b>		<b>79</b>
5	Fazit .....	79
6	Stärken und Schwächen der Software .....	80
7	Zukunft .....	80
7.1	Verbesserungsmöglichkeiten .....	80
7.2	Weitere Verwendungszwecke.....	81
8	Quellenverzeichnis .....	82
8.1	Geschriebene Quellen.....	82
8.2	Weitere Quellen .....	82
9	Abkürzungsverzeichnis / Worterklärungen .....	83
10	Index.....	86
11	Ehrenwörtliche Erklärung .....	88
12	Anhang.....	89



## v. Abbildungsverzeichnis

Abbildung 1: HTML5 Support von verschiedenen Browsern (Stand 24.05.2012) .....	11
Abbildung 2: Framework distribution top 10k sites (Stand 24.05.2012) .....	12
Abbildung 3: Framework distribution top 100k sites (Stand 24.05.2012) .....	13
Abbildung 4: Framework distribution top 1mio sites (Stand 24.05.2012) .....	13
Abbildung 5: ASPNET Usage Trends (Stand 24.05.2012) .....	14
Abbildung 6: PHP Usage Trends (Stand 24.05.2012) .....	14
Abbildung 7: Datamapper (Stand 07.07.2012) .....	17
Abbildung 8: Code Beispiel NitroORM (Stand 07.07.2012) .....	18
Abbildung 9: Code Beispiel gasORM (Stand 07.07.2012) .....	18
Abbildung 10: Beispiel groceryCRUD (Stand 07.07.2012) .....	19
Abbildung 11: mobile html5 getUserMedia (Stand 08.06.2012) .....	20
Abbildung 12: canluse getUserMedia (Stand 08.06.2012) .....	20
Abbildung 13: canluse file API (Stand 08.06.2012) .....	21
Abbildung 14: mobile html5 file API (Stand 08.06.2012) .....	21
Abbildung 15: Login backend .....	32
Abbildung 16: Registerformular backend .....	33
Abbildung 17: Dashboard - user waiting for authorization - backend .....	33
Abbildung 18: Dashboard - changes back/front - backend .....	33
Abbildung 19: Haupttabelle Patient backend .....	36
Abbildung 20: Tabelle Allergy backend .....	38
Abbildung 21: Auswahl Episode gallery backend .....	38
Abbildung 22: Vorschaubilder gallery backend .....	39
Abbildung 23: Originalbild gallery backend .....	39
Abbildung 24: Selection Patient wizard backend .....	39
Abbildung 25: Selection Episode wizard backend .....	40
Abbildung 26: File Upload wizard backend .....	41
Abbildung 27: MAP wizard backend .....	41
Abbildung 28: Logbuch Overview backend .....	43
Abbildung 29: Detail Logbuch backend .....	44
Abbildung 30: Login frontend .....	45
Abbildung 31: Registration frontend .....	45
Abbildung 32: Dashboard frontend .....	45
Abbildung 33: Selection Patient gallery front .....	46

Abbildung 34: Selection Episode gallery frontend.....	46
Abbildung 35: Vorschaubilder gallery frontend .....	46
Abbildung 36: Originalbild gallery frontend .....	46
Abbildung 37: Selection Patient wizard frontend.....	47
Abbildung 38: Selection Episode wizard frontend.....	47
Abbildung 39: File Upload wizard frontend .....	48
Abbildung 41: File Upload Android V. 4.0.4 (Samsung Galaxy Neuxs).....	49
Abbildung 40: File Upload iOS mit iCabMobile Browser.....	49
Abbildung 42: MAP wizard front .....	49
Abbildung 43: Settings frontend.....	51
Abbildung 44: Datenbankmodel - BA.....	52
Abbildung 45: class extension prefix - CodeIgniter – BA.....	59
Abbildung 46: form_validation - CodeIgniter – BA.....	60
Abbildung 47: ashirra_form_validation - CodeIgniter - BA.....	60
Abbildung 48: ashirra_rnd_string - CodeIgniter - BA.....	61
Abbildung 49: htaccess - CodeIgniter – BA .....	61
Abbildung 50: Beispiel fake_controller - CodeIgniter - BA.....	62
Abbildung 51: Beispiel Captcha - prototype - BA .....	62
Abbildung 52: groceryCRUD View - prototype - BA.....	71
Abbildung 53: jquerymobile - Mobile Builder .....	72
Abbildung 54: jquerymobile - Themroller .....	73
Abbildung 55: Menu frontend .....	73
Abbildung 56: Design backend .....	74

## vi. Tabellenverzeichnis

Tabelle 1: Browservergleich für iOS.....	24
Tabelle 2: Browservergleich für iOS File Upload - CSS3 & HTML5 support.....	25
Tabelle 3: Möglicher Tabellenaufbau für multilinguale Lösung .....	26
Tabelle 4: Google Maps V3, Probleme mit Adressen .....	42
Tabelle 4: Google Maps V3, Probleme mit Adressen .....	50
Tabelle 6: CSS3 / HTML5 - Präfixe der Anbieter .....	75
Tabelle 7: Stärken und Schwächen der Software .....	80

# Hauptteil

---

## 1 Methodik

In diesem Abschnitt werde ich beschreiben, wie ich in dieser BA vorgegangen bin. Dieser Teil wird sich in folgende Unterkategorien aufteilen:

- [Technologien](#),
- [Technisches System](#),
- [Vorabklärungen](#),
- [Planung](#).

Im Punkt „Technologien“ erkläre ich, für welche Technologien ich mich entschieden habe. Ausserdem werden diese Entscheide begründet.

Nachdem die Technologien klar waren, musste ich mich für das „Technische System“ entscheiden. Hier werden die Installationen aufgelistet und kurze Erklärungen zu diesen Systemen abgegeben. Diese Erklärungen behandeln diverse Systeme wie die IDE, Server etc.

Gewisse „Vorabklärungen“ waren wichtig, da die gewählten Technologien noch in der Draft Version vorliegen und unterschiedlich implementiert werden. Fragen wie „Funktionieren Photo Uploads auf den diversen Systemen?“, „Wie sollen die Bilder gespeichert werden?“, als BLOB oder auf dem Filesystem des Servers?

Im Punkt „Planung“ werde ich den Ablauf der Sitzungen, mein Product Backlog und die technischen Hilfsmittel für die Planung erläutern.

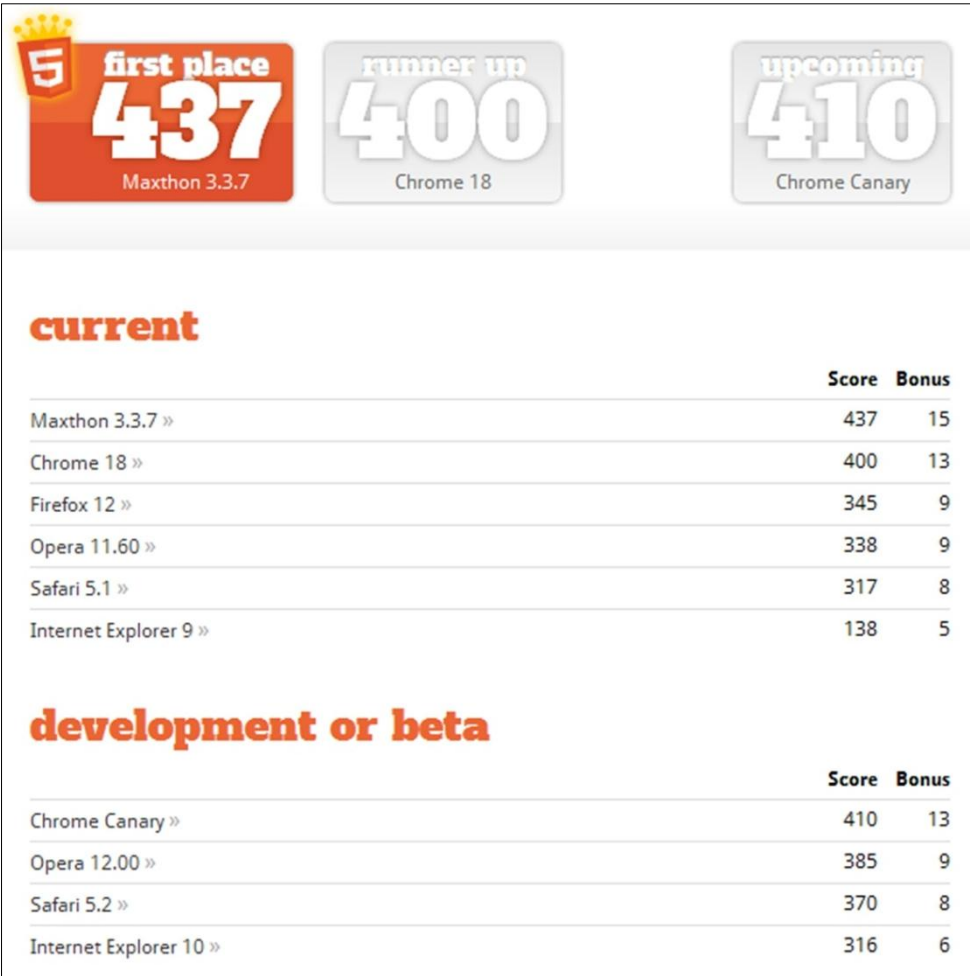
### 1.1 Technologien

Gewählte Technologien haben auf ein Projekt wie dieses einen grossen Einfluss. In den nachfolgenden Kapiteln werde ich meine Entscheide zu den einzelnen Technologien begründen. Die Titel widerspiegeln die gewählten Technologien. Mit den untenstehenden Links, kann man direkt zum gewünschten Thema springen.

- [HTML5](#),
- [PHP](#),
- [MySQL](#),
- [JavaScript](#).

### 1.1.1 HTML5<sup>1</sup>

Diese Technologie ist momentan in der Entwicklung und liegt als sogenannter Draft vor. Die meisten Browser haben HTML5 implementiert. Da es aber noch keinen Standard gibt und HTML5 noch im Draftzustand ist, wird dieser unterschiedlich unterstützt, sprich implementiert und interpretiert. Die meisten Grundfunktionen werden aber von jedem gängigen Browser bereits unterstützt. Ausserdem hat der Internet Explorer seit der Version 9 auch in diesem Bereich aufgeholt. Auf der Seite [html5test.com](http://html5test.com)<sup>2</sup> werden die Browser bewertet. (Abb. 1) Bei diesem Test werden Punkte vergeben, aber nur aufgrund der Implementation und nicht aufgrund des Handlings hinter dieser Implementation. Ein Beschrieb des Tests ist auf der Seite einsehbar.



first place		
5	437	Maxthon 3.3.7

runner up		
	400	Chrome 18

upcoming		
	410	Chrome Canary

current		
	Score	Bonus
Maxthon 3.3.7 »	437	15
Chrome 18 »	400	13
Firefox 12 »	345	9
Opera 11.60 »	338	9
Safari 5.1 »	317	8
Internet Explorer 9 »	138	5

development or beta		
	Score	Bonus
Chrome Canary »	410	13
Opera 12.00 »	385	9
Safari 5.2 »	370	8
Internet Explorer 10 »	316	6

Abbildung 1: HTML5 Support von verschiedenen Browsern (Stand 24.05.2012)

Trotz der unterschiedlichen Implementierungen bin ich davon überzeugt, dass HTML5 die „Zukunft“ ist und sein wird. Aufgrund der breiten Unterstützung durch die aktuellen Browser und die Möglichkeit ältere Browser mit Hilfe von JavaScript aufzuwerten, wird HTML5 weiter voranschreiten

<sup>1</sup> [http://www.w3schools.com/html5/html5\\_reference.asp](http://www.w3schools.com/html5/html5_reference.asp) (Stand 07.07.2012)

<sup>2</sup> <http://html5test.com/results/desktop.html> (Stand 24.05.2012)

und vermehrt zum Einsatz kommen. Durch den Einsatz dieser Technologie werden Applikationen plattformunabhängig und können in Windows, Android und iPhones aufgerufen werden. Darum habe ich mich für HTML5 entschieden. Meine Meinung wird dabei auch von Experten gestützt:

*Seit dem Start 2011 hat der Erfolg von HTML5 alle Erwartungen übertroffen. HTML5 wird nicht irgendein weiterer Standard, sondern vielmehr der Standard, nicht nur für Webseiten, sondern für alle Arten moderner Benutzerschnittstellen. HTML ist einfach anwendbar - in einer vertrauten Umgebung. Es bietet unzählige neue Features, die zuvor auf grundverschiedene, inkompatible und proprietäre Technologien verteilt waren. HTML5 scheint sein Versprechen zu halten.*

Andy Gryn, Senior Product Marketing Manager,  
 Automotive Marc Lapierre, QNX CAR Developer

Dieses Zitat stammt aus dem PDF „Warum HTML5 die HMI-Technologie der Zukunft ist“. <sup>3</sup>

### 1.1.2 PHP<sup>4</sup>

Zur Auswahl standen zwei Varianten:

- ASP.net<sup>5</sup> mit C#,
- PHP.

Beide haben zusammen im Internet einen Anteil von 58.74%<sup>6</sup> (Abb. 2) in den Top 10'000 Seiten.

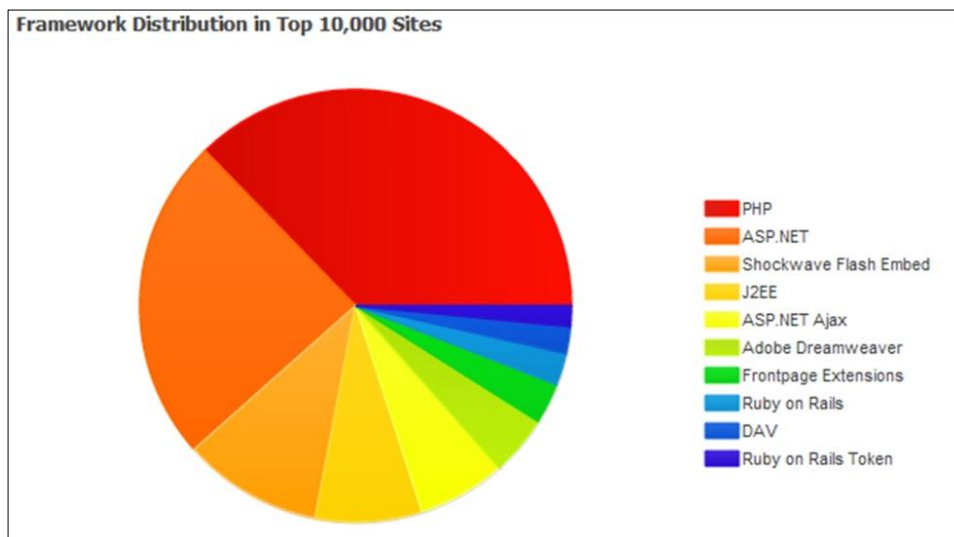


Abbildung 2: Framework distribution top 10k sites (Stand 24.05.2012)

<sup>3</sup> <http://www.qnx.com/download/feature.html?programid=23790> (Stand 08.08.2012)

<sup>4</sup> <http://php.org/> (Stand 07.07.2012)

<sup>5</sup> <http://www.asp.net/> (Stand 07.07.2012)

<sup>6</sup> <http://trends.builtwith.com> (Stand 18.05.2012)

Die untenstehende Abbildung (Abb.3) zeigt uns in einem Kuchendiagramm auf, wie stark die verschiedenen Sprachen in den Top 100'000 Seiten vertreten sind.

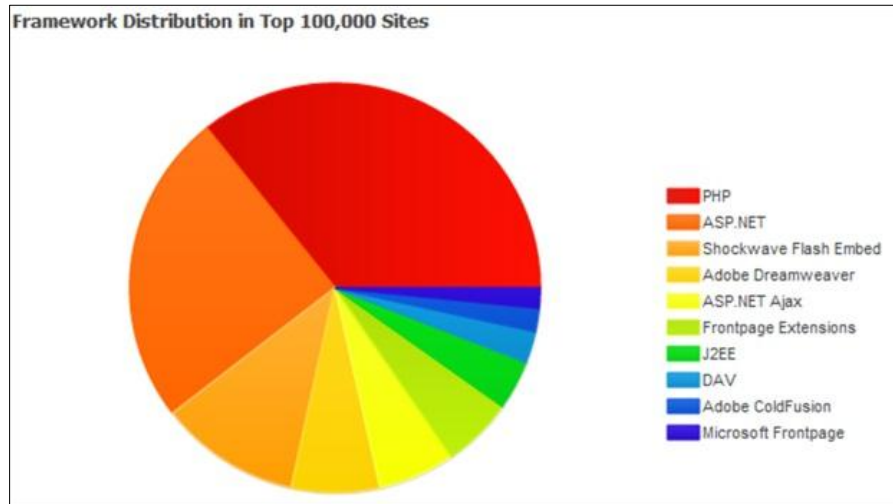


Abbildung 3: Framework distribution top 100k sites (Stand 24.05.2012)

Die untenstehende Abbildung (Abb.4) zeigt uns in einem Kuchendiagramm auf, wie stark die verschiedenen Sprachen in den Top 1'000'000 Seiten vertreten sind.

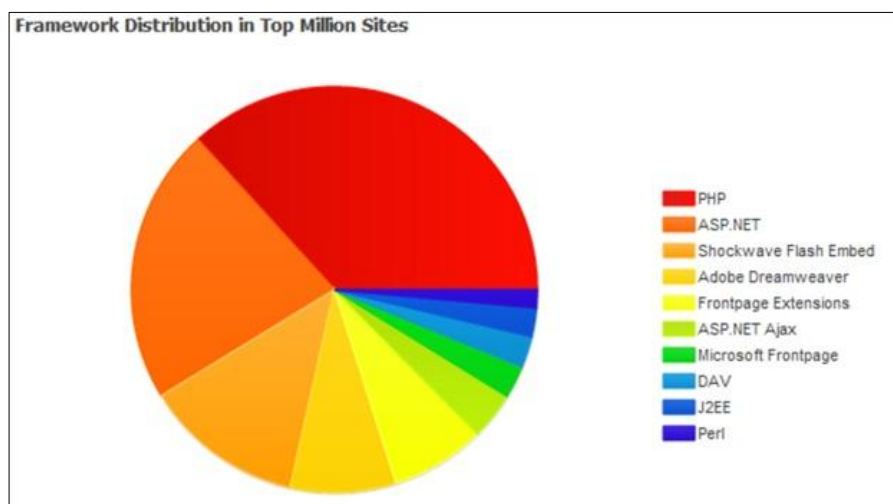


Abbildung 4: Framework distribution top 1mio sites (Stand 24.05.2012)

Wir können erkennen, dass sich der Anteil von ASP.net (ohne AJAX) und PHP unabhängig von den Top Seiten nur marginal verändert.

Die folgenden Abbildungen zeigen die Entwicklung von ASP.net (Abb. 5) und PHP (Abb. 6) über einen Zeitraum von einem Jahr.

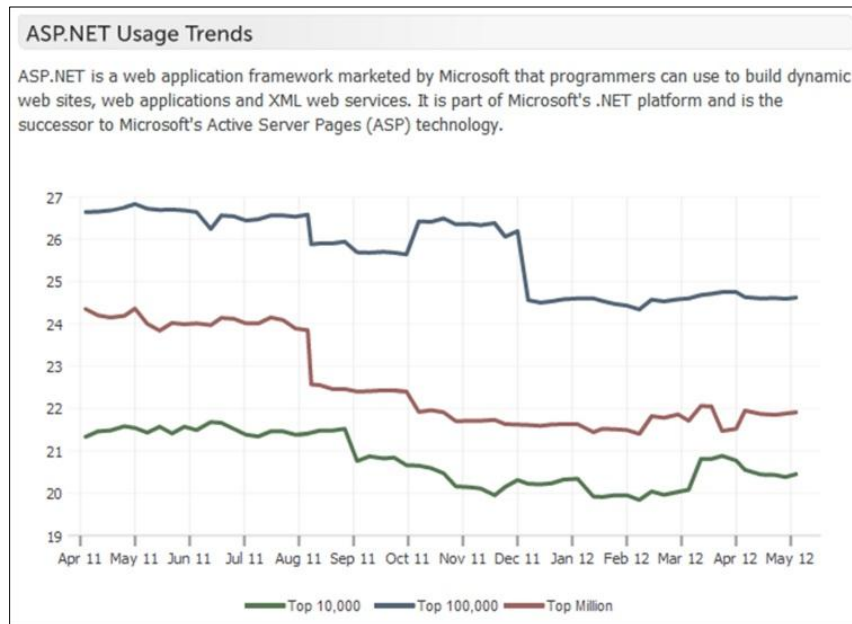


Abbildung 5: ASPNET Usage Trends (Stand 24.05.2012)

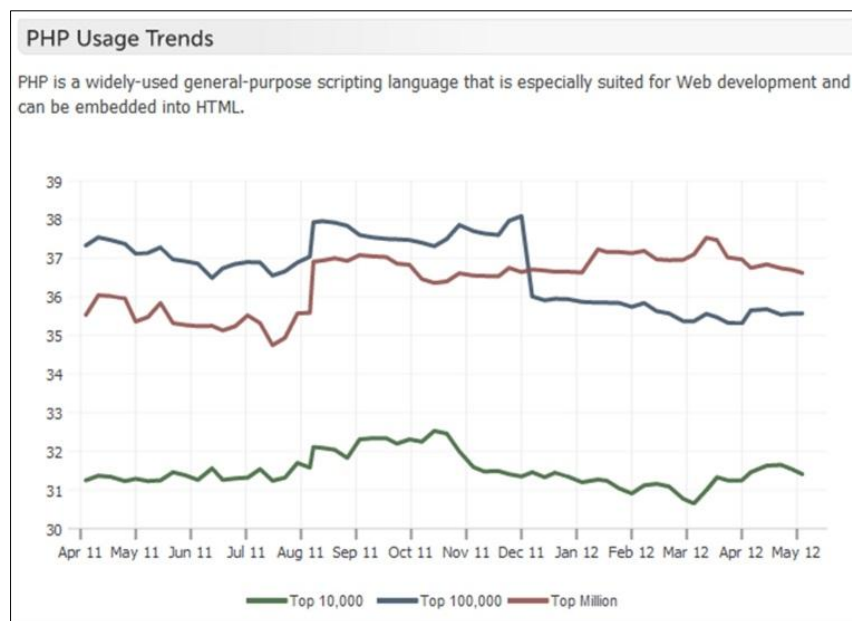


Abbildung 6: PHP Usage Trends (Stand 24.05.2012)

Warum ich mich für PHP entschieden habe, wird in der folgenden Auflistung gezeigt:<sup>7</sup>

- PHP hat einen grösseren Anteil am Gesamtmarkt und kommt auf einen Marktanteil von 36.75% (Stand 18.05.2012).

<sup>7</sup> <http://www.comentum.com/php-vs-asp.net-comparison.html> (Stand 24.05.2012)



- PHP und die dazugehörigen Softwares wie MySQL, „XAM(P)“ oder „LAM(P)“ sind nicht kostenpflichtig.
- Es sind gute, kostenlose IDEs für PHP erhältlich.
- Linux OS performt besser als auf Microsoft basierende Server bei Zugriffen auf das Filesystem z.B. für Images.
- C# ist eine schnellere Programmiersprache, aber dieser Geschwindigkeitsvorteil ist nur marginal und fällt somit nicht ins Gewicht und gegen PHP.

### 1.1.3 MySQL<sup>8</sup>

MySQL oder ähnliche Derivate werden oftmals mit PHP gebraucht. Zur Auswahl standen:

- MySQL,
- PostgreSQL,
- Microsoft SQL Server,
- SQLite,
- Oracle,
- Firebird.

PHP und MySQL werden von den meisten PHP Benutzern zusammen gebraucht. Ausserdem ist MySQL schon im LAMP<sup>9</sup> (Linux + Apache + MySQL + PHP) enthalten. In der Vergangenheit habe ich mit diesem Database Management System gute Erfahrungen gemacht. Foreign Key Support wird mittlerweile über InnoDB angeboten und auch Subselects sind möglich. Darum habe ich mich für die Software, MySQL, entschieden.

### 1.1.4 JavaScript<sup>10</sup>

Für die Client-Seite wird JavaScript oder Varianten davon verwendet. Varianten/Standards, welche zum Einsatz kommen könnten, sind zum Beispiel:

- AJAX (Asynchronous JavaScript and XML),
- JSON (JavaScript Object Notation),
- DOM (Document Object Model),

oder Bibliotheken wie:

- JQuery,
- DatePicker.

---

<sup>8</sup> <http://mysql.de/> (Stand 07.07.2012)

<sup>9</sup> <http://de.wikipedia.org/wiki/LAMP> (Stand 09.07.2012)

<sup>10</sup> <http://de.wikipedia.org/wiki/JavaScript> (Stand 07.07.2012)

## 1.2 Technisches System

Um die Applikation live zu testen, habe ich mich entschieden einen Server aufzusetzen. Dieser wurde mir von Herrn Abi Bossotto zur Verfügung gestellt.

Der Server enthält folgende Installationen:

Server: Linux CentOS<sup>11</sup>  
PHP: Version 5.3.X  
MySQL: Version 5.0.68

Alle gebrauchten Tools werden unten aufgeführt. Diese werden einzeln beschrieben und es wird erläutert, wieso ich mich für diese Tools/Technologien entschieden habe.

### 1.2.1 Programmierumgebung

- [NetBeans \(Version 7.1.2\)](#)
- [Virtuelle Maschine mit Win7](#)
- [PHP Framework CodeIgniter \(Version 2.1.0\)](#)
- [gasORM](#)
- [groceryCRUD](#)
- [FileZilla](#)

#### 1.2.1.1 NetBeans<sup>12</sup>

In der Vergangenheit haben wir immer mit anderen IDEs programmiert, unter diesen ist vor allem Eclipse zu erwähnen. Für meine BA habe ich mich aber für NetBeans entschieden, um meinen Horizont in Sachen IDEs zu erweitern.

#### 1.2.1.2 Virtuelle Maschine (VM)<sup>13</sup>

Damit ich unabhängiger von Computer/Laptop und Standort bin, habe ich mich entschieden eine VM mit einem Win7 Betriebssystem aufzusetzen. Diese VM läuft auf meiner externen Festplatte.

#### 1.2.1.3 PHP Framework<sup>14</sup>

Als PHP Framework standen mehrere zur Diskussion:

- Zend Framework,
- CodeIgniter,
- Symfony.

---

<sup>11</sup> <http://www.centos.org/> (Stand 07.07.2012)

<sup>12</sup> <http://netbeans.org/> (Stand 07.07.2012)

<sup>13</sup> <http://www.vmware.com/ch/> (Stand 07.07.2012)

<sup>14</sup> <http://codeigniter.com/> (Stand 07.07.2012)

Das Zend Framework und Symfony sind sehr gute, grosse und komplexe Frameworks. Man braucht darum viel Zeit, um sich in diese Frameworks einzuarbeiten.

Codelgniter ist hingegen ein leicht zu bedienendes Framework, welches eine gute und grosse Community hat. Es greift nicht zu stark in die Arbeit des Benutzers ein. Ausserdem ist die Dokumentation sehr gut strukturiert und mit Beispielen versehen. Zuerst wollte ich mich für das Zend Framework entscheiden, da ich aber nicht so viel Zeit für die Einarbeitung verwenden konnte, habe ich mich für die leichtere Variante, also Codelgniter, entschieden.

#### 1.2.1.4 *gasORM*<sup>15</sup>

gasORM ist eine Library für Codelgniter. Gemäss seinem Betreiber ist gasORM eine leichte und einfach zu benutzende Library für das object-relational mapping<sup>16</sup>. Zur Auswahl standen:

- gasORM,
- Doctrine,
- Datamapper ORM,
- NitroORM.

Wie unten in der Abbildung (Abb. 7) aufgeführt, fiel datamapper ORM weg, da das letzte Update 2 ½ Jahre zurück liegt.



**Datamapper ORM found a new home**  
Last modified by **WanWizard** on Wednesday, 12/29/2010 11:33:15 PM

Abbildung 7: Datamapper (Stand 07.07.2012)

Auf der anderen Seite hat sich „Doctrine“ auf den ersten Blick gut präsentiert. Da ich im Projekt viele neue Technologien, Libraries und ein Framework ausprobierte, fiel „Doctrine“ aufgrund des eigenen Abfrage-Dialekts aus der engeren Wahl.

NitroORM und gasORM waren ca. gleichwertig. Jedoch hat gasORM verschiedene Vorteile gegenüber NitroORM. gasORM kann die Modelle direkt von der Datenbank (DB) automatisch generieren lassen. Einzig die Relationen und eventuelle selbstgenerierte Primary Keys müssen manuell erstellt werden.

Nach dem Vergleich der Codes musste ich feststellen, dass gasORM einen kompakteren Code aufweist und meines Erachtens besser programmiert wurde. Die Modelle sind ausserdem reine Modelle und normale Abfragen werden in der Core Datei verarbeitet. Auch wurde mit Namespaces gearbeitet, was die Arbeit mit der Library übersichtlicher macht. Unten angeführt sehen wir zwei

<sup>15</sup> <http://gasORM-doc.taufanaditya.com/> (Stand 28.06.2012)

<sup>16</sup> [http://de.wikipedia.org/wiki/Objektrelationale\\_Abbildung](http://de.wikipedia.org/wiki/Objektrelationale_Abbildung) (Stand 30.07.2012)

Beispiele. Das erste zeigt den Code von NitroORM (Abb.8), das zweite die Aufstellung des Codes von gasORM (Abb.9).

```
<?php
// File: models/nitro/Person.php
class Person extends BasePerson
{
    public function getFullName( $order = array( 'last', 'first' ), $separator = ', ' )
    {
        $first = $this->getFirstName();
        $last = $this->getLastName();

        foreach( $order as $key )
            $name[] = ${$key};

        return implode( $separator, $name );
    }
}
```

Abbildung 8: Code Beispiel NitroORM (Stand 07.07.2012)

```
<?php

namespace Model;

use \Gas\Core;
use \Gas\ORM;

class User extends ORM {

    public $primary_key = 'id';

    function _init()
    {

        self::$relationships = array (
            'blog'           =>    ORM::has_many('\Model\Blog');
        );

        self::$fields = array(
            'id'              =>    ORM::field('auto[10]'),
            'username'        =>    ORM::field('char[64]'),
            'password'        =>    ORM::field('char[255]'),
            'email'           =>    ORM::field('char[255]'),
        );
    }

}
```

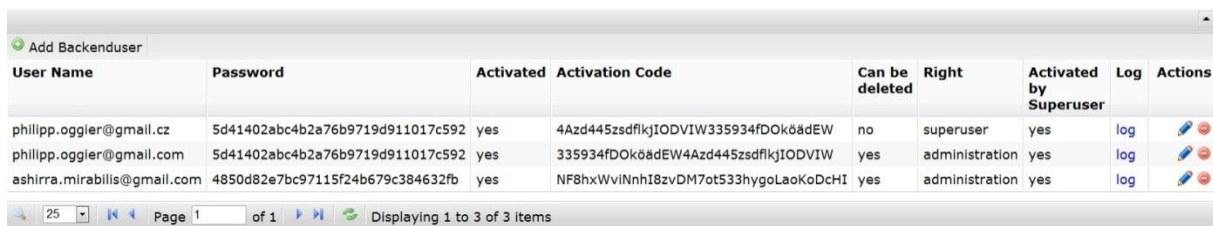
Abbildung 9: Code Beispiel gasORM (Stand 07.07.2012)







Aus diesen Gründen habe ich mich für gasORM entschieden.

### 1.2.1.5 *groceryCRUD*<sup>17</sup>

Um im Backend einfach und schnell Tabellen abbilden zu können, habe ich mich für die CodeIgniter Library “groceryCRUD” entschieden. Diese erlaubt es, nicht nur Tabellen anzuzeigen, es werden auch CRUD Funktionen unterstützt. Diese Library ist bereits getestet und wird von der CodeIgniter Community rege gebraucht. Neuigkeiten und Updates werden in kurzen Fristen immer wieder publiziert. Einziger Nachteil ist, dass der Code von nur einer Person entwickelt wird. In der nächsten Abbildung

(Abb. 11.) sehen wir ein Printscreen einer solchen Tabelle.



User Name	Password	Activated	Activation Code	Can be deleted	Right	Activated by Superuser	Log	Actions
philipp.oggier@gmail.cz	5d41402abc4b2a76b9719d911017c592	yes	4Azd445zsdflkjIODVIW335934fDOKöädEW	no	superuser	yes	log	 
philipp.oggier@gmail.com	5d41402abc4b2a76b9719d911017c592	yes	335934fDOKöädEW4Azd445zsdflkjIODVIW	yes	administration	yes	log	 
ashirra.mirabilis@gmail.com	4850d82e7bc97115f24b679c384632fb	yes	NF8hxWvINnh18zvDM7ot533hygoLaoKoDcHI	yes	administration	yes	log	 

Page 1 of 1  
Displaying 1 to 3 of 3 items

Abbildung 10: Beispiel groceryCRUD (Stand 07.07.2012)

### 1.2.1.6 *FileZilla*<sup>18</sup>

Um auf den Server per FTP zu zugreifen, habe ich FileZilla verwendet.

## 1.3 Vorabklärungen

### 1.3.1 Photo Upload

Trotz der weitläufigen Implementierung von HTML5 in den diversen Desktop- und Mobilebrowsern musste abgeklärt werden, wie weit der Camera access und die Unterstützung der File API in den diversen Browsern implementiert ist. Diese Abklärungen betreffen vor allem iOS Geräte, da Apple sehr strikt ist.

#### 1.3.1.1 *Camera access*

Gemäss diversen Abklärungen und basierend auf den Seiten [mobilehtml5.org](http://mobilehtml5.org/)<sup>19</sup> und [caniuse.com](http://caniuse.com/)<sup>20</sup> wird der direkte Zugang zur Kamera nur von Opera Mobile Version 12 implementiert und ist für Opera 12.0 und Chrome 21.0 geplant.

<sup>17</sup> <http://www.grocerycrud.com/> (Stand 03.07.2012)

<sup>18</sup> <http://www.filezilla.de/> (Stand 07.07.2012)

<sup>19</sup> <http://mobilehtml5.org/> (Stand 08.06.2012)

<sup>20</sup> <http://caniuse.com/> (Stand 08.06.2012 (letztes Update 27.05.2012))

Da wir dieses Problem umgehen können, indem wir einfach eine Foto schiessen und dieses später über die Gallery hochladen ([siehe File API](#)), ist dieses Manko vernachlässigbar.

# **getUserMedia/Stream API - Working Draft**

Method of accessing external device data (such as a webcam video stream). Formerly this was envisioned as the <device> element.

Usage stats: Global Support: 0.1%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
		3.6							
		8.0							
		9.0						10.0	2.1
	6.0	10.0				3.2		11.0	2.2
	7.0	11.0	17.0			4.0-4.1		11.1	2.3
	8.0	12.0	18.0	5.0		4.2-4.3		11.5	3.0
Current	9.0	13.0	19.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0
Near future	10.0	14.0	20.0	5.2	12.0				
Farther future		15.0	21.0	webkit					

Notes Known issues (0) Resources (2) Feedback

No notes

Abbildung 12: canuse getUserMedia (Stand 08.06.2012)

Feature	Safari on iOS	Android Browser	Google Chrome	Amazon Silk	BlackBerry Browser	Nokia Browser	Internet Explorer	Opera	Firefox	webOS Browser
Version tested	iPhone, iPad	Phones (1.0-2.3) Tablets (3.0+)	Android 4.0	Kindle Fire	Phones Tablet	Meego - Nokia N9 Symbian	Windows Phone	Mobile Mini	Android	Phones 1.x-2.x TouchPad 3.0
Notifications API W3C API Background alert notifications					✓ 2.0+				✓	
IndexedDB W3C API Agnostic database system (replacement for Web SQL)			✓						✓	
getUserMedia W3C API Camera access for <video> element								✓ 12+ (android)		
Animation Timing API W3C API Performant timers for HTML5 animations			✓						✓ 11+	✓
FullScreen API W3C API Allow the application to get a full screen mode			✓ Partial							
Page Visibility API W3C API Determine current visibility state			✓							
Remote Debugger Ability to attach a remote debugger, such as Web Inspector	✓ weinre & WebInspector	✓ weinre (inspector)	✓ weinre (inspector)	✓ usb debugging	✓ 7.0+	✓		✓ DragonFly		✓ weinre (inspector)

Abbildung 11: mobile html5 getUserMedia (Stand 08.06.2012)

Auf den Abbildungen (Abb. 12 + 13) sehen wir, wie gut **getUserMedia** bereits von den verschiedenen Browsern integriert worden ist. Es werden nur die wichtigsten und meistgebrauchten Browser aufgelistet. Ausserdem wird hier als Beispiel nur das **<video>** Tag aufgelistet. Diese Funktion ist aber auch für andere Medien zuständig, wie zum Beispiel Bilder direkt mit der Camera zu knipsen und über das **<input type="file">** hochzuladen.

„The **MediaStream** interface is used to represent streams of media data, typically (but not necessarily) of audio and/or video content, e.g. from a local camera or a remote site.“<sup>21</sup>

w3.org (W3C Working Draft)

<sup>21</sup> <http://www.w3.org/TR/webRTC/> (Stand 05.08.2012)



**Update:** Android Systeme mit dem neusten Betriebssystem (ab Version 4.0.4) unterstützen Kamera, Camcorder sowie Soundrekorder für das HTML5 Input Feld „File“.

### 1.3.1.2 File API

Aus „Sicherheitsgründen“ wird der File Upload bei Safari Mobile<sup>22</sup> ohne „Jailbreak“ nicht unterstützt. Dies könnte sich aber in Zukunft ändern.

Auf den Abbildungen (Abb. 13 + 14) sehen wir, wie gut das File API implementiert ist. Dies stellt uns eine Upload-Funktion für den HTML5-Tag `<input type="file">` zur Verfügung.

**When can I use File API?** [View in interactive mode](#)

Compatibility table for support of File API in desktop and mobile browsers.

■ = Supported 
 ■ = Not supported 
 ■ = Partially supported 
 ■ = Support unknown

**File API - Working Draft**

Method of manipulating file objects in web applications client-side, as well as programmatically selecting them and accessing their data.

Resources: [MDN article](#)

**Global user stats\*:**

Support:	51.91%
Partial support:	4.1%
Total:	56.01%

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
15 versions back			4.0						
14 versions back			5.0						
13 versions back		2.0	6.0						
12 versions back		3.0	7.0						
11 versions back		3.5	8.0						
10 versions back		3.6	9.0						
9 versions back		4.0	10.0						
8 versions back		5.0	11.0		9.0				
7 versions back		6.0	12.0		9.5-9.6				
6 versions back		7.0	13.0		10.0-10.1				
5 versions back		8.0	14.0		10.5				
4 versions back	5.5	9.0	15.0	3.1	10.6			10.0	
3 versions back	6.0	10.0	16.0	3.2	11.0	3.2		11.0	2.1
2 versions back	7.0	11.0	17.0	4.0	11.1	4.0-4.1		11.1	2.2
Previous version	8.0	12.0	18.0	5.0	11.5	4.2-4.3		11.5	2.3
Current	9.0	13.0	19.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0
Near future	10.0	14.0	20.0	5.2	12.0				
Farther future		15.0	21.0						

**Note:** Microsoft is currently [experimenting](#) with the technology. Partial support in Safari refers to lacking FileReader support.

Sub-features: [BlobBuilder API](#) [FileReader API](#) [Blob URLs](#)

Abbildung 13: canluse file API (Stand 08.06.2012)

Feature	Safari on iOS	Android Browser	Google Chrome	Amazon Silk	BlackBerry Browser	Nokia Browser	Internet Explorer	Opera	Firefox	webOS Browser
Version tested	iPhone, iPad	Phones (1.0-2.3) Tablets (3.0+)	Android 4.0 Kindle Fire		Phones Tablet	Meego - Nokia N9 Symbian	Windows Phone	Mobile Mini	Android	Phones 1.x-2.x TouchPad 3.0
WebGL Khronos Group API 3D Canvas for the web		✓ 2.3 only on Sony Xperia				✓ 2.0+			✓ 12+ (android)	
XMLHttpRequest 2.0 W3C API AJAX 2.0: upload files, progress	✓ 5.0+	✓ 4.0+	✓ 5.0+	✓ Partial		✓ 2.0+	✓ Belle+	✓ 12+	✓ 10+	✓ Partial
Navigation Timing API W3C API Performance events for WPO		✓ 4.0+	✓ 4.0+				✓		✓ 7+	
Network Information API W3C API Connection Type: 2G, 3G, 4G, WiFi		✓ 2.2+		✓						
File API W3C API Opening local files through input type		✓ 4.0+	✓			✓ 2.0+		✓ 12+ (partial)	✓ 11+	
CORS W3C API Cross origin Resource Sharing, for cross domain AJAX request	✓	✓ 4.0+	✓	✓		✓		✓ 12+	✓ 10+	✓
HTML Media Capture W3C API Taking pictures, record video, and audio from an input file type		✓ 4.0+	✓						✓ 11+	

Abbildung 14: mobile html5 file API (Stand 08.06.2012)

<sup>22</sup> <http://www.apple.com/de/safari/> (Stand 07.07.2012)

Um dieses Problem zu umgehen, stehen uns diverse Möglichkeiten zur Verfügung:

1. Email,
2. Browser,
3. App als „Proxy“,
4. Jailbreak.

### **Email**

Wir haben die Möglichkeit, dass ein iPhone/iPad-User ein Mail an die Administration schreiben kann. In der Mailfunktion können Bilder angehängt und verschickt werden. Diese müssen aber vom Administrator hochgeladen werden, was einen Mehraufwand bedeutet.

### **Browser**

Trotz den vielen Auflagen von Apple gibt es einige Browser, welche auf dem iPhone/iPad installiert werden dürfen. Zum Beispiel: Opera Mini, iCabMobile, iFox und andere.

### **App als „Proxy“**

Eine andere Möglichkeit diese Auflagen zu umgehen, ist der Einsatz von Native Apps. Diese können als „Proxy“ fungieren und den Upload auf eine gewünschte Website ermöglichen.

Ein Beispiel hierfür ist das App „aurigma up“<sup>23</sup>.

### **Jailbreak<sup>24</sup>**

Durch den Jailbreak ist es möglich, Softwares auf dem Gerät zu installieren, welche von Apple nicht offiziell freigegeben wurden. Zudem kann nach einem Jailbreak, der Apple eigene Browser (Safari), aufgewertet, beziehungsweise verändert werden.<sup>25</sup> Durch diese Änderung ist der Browser in der Lage Files hochzuladen.

Der Jailbreak ist mittlerweile auch in den USA erlaubt. Es gilt aber zu beachten, dass dabei die Garantie von Apple erlischt.<sup>26,27</sup>

### **1.3.1.3 Entscheidung**

Die Begründung der Lösungen ist nach der schlechtesten Lösung sortiert. Somit wird die letzte Lösung favorisiert.

---

<sup>23</sup> <http://www.aurigma.com/iphone/> (Stand 08.06.2012)

<sup>24</sup> [http://de.wikipedia.org/wiki/Jailbreak\\_\(iOS\)](http://de.wikipedia.org/wiki/Jailbreak_(iOS)) (Stand 08.06.2012)

<sup>25</sup> <http://www.funkyspacemonkey.com/safari-upload-enabler-enable-native-web-browser-file-uploading-iphone-ipad> (Stand 08.06.2012)

<sup>26</sup> <http://www.123recht.net/iPhone-Jailbreak-ist-legal-a72155.html> (Stand 08.06.2012)

<sup>27</sup> <http://www.conviva-plus.ch/index.php?page=367> (Stand 08.06.2012)



#### Möglichkeit 4 - Jailbreak

Ein Jailbreak für das iPhone/iPad kommt nicht in Frage, da

1. die Garantie erlischt,
2. sich die Rechtslage wieder ändern könnte,
3. das Gerät instabil werden kann und mit Performance Einbussen zu rechnen ist.

#### Möglichkeit 1 - Email

Ein Email ist theoretisch eine saubere und überschaubare Option. Diese Lösung würde aber einen administrativen Mehraufwand generieren, welchen es zu verhindern gilt.

#### Möglichkeit 3 - Proxy

Ein App als Vermittler zwischen iOS<sup>28</sup> und Browser zu nutzen, kann nur als Notlösung dienen. Es gilt zu beachten, dass wir ein externes App/einen Code brauchen, welches/welchen wir nicht völlig überwachen können. Somit stellt sich die Frage des Datenschutzes und der Sicherheit für sensitive Daten, wie z.B. klinische Bilder etc.

Ausserdem ist die Kontinuität der Weiterentwicklung, Support des Entwicklers, Support durch Apple selber (darf diese App noch Bilder hochladen) nicht gegeben.

#### Möglichkeit 2 - Browser (Lösung)

Einen von Apple erlaubten Browser zu installieren, welcher den Photo Upload unterstützt, bietet sich als einfache und effiziente Lösung an. Die Kontinuität von grösseren Browserherstellern ist besser gewährleistet, als die von App-Herstellern. Da wir direkt über einen Browser agieren, haben wir nicht ein App-in-the-middle, was die Sache zusätzlich vereinfacht, im Gegensatz zu [Möglichkeit 3](#).

Es war nicht leicht einen Browser zu finden, welcher den geforderten Ansprüchen genügt.

### 1.3.1.4 Browserwahl

#### Kriterien

- File-Upload gestattet
- CSS3 / HTML5 unterstützen
- Preisgrenze unter CHF 5.00

<sup>28</sup> <http://www.apple.com/de/ios/> (Stand 07.07.2012)

Getestet wurde nach den priorisierten Kriterien, siehe oben. Sobald ein Browser dem höchsten Kriterium nicht genügte, wurde dieser nicht weitergetestet.

### **Browser im Test (alphabetisch)**

**Tabelle 1: Browservergleich für iOS**

Browser	File Upload	CSS3 / HTML5	Preis < CHF 5.00
360 Lite	X		
Atomic Lite	X		
Bsecure	X		
DiigoBrowser	X		
Dolphin	X		
Geheimdienst Browser	X		
iCabMobile	✓	✓	✓
iFox	X		
Mango	X		
Mercury	X		
My Tools	X		
Opera Mini	✓	✓	✓
Sleipnir	X		
Snowbunny	X		

### **Test von CSS3 und HTML5**

Für den Test von CSS3 und HTML5 kamen folgende Tests zum Einsatz:

- Praxistest mit einer selber erstellten Testseite (Funktionalitäten, wie das Tabbing für die Formulare),
- Testseite für CSS3: [css3test.com](http://css3test.com),<sup>29</sup>
- Testseite für HTML5: [html5test.com](http://html5test.com).<sup>30</sup>

<sup>29</sup> <http://css3test.com> (Stand 11.06.2012)

<sup>30</sup> <http://html5test.com> (Stand 11.06.2012)

Tabelle 2: Browservergleich für iOS File Upload - CSS3 &amp; HTML5 support

	CSS3	HTML5
<b>Opera Mini</b>	43% (300/918 tests)	63 AND NO BONUS POINTS
<b>iCabMobile</b>	52% (455/918 tests)	324 AND 9 BONUS POINTS

Wie wir aus der Tabelle entnehmen können, sind Opera Mini und iCabMobile mit der CSS3 Unterstützung recht nahe beisammen. Hingegen herrscht eine riesige Kluft bei der Implementierung von HTML5. iCabMobile kann hier klar besser abschneiden.

### Entscheid

Der Opera Mini Browser ist gratis, hingegen ist der iCabMobile Browser mit CHF 2.00 zu veranschlagen.

Nach allen Tests entschied ich mich für den iCabMobile Browser für Apple Produkte. Dieser unterstützt die bisher benötigten Funktionen und wird auch laufend up-to-date gehalten.

Somit wird folgendes festgehalten (Stand 11.06.2012):

- iPhone / iPad => iCabMobile

Bei Android sind wir in einer glücklicheren Lage. Der Standard Browser von Android unterstützt alle benötigten Funktionen. Sollten wir uns für einen externen Browser entscheiden, können wir zum Beispiel auf den Opera Mobile (gratis) ausweichen.

- CSS3: 48%
- HTML5: 367/500
- benötigten Funktionen werden unterstützt

Die Entwicklung auf dem Gebiet CSS3 und HTML5 schreitet stetig voran. Ich bin gespannt, wie die einzelnen Browser am Ende meiner Arbeit abschliessen werden. Eventuell werden sich bis dahin auch andere Browser für meine Belange anbieten.

Mittlerweile hat Apple folgendes angekündigt (Ankündigung 11.6.2012):

*Additional new iOS 6 features include:*

*enhancements to Safari, the world's most popular mobile browser, such as iCloud tabs, offline reading lists, **photo uploads** and full screen view; (Apple, 2012)<sup>31</sup>*

Sollte iOS 6 vor dem Ende meiner Arbeit erscheinen, wird die obenerwähnte Lösung wohl hinfällig sein.

<sup>31</sup> <http://www.apple.com/pr/library/2012/06/11Apple-Previews-iOS-6-With-All-New-Maps-Siri-Features-Facebook-Integration-Shared-Photo-Streams-New-Passbook-App.html> (Stand 25.6.2012)

## 1.3.2 Multilingual

### 1.3.2.1 Codelgniter

Das Framework Codelgniter der Version 2.1.0 unterstützt eine minimale Mehrsprachigkeit. Diese Funktion eignet sich aber nur für die Labels und gemäss Aussagen in diversen Foren, sollte diese Funktion das Laden der Seite sehr verlangsamen.

### 1.3.2.2 Lösungsansätze

Um dieses Problem zu lösen, stehen mehrere Ansätze zur Verfügung, speichern des Contents in:

1. Flat Files,<sup>32</sup>
2. Tabelle mit allen Übersetzungen,
3. Datenbank mit einzelnen Übersetzungen pro Seite,
4. Datenbank mit Tabellen pro Sprache.

### 1.3.2.3 Lösung

#### Flat File versus Datenbank

Flat Files scheinen die einfachste Lösung zu sein. Diese können schnell von jedermann in einem einfachen Editor verändert, gelöscht und gespeichert werden. Jedoch bieten Flat Files nicht den Vorteil, welche eine Datenbank bieten kann.

Eine Datenbank kann:

- multiprocess/multithreading Zugriffe verwalten,
- sehr komplizierte Abfragen bewältigen,
- die Datenintegrität kann einfacher gewährleistet werden,
- Daten einfach updaten,
- Transactions und konkurrente Zugriffe verwalten.

Dies sind nur einige Vorteile einer Datenbank. Es ist ausserdem nur ein geringer Aufwand, da schon eine Datenbank benützt wird.

Somit schliesse ich Lösung 1 – Flat Files aus.

#### Datenbanklösung

Theoretisch würde eine Tabelle mit allen Übersetzungen reichen.

**Tabelle 3: Möglicher Tabellenaufbau für multilinguale Lösung**

Seite	Bezeichner	Sprache1	Sprache2	Sprache3
Imagecategory	Category	Korpus	corpus	corps

<sup>32</sup> <http://searchsqlserver.techtarget.com/definition/flat-file> (Stand 07.07.2012)

Dies hat den Vorteil, dass alle Sprachen denselben Bezeichner haben und somit Fehler verringert werden können. Es hat aber auch den Nachteil, dass man sehr unflexibel wird.

Des Weiteren kann pro Seite eine Übersetzungstabelle erstellt werden. Dies lohnt sich aber anhand der wenigen Übersetzungen nicht.

Aufgrund der Erweiterbarkeit bevorzuge ich die Lösung mit den einzelnen Tabellen pro Sprache. Dies ermöglicht auch das bessere Handling der Zugriffsrechte für Administratoren, welche für jeweils eine Sprache zuständig sind.

### 1.3.3 Bilder BLOB oder Ablage im Dateisystem

#### 1.3.3.1 BLOB<sup>33</sup>

BLOB bedeutet „Binary Large Objects“, dieses Format wird mittlerweile von vielen Datenbankmanagementsystemen gut beherrscht. Bilder können in diesem Format in die Datenbank gespeichert werden.<sup>34</sup>

##### Vorteile:

- Integrität der Bilder kann gewährleistet werden,
- Metadaten und die Bilddatei werden näher miteinander verknüpft,
- einfache Transportierbarkeit über Datenbank dump,
- Rechteverwaltung im Dateisystem entfällt,
- Einsatz von Triggern zur Versionierung und Zustandsdaten lassen sich leichter mit der Datei in der Datenbank verknüpfen.

##### Nachteile:

- Für das Laden jeder Bilddatei muss ein PHP-Prozess durchlaufen werden. Dieser Prozess dauert ca. 10-mal länger als das Laden durch den Webserver.
- Bilddateien in der Datenbank werden immer gesendet. Ansonsten könnte der Webserver ein "HTTP/1.x 304 Not modified" Header senden, um dem Browser anzuzeigen, dass ein Bild nicht verändert wurde.
- Unterschiedliche Implementierung von BLOB in Datenbanksystemen. MySQL kann BLOBs nur teilweise bearbeiten und hat zudem einen begrenzten Sendepuffer.

---

<sup>33</sup> [http://de.wikipedia.org/wiki/Binary\\_Large\\_Object](http://de.wikipedia.org/wiki/Binary_Large_Object) (Stand 30.07.2012)

<sup>34</sup> <http://www.php-faq.de/q-db-blob.html> (Stand 14.06.2012)

- Backup wird komplizierter. Bei einem Datenbank dump werden die Bilder ständig mitgesichert, obwohl diese keine Änderungen erfahren haben. Dies macht die Speichervolumen gross, diese sind fast nicht mehr zu komprimieren.

### **1.3.3.2 Entscheid**

Wie oben erwähnt wurde, bieten BLOBs Vor- und Nachteile. Aus nachgenannten Gründen werde ich mich gegen BLOBs entscheiden:

1. Integrität der Bilder ist sehr wichtig, kann aber auch mit einer externen Speicherung erreicht werden.
2. Metadaten und Bilder müssen nicht direkt miteinander verknüpft sein, dies setzt jedoch eine gute Logik und Kontrolle voraus.
3. Performance ist schlechter.
4. Fragmentarische Bearbeitung der Bilder kann erfolgen.
5. Das Backupvolumen der Datenbank kann klein gehalten werden, da die Bilder separat gesichert werden können. Bilder können so einfacher exportiert werden.

## **1.4 Planung**

Die Planung wird erst jetzt vorgestellt, da diese stark von den getroffenen Technologien abhängig ist. In der Planung werde ich erklären, wie das Projekt planungstechnisch aufgebaut ist, wie sieht die allgemeine Planung aus, wie stellt sich das Product Backlog zusammen und welche Hilfsmittel wurden verwendet.

### **1.4.1 Allgemeine Planung**

Zu Beginn der BA habe ich mich mit Herrn Dr. Henning Müller getroffen und das weitere Vorgehen besprochen. Dabei wurde klar, was für Ansprüche an die Applikation gestellt werden. Wir haben ausserdem entschieden, dass wir uns wöchentlich treffen und die neusten Ergebnisse besprechen. Dadurch war gewährleistet, dass ich auf dem richtigen Weg bleibe und kleine Korrekturen einfacher angebracht werden können. Während des Aufenthalts von Herrn Dr. Müller in China wurden die Sitzungen von Herrn Adrien Depeursinge abgehalten.

Wie in agilen Projektmethoden üblich, wurden wöchentliche Sitzungen abgehalten und ein Product Backlog erstellt. Trotzdem wurde das ganze Projekt nicht mit einer reinen agilen Methode geplant. Dies hat vor allem den Grund, dass der administrative Overhead verkleinert werden sollte.

Angelehnt an agile Methoden kamen folgende Teile zur Anwendung:

- Wöchentliche Sitzungen (Sprint = 1 Woche, inkl. Review),
- Laufende Prototypen,
- Product Backlog, später als mandatory/optional Teil geführt.

## 1.4.2 Product Backlog (PB)

Um alle Aspekte der Arbeit zu erfassen, wurde statt einem Pflichtenheft ein PB erstellt. Dieses wurde in folgende Bereiche unterteilt:

- Peripherie,
- Design,
- Backend,
- Frontend.

Das PB diene als erste Grundlage. Da die Vorgaben klar waren, wurde später auf eine reine agile Methode verzichtet. Es kamen aber trotzdem Teile von SCRUM zum Einsatz.

Das Product Backlog ist im Anhang enthalten.

### 1.4.2.1 Peripherie

Es wurde eine Liste mit Tools und Installationen erstellt, welche ausgeführt werden müssen. Unter diese Installationen gehören auch meine benutzten Programmierertools, eine Installation einer VM, damit ich unabhängiger vom Arbeitsplatz war und andere Tools.

### 1.4.2.2 Design

Planung der benötigten Design-Elemente. Als Beispiel:

*Ein Benutzer möchte genug grosse Buttons haben, welche auf dem Smartphone einfach getroffen werden können.*

Im Design-Teil des PB wurden die ersten Ideen für das Design festgehalten.

### 1.4.2.3 Backend

Im Teil Backend wurde erfasst, welche Anforderungen an die Verwaltung gestellt werden, was implementiert werden soll und welche Formulare/Daten in der Applikation verändert werden können.

### 1.4.2.4 Frontend

Im Frontend wurde beschrieben, welche Funktionen implementiert sein müssen oder können. Es kam auch zu späteren Änderungen, wie:

*Als Frontuser möchte ich ein "change request" für den client machen können.*

Dieser Teil wurde später anders gelöst, indem der Frontuser direkt Daten erfassen kann.



### 1.4.3 Tools

Auch ohne reine agile Methode musste ich irgendwie den Überblick bewahren. Damit ich dies vollbringen konnte, kamen folgende Tools und Technologien zum Einsatz:

- Excel-Tabellen,
- Online-Blog.

Mit Excel-Tabellen wurde vor allem das Product Backlog administriert. Im Online-Blog wurden alle Links mit den dazugehörigen Kommentaren gespeichert, damit man eine chronologische Reihenfolge hat, um diese später in der Dokumentation als Quelle zu gebrauchen.

# Resultate

---

## 2 Einleitung

Das System baut auf zwei Komponenten auf. Für die Verwaltung der Daten und der User ist das Backendsystem verantwortlich. Für den täglichen Gebrauch in Zusammenhang mit den Patienten ist das Frontend massgebend. Beide Systeme sind mit einem Login gesichert. Ausserdem kommt reCaptcha<sup>35</sup> von Google zum Einsatz, damit sich Maschinen nur schwer anmelden können. Das Design ist mit CSS3 erstellt und beim Frontend für Mobile-Geräte angepasst worden. Ausserdem wird ein selbsterstelltes JavaScript mit Google Maps V3 Technologie gebraucht, um nur einige Implementationen zu nennen.

In den folgenden Abschnitten werde ich die Software beschreiben und anschliessend auf die technischen Hintergründe eingehen.

## 3 Beschrieb der Software

### 3.1 Backend

#### 3.1.1 Login



The screenshot shows a login form titled 'LOGIN'. It contains the following fields and elements:

- USERNAME:** A text input field containing 'philipp.oggier@gmail.com'.
- PASSWORD:** A password input field with masked characters '\*\*\*\*\*'.
- REMEMBER ME:** A checkbox that is checked.
- Login:** A black button with white text.
- REGISTER:** A link at the bottom left.

Das Login ist einfach aufgebaut. Es beinhaltet folgende Felder:

- Username,
- Password,
- Button zum Login
- Link zur Registrierung.

Diese Felder werden beim Bestätigen des Buttons **[login]** per Validation überprüft und bei korrekter Eingabe auf die Home Seite weitergeleitet.

Abbildung 15: Login backend

---

<sup>35</sup> <http://www.google.com/recaptcha> (Stand 26.07.2012)

### 3.1.2 Registrierung

Abbildung 16:  
Registerformular backend

Um sich in das Backendsystem einzuloggen, muss sich der User zuerst registrieren. Auf der Registrationsseite muss er folgendes ausfüllen:

- Username (Muss Feld, einzigartig, gültige Emailadresse),
- Password (Muss Feld, Übereinstimmung mit Confirmation (zweites Password Feld) und muss je ein Gross-, Kleinbuchstaben und eine Nummer beinhalten),
- Password Confirmation (Muss Feld),
- reCaptcha.

Mit der Bestätigung auf den Button [\[register\]](#) wird man registriert, mit

[\[back\]](#) kommt man zurück auf die Login Seite.

### 3.1.3 Dashboard

Auf dem Dashboard werden drei wichtige Informationsteile aufgelistet:

USER	LINK
a@biggboss.ch	<a href="#">goto</a>

Abbildung 17: Dashboard - user waiting for authorization - backend

- Der erste Teil ist „User waiting for authorization“ und ist nur für die Superuser ersichtlich.

LAST CHANGES IN FRONTEND				
TIME	IDENTIFIER	CATEGORY	BACKEND USER	GOTO
2012-07-10 13:48:19		Update	philipp.oggier@gmail.cz	<a href="#">goto</a>
2012-07-10 13:47:52		Delete	philipp.oggier@gmail.com	<a href="#">goto</a>

LAST CHANGES IN BACKEND				
TIME	IDENTIFIER	CATEGORY	BACKEND USER	GOTO
2012-08-03 14:37:33	weight	Insert	philipp.oggier@gmail.cz	<a href="#">goto</a>
2012-08-03 14:36:21	weight	Insert	philipp.oggier@gmail.cz	<a href="#">goto</a>

Abbildung 18: Dashboard - changes back/front - backend

- Im zweiten und dritten Teil werden je die letzten fünf Änderungen im Backend sowie im Frontend wiedergegeben, also die letzten Einträge in den Logbüchern.
- Über einen Link kann zu den jeweiligen Daten noch eine Detailseite aufgerufen werden. Diese werden in den dazugehörigen Abschnitten erläutert.

### 3.1.4 User

Dieser Menüpunkt regelt die Verwaltung der Frontuser, Backenduser und der Relations zwischen Frontuser und Clients.

#### 3.1.4.1 Frontuser

In dieser Tabelle sind alle Frontuser ersichtlich. Diese Tabelle ist anpassbar und die Seiten sind nummeriert, eine Sortierfunktion und eine Suche sind integriert. [Search all](#) funktioniert aber nur mit Tabellen, welche mit keiner eigenen Column ergänzt wurden.

Es werden folgende Columns angezeigt:

- Username,
- Password (verschlüsselt mit MD5),
- Activated (0 -1 werden als yes oder no angezeigt),
- Activation code,
- Log (beinhaltet einen Link zu den Logs dieses Users),
- Clients (beinhaltet einen Link zu den jeweiligen Patienten dieses Users).

Die Tabelle hat eine Add-Funktion, welche es den Superusern erlaubt neue Frontuser zu erfassen. Bei der Erfassung wird gleichzeitig eine Mail an den neuen User verschickt, in welcher er den Account freischalten oder falls nicht gewünscht löschen kann.

Über die Edit-Funktion kann der Superuser den Account manuell freischalten oder den Usernamen ändern.

Über den Delete-Button kann ein User komplett gelöscht werden.

#### 3.1.4.2 Backenduser

In dieser Tabelle sind alle Backenduser ersichtlich. Diese Tabelle ist anpassbar und die Seiten sind nummeriert, eine Sortierfunktion und eine Suche sind integriert. [Search all](#) funktioniert aber nur mit Tabellen, welche mit keiner eigenen Column ergänzt werden.

Über den Sinn folgende Daten anzeigen zu lassen, kann man streiten. Fakt ist, dass folgende Kolonnen angezeigt werden:

- Username,
- Password (verschlüsselt mit MD5),
- Activated (0 -1 werden als yes oder no angezeigt),
- Activation code,
- Can be deleted (0 – 1 werden als yes oder no angezeigt),

- Right (superuser, administration),
- Log (beinhaltet einen Link zu den Logs dieses Users),
- Clients (beinhaltet einen Link zu den jeweiligen Patienten dieses Users).

Das Feld **can\_be\_deleted** ist bereits erfasst für zukünftige Erweiterungen. Im Moment wird dieses nicht verwendet.

Die Tabelle hat eine Add-Funktion, welche es den Superusern erlaubt neue Backenduser zu erfassen. Bei der Erfassung wird gleichzeitig eine Mail an den neuen User verschickt, in welcher er den Account freischalten oder falls nicht gewünscht löschen kann.

Über die Edit-Funktion kann der Superuser den Account manuell freischalten (**activation\_by\_superuser** und/oder **activated**), den Usernamen ändern oder die Rechte dieses Users setzen.

Über den Delete-Button kann ein User durch einen Superuser komplett gelöscht werden.

### **3.1.4.3 Relations**

In dieser Tabelle werden alle Frontuser und ihre Patienten aufgelistet. Das heisst, damit ein Frontuser den Patienten im Frontend überhaupt sehen kann, müssen diese miteinander verknüpft werden.

Über den Button **[Add Client to Frontuser]** kann man diese Relation zuordnen. Es erscheinen zwei Listen mit allen Frontusern und allen Patienten. Die Edit-Funktion ist gleich aufgebaut nur hat diese schon die vorher gesetzten Werte, da nur bestehende Einträge geändert werden können.

Alle Werte werden nicht mit dem Foreign Key angezeigt, sondern mit einem Wert, hier zum Beispiel der **user\_name** und die **client\_nr**.

### 3.1.5 Patient

Client	Birthday	Sex	Height	Weight	Allergy	Anamnese	Episode Of Care	Actions
PA1010	1985-06-13	female	1.65	<a href="#">weight - 10.1</a>	Pollen, Antibiothika	<a href="#">anamnese</a>	<a href="#">episodes</a>	
PA1013	2012-07-19	female	100	<a href="#">weight - 120</a>		<a href="#">anamnese</a>	<a href="#">episodes</a>	
PA1014	2012-07-24	female	1.89	<a href="#">weight</a>		<a href="#">anamnese</a>	<a href="#">episodes</a>	
PA1015	2012-07-25	female	1.86	<a href="#">weight</a>		<a href="#">anamnese</a>	<a href="#">episodes</a>	
PA1013	2012-07-18	male	1.86	<a href="#">weight</a>	Antibiothika	<a href="#">anamnese</a>	<a href="#">episodes</a>	
PA1022	2012-07-19	male	1.86	<a href="#">weight</a>		<a href="#">anamnese</a>	<a href="#">episodes</a>	
PA4040	1983-03-18	male	1.86	<a href="#">weight</a>	Pollen, Antibiothika	<a href="#">anamnese</a>	<a href="#">episodes</a>	
PA8888	1983-03-18	male	1.55	<a href="#">weight</a>	Pollen, Antibiothika	<a href="#">anamnese</a>	<a href="#">episodes</a>	
PA6666	1983-03-10	male		<a href="#">weight</a>		<a href="#">anamnese</a>	<a href="#">episodes</a>	
PA0110	1984-10-01	male	1.87	<a href="#">weight</a>	Pollen, Antibiothika	<a href="#">anamnese</a>	<a href="#">episodes</a>	

10 Page 1 of 2 Displaying 1 to 10 of 17 items

Abbildung 19: Haupttabelle Patient backend

Dieser Menüpunkt enthält zwei Unterpunkte. Im ersten Menüpunkt „Patient“ werden alle Informationen über den Patienten verwaltet. Im zweiten Menüpunkt „Allergy“ wird eine Liste aller Allergien erfasst, damit diese beim Patienten einheitlich sind und keine redundanten Daten generiert werden.

#### 3.1.5.1 Patientendaten

In dieser Tabelle werden alle Daten des Patienten verwaltet. Diese sind:

- Client (Nummer des Patienten),
- Birthday,
- Sex,
- Height,
- Weight (Link zur Liste aller Gewichtsangaben des Patienten, Anzeige des letzten Gewichts),
- Allergy (Verbunden mit der Allergieliste, alle Allergien werden aufgelistet),
- Anamnese (Link zu den Anamnese-Einträgen des Patienten),
- Episode of Care (Link zu den Episodes of Care des Patienten).

Diese Tabelle hat eine Funktion [\[Add Client\]](#). In dieser Tabelle können folgende Daten erfasst werden:

- Client (Muss dem Pattern entsprechen PA[0-9] [0-9] [0-9] [0-9], Muss Feld),
- Birthday (Bereich von heute -100 Jahre und dem heutigen Datum, Muss Feld),

- Sex ( Enum: female, male),
- Height (Dezimal mit zwei Stellen),
- Allergy (Kann von der Allergieliste ausgewählt werden).

### **Weight**

Hier wird das Gewicht des Patienten erfasst. Aufgelistet werden die verschiedenen Daten und das dazugehörige Gewicht des Patienten. Über die Add-Funktion können neue Daten erfasst werden:

- Die Patientennummer wird als readonly angezeigt, damit der User immer weiss, für wen er Daten anlegt,
- Date Weight (Muss Feld und die Datumsspanne gehen von heute bis – 30 Tage),
- Weight (Gewicht in Kilogramm mit max. 3 Nachkommastellen, 0.6kg – 300kg).

Ausserdem hat es wieder eine Edit- und eine Delete-Funktion.

### **Anamnese**

Hier werden die Anamnese-Texte des Patienten verwaltet. Die Daten sind direkt nach dem Erfassungsdatum sortiert.

Über die Add-Funktion können neue Daten erfasst werden:

- Die Patientennummer wird als readonly angezeigt, damit der User weiss, für wen er Daten anlegt,
- Datum (Muss Feld),
- Text (Muss Feld).

Ausserdem hat es wieder eine Edit- und eine Delete-Funktion.

### **Episode of Care**

Hier werden die Episoden des Patienten verwaltet. Die Daten sind direkt nach dem Beginn-Datum sortiert.

Über die Add-Funktion können neue Daten erfasst werden:

- Die Patientennummer wird als readonly angezeigt, damit der User weiss, für wen er Daten anlegt,
- Episode Name (Muss Feld),
- Begin (Muss Feld, die Datumsspanne geht von +30 Tage bis -180 Tage),
- End (die Datumsspanne geht von +30 Tage bis -180 Tage),
- First Diagnosis,
- End Diagnosis.

Die Datumsspannen gehen in die Vergangenheit und in die Zukunft, um Daten nachbeziehungsweise vor zu erfassen. Ausserdem hat es wieder eine Edit- und eine Delete-Funktion.

### 3.1.5.2 Allergy



Name	Actions
Antibiotika	[Edit] [Delete]
Pollen	[Edit] [Delete]

10 Page 1 of 1 Displaying 1 to 2 of 2 items

Abbildung 20: Tabelle Allergy backend

In dieser Liste werden alle Allergien verwaltet. Diese Liste wird einmalig bearbeitet und kann dann für jeden Patienten verwendet werden.

Über die Add-Funktion können neue Daten erfasst werden.

- Name (Nur alphabetische Zeichen sind zugelassen, einmalig)

Ausserdem hat es wieder eine Edit- und eine Delete-Funktion.

### 3.1.6 Fotogalerie

Um die Fotos der jeweiligen Patienten einzusehen, muss der Benutzer zuerst auf einer Auswahlseite eine Klienten Nummer aus einem Dropdown-Menu auswählen. Die Patienten sind mit ihrer eindeutigen Klienten Nummer vermerkt. Nach der Auswahl werden direkt darunter Angaben zu dem ausgewählten Klienten ausgegeben. Somit kann der Benutzer nochmals kontrollieren, ob er den richtigen ausgewählt hat. Mit dem Button [continue](#) kommt er auf die Fotogalerie.

Auf der Galerie sehen wir ein Menu, oben links einen Beschrieb, der darunter aufgelisteten Daten. Diese Listen widerspiegeln die Episodes of Care (id-name-begin). Oben rechts hat es noch ein about, welches auf die Firma verlinkt von welcher das Grundprogramm ist. Mit Close können wir wieder zurück auf die [gallery\\_client](#) Seite wechseln.



id	name	begin
7	TEST	2012-07-10
8	SDF	2012-07-19

Abbildung 21:  
Auswahl Episode  
gallery backend





Abbildung 22:  
Vorschaubilder gallery  
backend

- Name,
- Comment,
- Address (GeoData).



Abbildung 23: Originalbild gallery backend

### 3.1.7 Wizard

Hinter dem Wizard steckt die Idee ohne Unterbruch einen ganzen Ablauf abzuwickeln. Während meiner Arbeit habe ich mehrere Tests mit verschiedenen Interfaces gemacht. Zuerst waren die Patienten und Episoden getrennt vom Wizard. Wenn man aber etwas vergessen hatte, war ständiges Wechseln zwischen den Formularen vorprogrammiert. Durch die Tests mit meinen älteren Testpersonen (Familie und Bekannte) bin ich auf die Idee gekommen, einen Wizard zu integrieren. Dieser kam gut bei den Testern an.

Mit dem neuen Wizard ist es möglich, während des Erfassens direkt neue Klienten und Episoden zu erstellen, mit einem Klick erfassen und mit dem nächsten wieder zurück auf den Wizard zu wechseln. In den folgenden Unterkapiteln werden die Abläufe und die Möglichkeiten im Wizard beschrieben.

#### 3.1.7.1 Ablauf

Abbildung 24: Selection  
Patient wizard backend

#### Selection Patient

Hier kann ich die Patientennummer auswählen oder mit dem [\[new patient\]](#) Button einen neuen erstellen. Bei der Auswahl erscheinen Informationen zur ausgewählten Patientennummer ([client\\_nr](#), [age](#), [sex](#)). Mit [\[continue\]](#) kann man weiter zur Episodenauswahl gehen.

## New Patient

Patienten können unabhängig des Benutzerstatus erfasst werden. Im Frontend werden die Patienten nur mit den nötigsten Daten erfasst. Folgende Felder sind zugänglich und mit folgenden Werten zu versehen:

- Patient (Muss dem Pattern entsprechen **PA[0-9] [0-9] [0-9] [0-9]**, Muss Feld),
- Birthday (Bereich von -100 Jahre bis heute und dem heutigen Datum, Muss Feld),
- Sex (Enum: female, male),
- Height (Dezimalzahl mit zwei Stellen).

Diese Informationen können mit **[save]** gespeichert werden und/oder man kann mit **[back]** wieder zurück zu „Selection Patient“ gelangen.

## Selection Episode

Hier kann man die Episoden des Patienten auswählen oder mit dem „new episode“ Button eine neue erstellen. Bei Auswahl einer Episode erscheinen unter der Auswahl Informationen zur ausgewählten Episode (name, begin-end, first diagnosis, end diagnosis). Mit **[continue]** kann man weiter zur Fotogalerie gelangen.

Abbildung 25: Selection Episode wizard backend

## New Episode

Hier kann ich eine neue Episode erstellen. Folgende Felder sind zugänglich und mit folgenden Werten zu versehen:

- Die Patientennummer wird als readonly angezeigt, damit der User weiss, für wen er Daten anlegt,
- Name (Muss Feld),
- Begin (Muss Feld, die Datumsspanne geht von +30 Tage bis -180 Tage),
- End (die Datumsspanne geht von +30 Tage bis -180 Tage),
- First Diagnosis,
- End Diagnosis.

Die Datumsspannen gehen in die Vergangenheit und in die Zukunft um Daten nach- beziehungsweise vor zu erfassen. Diese Informationen können mit **[save]** gespeichert werden und/oder man kann mit **[back]** (wie **cancel**) wieder zurück zu „Selection Episode“ wechseln.

## File Upload

Auf dieser Seite können nun die Bilder hochgeladen werden. Es stehen folgende Felder zur Verfügung:

- File (Ein Bild mit dem Format: gif, png, jpg, jpeg),
- Name (nicht der Name des Files sondern der, welcher später angezeigt wird),
- Category (Liste von Kategorien z.B. corpus, underarm etc.),
- Comment (Text mit max. 150 Zeichen).

Darunter befinden sich zwei verschiedene Buttons [\[finish\]](#) und [\[GeoData\]](#).

Wir haben nun die Möglichkeit mit finish direkt das Bild mit den dazugehörigen Daten hochzuladen.

Nach dem Hochladen des erfassten Bildes können wir direkt ein weiteres Bild dieses Patienten in der ausgewählten Episode hochladen.

Entscheiden wir uns hingegen für [\[GeoData\]](#) kommen wir auf eine neue Seite.

Abbildung 26: File Upload wizard backend

## GeoData

Auf dieser Seite haben wir die Möglichkeit Geolocation Information zu diesem Bild hochzuladen. Wir haben auch hier wiederum mehrere Möglichkeiten:

1. MAP [\[LocateME\]](#),
2. Doppelklick auf die Map,
3. Adresszeile.

Mit dem [\[LocateME\]](#) können wir unsere Position sehr genau ermitteln. Damit wir unsere Position ermitteln lassen können, müssen wir dem Browser die dazu nötige Erlaubnis erteilen. Wenn unsere Position gefunden worden ist, werden direkt Adresse, Längen- und Breitengrad von Google ermittelt und in die dazugehörigen Felder geschrieben.

Eine andere Möglichkeit die Adresse zu ermitteln, ist ein Doppelklick auf die Karte. Der Positionsmarker wird neu

Abbildung 27: MAP wizard backend

gesetzt, falls dieser schon existiert. Es kann sich somit immer nur ein Positionsmarker auf der Karte befinden. Adresse, Längen- und Breitengrad werden von Google ermittelt und in die dazugehörigen Felder geschrieben.

Als letztes haben wir die Möglichkeit unsere Position über die Adresszeile selber zu finden. Hier müssen wir nur unsere Adresse eingeben, in einer Liste unter dem Adressfeld werden von Google Vorschläge via Autovervollständigung gemacht. Neue Adressen werden zum Teil nicht gefunden, da diese bei Google noch nicht aktuell sind. Ausserdem werden manche Postleitzahlen den politischen Gemeinden zugeteilt. Das unten aufgeführte Beispiel zeigt anschaulich das Problem auf.

**Tabelle 4: Google Maps V3, Probleme mit Adressen**

	Neue Adresse	Alte Adresse	Vorschlag neue Daten	Vorschlag alte Daten
<b>Strasse</b>	Feithierenstrasse	Obere Feithierenstrasse	Obere Feithierenstrasse	Obere Feithierenstrasse
<b>Nr.</b>	142	32	2	32
<b>PLZ / Ort</b>	3952 Susten	3952 Susten	3952 Leuk	3952 Leuk
<b>Land</b>	Switzerland	Switzerland	Switzerland	Switzerland

Alte Daten	Falsch	Korrekt
------------	--------	---------

Da die Adresse noch nicht im System nacherfasst worden ist und vor der Neuadressierung keine solch hohen Hausnummern vorhanden waren, versucht Google eine Adresse zu finden. In diesem Fall wird die letzte Ziffer der Hausnummer als richtig interpretiert und zusätzlich die alte Strasse genommen.

Uns fällt auf, dass die Postleitzahl richtig zugewiesen wurde. Aber es wird der Ort der politischen Gemeinde angegeben und nicht der Ort zu welchem die Postleitzahl eigentlich gehört.

**3953 Leuk**

**3952 Susten**

Trotz dieses kleinen Mankos ist dieses Feature recht nützlich, um schnell Adressen zu finden.

Auch auf dieser Seite haben wir wieder zwei Möglichkeiten zum Navigieren.

- [\[finish\]](#)
- [\[back\]](#)

Bei [\[back\]](#) werden keine Geolocation-Daten gespeichert. Wir kehren auf die Upload-Seite zurück und können ein neues Bild erfassen. Im Wizard können wir aber keine Geolocation-Daten für das alte Bild nacherfassen.

Bei [\[finish\]](#) werden die Geolocation-Daten zu diesem Bild gespeichert.

Auch bei dieser Auswahl werden wir zurück auf die Upload-Seite verwiesen und wir können weiter neue Bilder hochladen. Somit können wir andere Bilder dieses Patienten in der ausgewählten Episode hochladen.

## 3.1.8 Settings

Im Menüpunkt Settings hat es Unterkategorien. Diese heissen „Own Settings“, Log Back und Log Front.

### 3.1.8.1 Own Settings

Die „Own Settings“ sind gleich aufgebaut wie die anderen Tabellen. Es erscheinen der Username, das zugeordnete Recht und ein Link zum eigenen Logbuch. Diese Tabelle hat keine Add-Funktion sondern nur eine Edit- und Delete-Funktion. Man kann nur das Password ändern.

- Password (Muss Feld, Übereinstimmung mit Confirmation und muss je ein Gross-, Kleinbuchstaben und eine Nummer beinhalten)
- Password Confirmation (Muss Feld)

### 3.1.8.2 Log Back / Log Front

Diese Tabellen beinhalten die Logs der jeweiligen Systeme (Frontend, Backend). Sie sind folgendermassen aufgebaut:

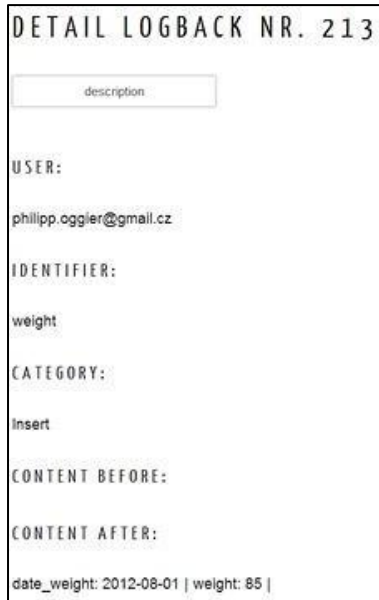
- Time (Datum und Zeit des Events),
- Identifier (Dazugehörige Tabelle, wo der Event stattgefunden hat),
- Category (Insert, Delete, Update),
- Frontend/Backend User (Wer für den Event zuständig war),
- Detail (Link zur Detailseite).

Time	Identifier	Category	Backend User	Detail
2012-08-03 - 14:37:33	weight	Insert	philipp.oggier@gmail.cz	<a href="#">detail</a>
2012-08-03 - 14:36:21	weight	Insert	philipp.oggier@gmail.cz	<a href="#">detail</a>
2012-08-03 - 14:33:47	weight	Insert	philipp.oggier@gmail.cz	<a href="#">detail</a>
2012-08-03 - 14:32:17	weight	Insert	philipp.oggier@gmail.cz	<a href="#">detail</a>
2012-08-03 - 14:26:50	relation	Insert	philipp.oggier@gmail.cz	<a href="#">detail</a>
2012-08-03 - 14:26:36	relation	Insert	philipp.oggier@gmail.cz	<a href="#">detail</a>
2012-07-30 - 10:25:22	relation	Insert	philipp.oggier@gmail.com	<a href="#">detail</a>
2012-07-27 - 15:45:57	relation	Insert	philipp.oggier@gmail.com	<a href="#">detail</a>
2012-07-27 - 15:33:45	frontuser	Insert	philipp.oggier@gmail.com	<a href="#">detail</a>
2012-07-26 - 15:22:41	episode	Insert	philipp.oggier@gmail.com	<a href="#">detail</a>

Abbildung 28: Logbuch Overview backend

Logs können nicht angelegt werden, da diese vom System im Hintergrund generiert werden oder besser gesagt bei einem Event generiert werden. Ausserdem können diese nicht editiert und nicht gelöscht werden.

### 3.1.8.3 Log Detail



Auf einer eigenen Seite werden alle Informationen über diesen Logeintrag zusammengetragen und aufgelistet.

- User (not exist anymore, wenn der User gelöscht wurde)
- Identifier (Dazugehörige Tabelle, wo der Event stattgefunden hat)
- Category (Insert, Delete, Update)
- Content before (Nur bei delete und update)
- Content after (Nur bei insert und update)

Abbildung 29: Detail Logbuch backend

### 3.1.9 Logout

Das Logout zerstört die Session und leitet den Benutzer auf die Login-Seite weiter.

## 3.2 Frontend

### 3.2.1 Login

Das Login ist einfach aufgebaut. Es beinhaltet folgende Felder:

- Username,
- Password,
- Link zur Registrierung.

Diese Felder werden beim Bestätigen des Buttons [\[login\]](#) per Validation überprüft und bei korrekter Eingabe auf die Home-Seite weitergeleitet.

Abbildung 30: Login frontend

### 3.2.2 Registrierung

Abbildung 31: Registration frontend

Um sich in das Frontendsystem einzuloggen, muss sich der User zuerst registrieren. Auf der Registrierungsseite muss er folgendes ausfüllen:

- Username (Muss Feld, einzigartig, gültige Emailadresse),
- Password (Muss Feld, Übereinstimmung mit Confirmation und muss je ein Gross-, Kleinbuchstaben und eine Nummer beinhalten),
- Password Confirmation (Muss Feld),
- reCaptcha.

### 3.2.3 Dashboard

Auf dem Dashboard werden die letzten Aktivitäten des eingeloggten Users aufgezeigt. Über einen Link kann zu den jeweiligen Daten noch eine Detailseite aufgerufen werden. Dies wird in den dazugehörigen Abschnitten noch erläutert.

TIME	IDENTIFIER	CATEGORY	GOTO
2012-07-10 13:48:19		Update	<a href="#">goto</a>

Abbildung 32: Dashboard frontend

### 3.2.4 Fotogalerie



Abbildung 33: Selection Patient gallery front

Um die Fotos der jeweiligen Patienten einzusehen, muss der Benutzer zuerst auf einer Auswahlseite eine Klienten Nummer aus einem Dropdown-Menu auswählen. Die Patienten sind mit ihrer eindeutigen Klienten Nummer vermerkt. Nach der Auswahl werden direkt darunter Angaben zu dem ausgewählten Klienten ausgegeben. Somit kann der Benutzer nochmals kontrollieren, ob er den richtigen ausgewählt hat. Mit dem Button **continue** kommt er auf die Fotogalerie.



Abbildung 34: Selection Episode gallery frontend

Auf der Galerie sehen wir ein Menu, oben links einen Beschrieb, der darunter aufgelisteten Daten. Diese Listen widerspiegeln die Episodes of Care (id-name-begin). Oben rechts hat es noch ein about, welches auf die Firma verlinkt von welcher das Grundprogramm ist. Mit Close können wir wieder zurück auf die **gallery\_client** Seite wechseln.

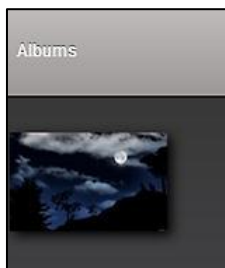


Abbildung 35: Vorschaubilder gallery frontend

Wenn wir auf einen Listeneintrag klicken, kommen wir auf eine Seite mit allen Thumbnails dieser Episode. Mit einem Klick auf ein Thumbnail kommen wir auf die Fotoansicht. Hier werden die Fotos angezeigt und darunter sehen wir zusätzliche Informationen:

- Name,
- Comment,
- Address (GeoData).



Abbildung 36: Originalbild gallery frontend

### 3.2.5 Wizard

Hinter dem Wizard steckt die Idee ohne Unterbruch einen ganzen Ablauf abzuwickeln. Während meiner Arbeit habe ich mehrere Tests mit verschiedenen Interfaces gemacht. Zuerst waren die Patienten und Episoden getrennt vom Wizard. Wenn man aber etwas vergessen hatte, war ständiges Wechseln zwischen den Formularen vorprogrammiert. Durch die Tests mit meinen älteren Testpersonen (Familie und Bekannte) bin ich auf die Idee gekommen, einen Wizard zu integrieren. Dieser kam gut bei den Testern an.



Mit dem neuen Wizard ist es möglich, während des Erfassens direkt neue Klienten und Episoden zu erstellen, mit einem Klick erfassen und mit dem nächsten wieder zurück auf den Wizard zu wechseln. In den folgenden Unterkapiteln werden die Abläufe und die Möglichkeiten im Wizard beschrieben.

### 3.2.5.1 Ablauf

#### Selection Patient

Hier kann ich die Patientennummer auswählen oder mit dem **[new patient]** Button einen neuen erstellen. Bei der Auswahl erscheinen Informationen zur ausgewählten Patientennummer (**client\_nr**, **age**, **sex**). Mit **[continue]** kann man weiter zur Episodenauswahl gehen.

Abbildung 37: Selection Patient wizard frontend

#### New Patient

Patienten können unabhängig des Benutzerstatus erfasst werden. Im Frontend werden die Patienten nur mit den nötigsten Daten erfasst. Folgende Felder sind zugänglich und mit folgenden Werten zu versehen:

- Patient (Muss dem Pattern entsprechen **PA[0-9] [0-9] [0-9] [0-9]**, Muss Feld),
- Birthday (Bereich von -100 Jahre bis heute und dem heutigen Datum, Muss Feld),
- Sex (Enum: female, male),
- Height (Dezimalzahl mit zwei Stellen).

Diese Informationen können mit **[save]** gespeichert werden und/oder man kann mit **[back]** wieder zurück zu „Selection Patient“ gelangen.

#### Selection Episode

Hier kann man die Episoden des Patienten auswählen oder mit dem „new episode“ Button eine neue erstellen. Bei Auswahl einer Episode erscheinen unter der Auswahl Informationen zur ausgewählten Episode (name, begin-end, first diagnosis, end diagnosis). Mit **[continue]** kann man weiter zur Fotogalerie gelangen.

Abbildung 38: Selection Episode wizard frontend

## New Episode

Hier kann ich eine neue Episode erstellen. Folgende Felder sind zugänglich und mit folgenden Werten zu versehen:

- Die Patientennummer wird als readonly angezeigt, damit der User weiss, für wen er Daten anlegt,
- Name (Muss Feld),
- Begin (Muss Feld, die Datumsspanne geht von +30 Tage bis -180 Tage),
- End (die Datumsspanne geht von +30 Tage bis -180 Tage),
- First Diagnosis,
- End Diagnosis.

Die Datumsspannen gehen in die Vergangenheit und in die Zukunft um Daten nach- beziehungsweise vor zu erfassen. Diese Informationen können mit [\[save\]](#) gespeichert werden und/oder man kann mit [\[back\]](#) (wie [cancel](#)) wieder zurück zu „Selection Episode“ wechseln.

## File Upload

Auf dieser Seite können nun die Bilder hochgeladen werden. Es stehen folgende Felder zur Verfügung:

- File (Ein Bild mit dem Format: gif, png, jpg, jpeg),
- Name (nicht der Name des Files sondern der, welcher später angezeigt wird),
- Category (Liste von Kategorien z.B. corpus, underarm etc.),
- Comment (Text mit max. 150 Zeichen).

Darunter befinden sich zwei verschiedene Buttons [\[finish\]](#) und [\[GeoData\]](#).

Wir haben nun die Möglichkeit mit finish direkt das Bild mit den dazugehörigen Daten hochzuladen.

Abbildung 39: File Upload wizard frontend

Nach dem Hochladen des erfassten Bildes können wir direkt ein weiteres Bild dieses Patienten in der ausgewählten Episode hochladen.

Entscheiden wir uns hingegen für [\[GeoData\]](#) kommen wir auf eine neue Seite.

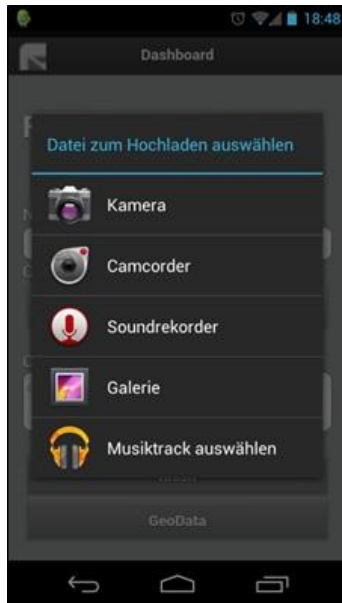


Abbildung 40: File Upload Android V. 4.0.4 (Samsung Galaxy Neuxs)

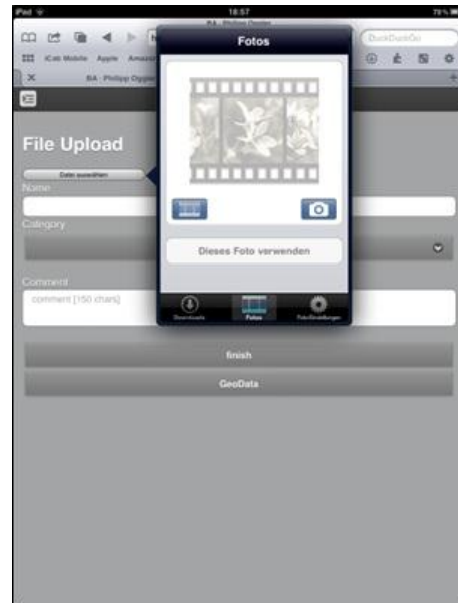


Abbildung 41: File Upload iOS mit iCabMobile Browser

## GeoData

Auf dieser Seite haben wir die Möglichkeit Geolocation Information zu diesem Bild hochzuladen. Wir haben auch hier wiederum mehrere Möglichkeiten:

4. MAP [[LocateME](#)],
5. Doppelklick auf die Map,
6. Adresszeile.

Mit dem [[LocateME](#)] können wir unsere Position sehr genau ermitteln. Damit wir unsere Position ermitteln lassen können, müssen wir dem Browser die dazu nötige Erlaubnis erteilen. Wenn unsere Position gefunden worden ist, werden direkt Adresse, Längen- und Breitengrad von Google ermittelt und in die dazugehörigen Felder geschrieben.

Eine andere Möglichkeit die Adresse zu ermitteln, ist ein Doppelklick auf die Karte. Der Positionsmarker wird neu gesetzt, falls dieser schon existiert. Es kann sich somit immer nur ein Positionsmarker auf der Karte befinden. Adresse, Längen- und Breitengrad werden von Google ermittelt und in die dazugehörigen Felder geschrieben.

Als letztes haben wir die Möglichkeit unsere Position über die Adresszeile selber zu finden. Hier müssen wir nur unsere



Abbildung 42: MAP wizard front

Adresse eingeben, in einer Liste unter dem Adressfeld werden von Google Vorschläge via Autovervollständigung gemacht. Neue Adressen werden zum Teil nicht gefunden, da diese bei Google noch nicht aktuell sind. Ausserdem werden manche Postleitzahlen den politischen Gemeinden zugeteilt. Das unten aufgeführte Beispiel zeigt anschaulich das Problem auf.

**Tabelle 5: Google Maps V3, Probleme mit Adressen**

	Neue Adresse	Alte Adresse	Vorschlag neue Daten	Vorschlag alte Daten
<b>Strasse</b>	Feithierenstrasse	Obere Feithierenstrasse	Obere Feithierenstrasse	Obere Feithierenstrasse
<b>Nr.</b>	142	32	2	32
<b>PLZ / Ort</b>	3952 Susten	3952 Susten	3952 Leuk	3952 Leuk
<b>Land</b>	Switzerland	Switzerland	Switzerland	Switzerland

Alte Daten	Falsch	Korrekt
------------	--------	---------

Da die Adresse noch nicht im System nacherfasst worden ist und vor der Neuadressierung keine solch hohen Hausnummern vorhanden waren, versucht Google eine Adresse zu finden. In diesem Fall wird die letzte Ziffer der Hausnummer als richtig interpretiert und zusätzlich die alte Strasse genommen.

Uns fällt auf, dass die Postleitzahl richtig zugewiesen wurde. Aber es wird der Ort der politischen Gemeinde angegeben und nicht der Ort zu welchem die Postleitzahl eigentlich gehört.

**3953 Leuk**

**3952 Susten**

Trotz dieses kleinen Mankos ist dieses Feature recht nützlich, um schnell Adressen zu finden.

Auch auf dieser Seite haben wir wieder zwei Möglichkeiten zum Navigieren.

- [\[finish\]](#)
- [\[back\]](#)

Bei [\[back\]](#) werden keine Geolocation-Daten gespeichert. Wir kehren auf die Upload-Seite zurück und können ein neues Bild erfassen. Im Wizard können wir aber keine Geolocation-Daten für das alte Bild nacherfassen.

Bei [\[finish\]](#) werden die Geolocation-Daten zu diesem Bild gespeichert.

Auch bei dieser Auswahl werden wir zurück auf die Upload-Seite verwiesen und wir können weiter neue Bilder hochladen. Somit können wir andere Bilder dieses Patienten in der ausgewählten Episode hochladen.

### 3.2.6 Settings

Bei den Settings erscheinen der Username und das Password des eingeloggten Benutzers.

Im Moment kann nur das Password geändert werden.

- Password (Muss Feld, Übereinstimmung mit Confirmation und muss je ein Gross-, Kleinbuchstaben und eine Nummer beinhalten)
- Password Confirmation (Muss Feld)

The image shows a web form titled "Settings". It contains three input fields: "Username" with the value "philipp.oggier@gmail.com", "Password" (empty), and "Password (re-typing)" (empty). Below the fields is a "save" button.

Abbildung 43: Settings frontend

### 3.2.7 Logout

Das Logout zerstört die Session und leitet den Benutzer auf die Login Seite weiter.

## 4 Technische Errungenschaften und Daten

### 4.1 Datenbank

Als eines der wichtigsten Elemente einer Applikation zählt die Datenbank. Modelliert wurde die Datenbank mit MySQL Workbench<sup>36</sup>. In den nächsten Abschnitten werde ich den Aufbau der Datenbank erläutern. Abbildung 15 zeigt das Model. (Im Anhang befindet sich eine ausklappbare Version)

#### 4.1.1 Model

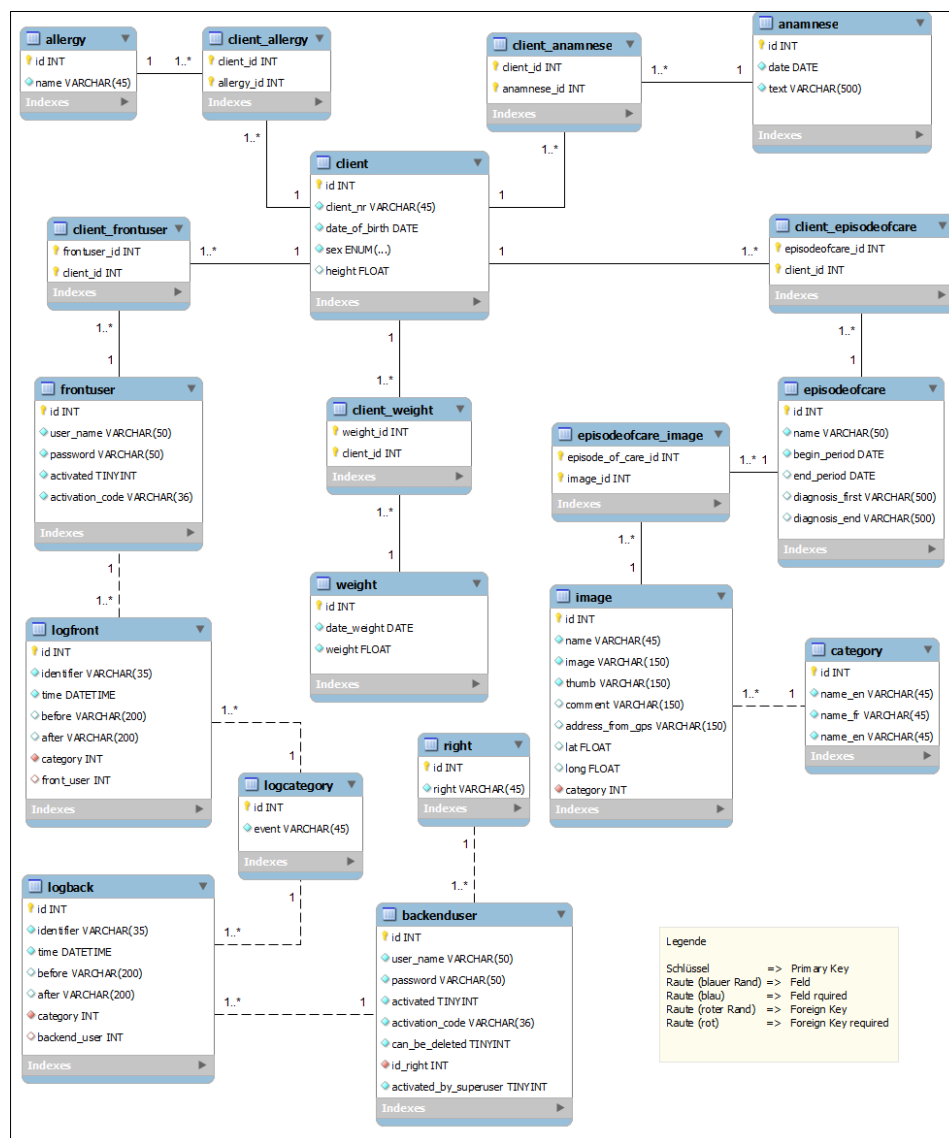


Abbildung 44: Datenbankmodel - BA

<sup>36</sup> <http://www.mysql.de/products/workbench/> (Stand 30.07.2012)

Die Datenbank besteht aus 19 Haupt- und Hilfstabellen. Es gibt im Prinzip zwei Unterteilungen, zuerst die Benutzer (Front- und Backend), sowie auf der anderen Seite die Patienten **client**. Verbunden sind diese zwei Teile über **frontuser** und **client**.

#### **4.1.1.1 Client**

Die Daten für den Patienten setzen sich aus einer Haupttabelle **client** und aus mehreren n:n Beziehungen zusammen. In der Haupttabelle sind Werte gespeichert, welche einmalig pro Patienten sind. Diese sind die Patientennummer, das Geburtsdatum, das Geschlecht und die Grösse. Theoretisch könnte man die Grösse auch in eine weitere Tabelle auslagern, wenn man eine Historie der Grösse haben möchte.

In den Nebentabellen für den Patienten werden Werte gespeichert, welche laufend verändert werden, mehrere Einträge beinhalten, Daten welche für alle zur Verfügung gestellt werden oder welche chronologisch gesammelt werden sollen.

##### **Allergy**

Die Tabelle **allergy** gibt eine Liste der vorhandenen Allergien wieder. Auf diese können alle User zurückgreifen, wenn sie einen Patienten erfassen. Somit ist gewährleistet, dass Daten über Allergien an einer Stelle erfasst und geändert und damit die Fehlerquote minimiert, sowie die Redundanz eliminiert werden kann. Die Verbindung und Zuordnung zu den Patienten erfolgt über **client\_allergy**.

##### **Weight**

In der Tabelle **weight** werden alle erfassten Gewichte von allen Patienten aufgelistet. Diese haben ein Datumsfeld und ein Gewicht. Somit ist eine chronologische Ansicht der Gewichte eines Patienten ersichtlich. Ausserdem wäre eine Statistik über ähnliche Patienten möglich.

Die Verbindung und Zuordnung zu den Patienten erfolgt über **client\_weight**.

##### **Anamnese<sup>37</sup>**

Die Tabelle **anamnese** speichert die Krankengeschichten der Patienten ab. Diese haben ein Datumsfeld und ein Textfeld, in welcher die Krankengeschichte für dieses Datum erfasst werden kann. Durch die Speicherung in dieser Art, ist es möglich, einzelne Daten einfach herauszufiltern oder eine chronologische Ansicht zu erstellen.

Die Verbindung und Zuordnung zu den Patienten erfolgt über **client\_anamnese**.

---

<sup>37</sup> <http://de.wikipedia.org/wiki/Anamnese> (Stand 30.07.2012)

## Episode of Care

Eine Episode of Care ist die Zeit, in welcher der Patient behandelt wird. Die Tabelle **episodeofcare** speichert alle Informationen zu dieser Behandlungsperiode. Sie besteht aus einem Namensfeld **name** dieser Episode, welcher frei vergeben werden kann, einem **begin\_period** und **end\_period**, welche die Behandlungsperiode widerspiegeln, einer ersten Diagnose, die Diagnose bei der Einlieferung **diagnosis\_first** und einer letzten am Ende der Periode **diagnosis\_end**. So können später die Diagnosen auch verglichen und eventuell Abläufe verbessert werden. Die Verbindung und Zuordnung zu den Patienten erfolgt über **client\_episodeofcare**.

## Image

Jede Episode of Care kann mehrere Bilder enthalten. Diese werden in der Tabelle **image** (Daten) und die Bilder selber direkt auf dem Server gespeichert. Diese haben ein Namensfeld **name** des Images, welche nicht mit dem Namen der Bilder übereinstimmen müssen und kann frei vergeben werden. Es existieren zwei ähnliche Felder, das eine enthält den Pfad zum Image **image** und das andere zum Thumbnail **thumb**. Theoretisch würde ein Feld reichen. Aus den Angaben wie Patientenummer, Episode und dem Namen des Bildes könnte man später den Pfad für **image** und **thumb** generieren. Mit der aktuellen Situation ist es aber einfacher unterschiedliche Pfade zu nutzen. Man wird flexibler, sollte man die Ordnerstruktur später ändern wollen. Um das Bild zu beschreiben, haben wir ein Kommentarfeld **comment** vorgesehen, welches uns einen Text bis zu einer Länge von 150 Zeichen gestattet. Es ist ausserdem möglich, Geolocation-Daten für ein Bild zu speichern. Diese setzen sich aus drei Feldern zusammen. Die Adresse welche von Google ermittelt wird **address\_from\_gps**, die Latitude **lat** und die Longitude **long**. Ausserdem hat **image** ein Feld für den Foreign Key für die Tabelle **category**.

Die Verbindung und Zuordnung zu den Episodes of Care erfolgt über **episodeofcare\_image**.

## Category

In der Tabelle **category** können wir eine Liste mit Kategorien für die Bilder erfassen. Beispiele wären hier Körperregion wie: Unterarm, Korpus etc.

### 4.1.1.2 User

Die User sind in zwei Kategorien geteilt. Auf der einen Seite haben wir Frontuser, z.B. medizinisches Personal und auf der anderen Seite die Backenduser, z.B. das Sekretariat. Ein Doktor oder eine Krankenschwester können aber auch Backenduser sein.



## Frontenduser

Die Frontenduser sind die Benutzer, welche das Tool täglich bei der Arbeit mit dem Patienten einsetzen. Dies können Krankenschwestern, Ärzte und andere medizinische Fachkräfte sein. Die Tabelle **frontuser** setzt sich aus einem **user\_name**, welcher einmalig ist, einem **password**, welches als MD5 Hash gespeichert ist, **activated**, welches einen **tinyint** speichert, dieser Wert dient als **boolean**, sprich **TRUE** oder **FALSE** und dem **activation\_code** zusammen. Die Verbindung und Zuordnung zu den Patienten erfolgt über **client\_frontuser**. Ausserdem werden die Frontuser für das Logbuch **logfront** benutzt.

## Backenduser

Der Teil Backend ist für verwaltende Benutzer gedacht. Diese können zum Beispiel Sekretäre und Sekretärinnen, aber auch medizinisches Fachpersonal sein.

Die Tabelle **backenduser** setzt sich ähnlich zusammen wie die **frontuser**, hat aber drei Felder mehr. Diese setzt sich aus einem **user\_name**, welcher einmalig ist, einem **password**, welches als MD5 Hash gespeichert ist, **activated**, welches einen **tinyint** speichert, dieser Wert dient als **boolean**, sprich **TRUE** oder **FALSE** und dem **activation\_code** zusammen. Zusätzlich hat diese Tabelle noch den Eintrag **can\_be\_deleted**, dieses Feld hat auch einen Typen **tinyint** für den zu speichernden **boolean** Wert. Im Moment hat dieses Feld aber keine Bedeutung und ist für spätere Erweiterungen oder Ergänzungen vorgesehen. Das Feld **id\_right** ist der Fremdschlüssel zu der Tabelle **right** und **activated\_by\_superuser** enthält einen **boolean**, welcher widerspiegelt, ob ein Superuser den **backenduser** freigegeben hat. Verbunden ist diese Tabelle mit **right** und mit dem Logbuch **logback**.

## Right

Die Tabelle **right** enthält ein Feld für die Rechte, zurzeit existieren zwei:

- Superuser,
- Administration.

Der Superuser kann zusätzlich zu den Rechten des Administrators alle Front- und Backenduser verwalten und Patienten den Frontusern zuordnen.

## Logfront/Logback

Die Tabelle **logfront** und **logback** sind gleich aufgebaut. Sie sind aber trotzdem getrennt, damit man später einfacher Ergänzungen anbringen kann. Im Moment ist eine Vermischung der Logbücher nicht möglich.

Die Tabellen setzt sich aus einem **identifizier**, welcher die veränderte Tabelle festhält, **time**, welches automatisch das aktuelle Datum und die Zeit im Format **h:m:s** abspeichert. Die Felder **before** und **after** speichern alle Daten vor und nach der Veränderung ab. Bei einem Insert wird im **before** Feld, bei einem Delete im **after** Feld nichts gespeichert. Durch diese Speicherung kann später alles nachvollzogen werden.

Verbunden ist diese Tabelle mit **frontuser**, **backenduser** und mit der Kategorie-Tabelle **logcategory**.

### Logcategory

Die Tabelle **logcategory** enthält ein Feld für Events, zurzeit existieren deren drei:

- Delete,
- Update,
- Insert.

## 4.2 CodeIgniter

Wie in der Methodik bereits erwähnt, habe ich mich für das CodeIgniter Framework entschieden. Immer noch viele Entwickler programmieren PHP von Grund auf und haben ihre eigenen Libraries, welche sie einsetzen. Dies hat den Vorteil, dass man selber für die Libraries zuständig ist und somit alles selber im Griff hat. Man weiss genau, wie die Klassen aufgebaut und genutzt werden. Dies ist aber in der heutigen, schnelllebigen Zeit fast nicht mehr möglich. Frameworks sind getestet, werden gewartet und Fehler durch die Community schneller aufgedeckt. Eine aktive Community bereichert ausserdem das Framework indem laufend neue Libraries beigesteuert werden. Warum sollte man das Rad neu erfinden? Dies würde nur Zeit und Geld kosten.

### 4.2.1 Was hat CodeIgniter zum Projekt beigetragen?

CodeIgniter hat mir einen Rahmen geboten, mit dem man objektorientiert und nach dem MVC Muster PHP Seiten erstellen kann. Wie fast überall im Leben gilt der Spruch:

*„easy to learn, hard to master“, Blizzard Entertainment.*

Ein weiterer Vorteil ist, dass die einzelnen Module des Frameworks bereits getestet und tausendfach im Einsatz sind. In den weiteren Punkten wird die Verwendung des Frameworks noch besser und vertiefter erläutert.

#### 4.2.1.1 Aufbau

Der Aufbau von CodeIgniter ist recht einfach. Als erstes fällt auf, dass die Applikation und das System voneinander getrennt sind. Alles was CodeIgniter von Haus aus zur Verfügung stellt und welche applikations unabhängig sind, findet man im Systemordner wieder. Durch die strikte Trennung wird die Fehlerquote minimiert und das System bleibt „sauber“. Somit kann dieses für andere Projekte/Applikationen mitverwendet werden.

Alle Ergänzungen, Drittbibliotheken und die Konfigurationsdateien befinden sich im Applikationsbereich. Externe Libraries, sowie auch eigene sind einfach zu implementieren, was die Flexibilität enorm erhöht.

#### 4.2.1.2 Beispiel

##### Controller

```
class Register extends CI_Controller {

    function __construct() {
        parent::__construct();
    }
    function index() {
        $this->register();
    }
    function register() {

        $this->load->library('form_validation');
        $this->load->library('Ashirra_Random_string');
        $this->load->helper('recaptcha_helper');
        $this->config->load('recaptcha');

        .....other code.....

        $this->load->view('inc/login_header');
        $this->load->view('login/registration_view_back', $data);
        $this->load->view('inc/login_footer');

        .....other code.....

    }
}
```

Dieses Beispiel ist dem Registrierungsformular des Backends entnommen. Jeder Controller erweitert den **CI\_Controller**.

### Steuerung URL

Stamm URL: `oggierp.pre-view.ch/magicpictures`

URL Register: `oggierp.pre-view.ch/magicpictures/index.php/loginfront/register`

Loginfront ist ein Ordner und der Controller `register` befindet sich in diesem. Es wird solange gesucht bis der Controller, sprich die PHP Datei gefunden worden ist.

Beispiel: `../register/check/1/2/3`

Mit der oben erwähnten Variante, würde das System den Controller bei `register` finden. Der nächste Parameter steht für eine `function()` im Controller, hier `check()`. Alle folgenden Parameter stehen für die Variablen dieser Funktion.

### Load

Will man configuration files, Libraries oder Helper benutzen, können diese direkt in den Controller geladen werden.

```
$this->load->library('form_validation');
```

Mit dem Laden dieser Library wird zugleich die eigene Extension der `form_validation` mitgeladen. Sollte es noch keine Library geben mit dem Pattern `prefix_ci_library` muss diese Library separat geladen werden.

```
$this->load->library('Ashirra_Random_string');
```

Unten sehen wir ein Anwendungsbeispiel, welche die `form_validation` benutzt.

```
$this->form_validation->set_rules  
('password1','Password','required|matches[passwrd2]|password_check[1,1,1]');
```

### View

Im Controller wird die View über einen einfachen Befehl geladen.

```
$this->load->view('inc/login_header');  
$this->load->view('login/registration_view_back',$data);  
$this->load->view('inc/login_footer');
```

Hierzu wird im View-Ordner `inc/login_header` gesucht. Es ist zu erwähnen, dass `inc` ein Ordner ist und `login_header` die eigentliche View, sprich PHP Datei. Somit können verschieden tiefe Ordnerstrukturen aufgebaut werden, was die Verwaltung sehr erleichtern kann.

## 4.2.2 Ergänzungen

Alle Ergänzungen, welche ich selber geschrieben habe, fangen mit dem Präfix `Ashirra_` an. Damit der Controller weiss, welche Dateien Erweiterungen sind, wird dies in der Konfigurationsdatei angegeben.

```

/*
-----
Class Extension Prefix
-----

This item allows you to set the filename/classname prefix when extending
native libraries. For more information please see the user guide:

http://codeigniter.com/user\_guide/general/core\_classes.html
http://codeigniter.com/user\_guide/general/creating\_libraries.html
*/
$config['subclass_prefix'] = 'Ashirra_';

```

Abbildung 45: class extension prefix - CodeIgniter – BA

#### 4.2.2.1 Validation

Zur Überprüfung der Form-Felder kommt **form\_validation** von CodeIgniter zur Anwendung. Dieser wurde von mir ergänzt und erweitert.

##### Eigene Features:

- **password\_check**: Diese Erweiterung kontrolliert, ob das Passwort Gross-, Kleinbuchstaben und/oder Nummern beinhaltet **password\_check[1,1,1]**,
- **min\_value**: Überprüft, ob ein Wert grösser oder gleich wie der eingegebene Wert ist.
- **max\_value**: Überprüft, ob ein Wert kleiner oder gleich wie der eingegebene Wert ist.
- **decimal**: Überprüft, ob die eingegebene Zahl eine Dezimalzahl ist.
- **decimalPlaces**: Überprüft, ob die eingegebene Zahl eine Dezimalzahl ist. Und dazu noch die gewünschte Kommastelle. 10.309kg entsprechen der Validation **decimalPlaces[3]**.
- **client**: Kontrolliert, ob der Client dem regex<sup>38</sup> Muster entspricht **'/^P{1}A{1}[0-9]{4}\$/'**, was den Benutzer zwingt eine Eingabe wie PA1010 zu machen.
- **check\_end\_date**: Checkt, ob das Startdatum gleich oder weiter zurück in der Vergangenheit liegt, als das Enddatum.
- **check\_captcha**: Kontrolliert mit einem Callback, ob die eingegebenen Zeichen übereinstimmen.

<sup>38</sup> [http://de.wikipedia.org/wiki/Regul%C3%A4rer\\_Ausdruck](http://de.wikipedia.org/wiki/Regul%C3%A4rer_Ausdruck) (Stand 26.07.2012)

Beispiele für die Validation im Controller:

```
$this->load->library('form_validation');
$this->form_validation->set_rules('user_name','Username', 'required|min_length[6]|is_unique[backenduser.user_name]|valid_email');
$this->form_validation->set_rules('password1','Password', 'required|matches[password2]|password_check[1,1,1]');
$this->form_validation->set_rules('password2','Password Confirmation', 'required');
$this->form_validation->set_rules('recaptcha_response_field', 'reCaptcha', 'required|check_captcha');
```

Abbildung 46: form\_validation - CodeIgniter – BA

Erweiterung der **form\_validation**

```
<?php

class Ashirra_Form_validation extends CI_Form_validation
{
    /**
     * Checks if a value is greather equal the min_value
     *
     * @param type $value
     * @param type $min
     * @return boolean
     */
    function min_value($value,$min){

        $CI =& get_instance();
        $CI->form_validation->set_message('min_value',
            'The %s must be at least '.$min.'!');

        if($value>=$min)
            return true;
        return false;
    }
}
```

Abbildung 47: ashirra\_form\_validation - CodeIgniter - BA

#### 4.2.2.2 Mail / Activation

CodeIgniter unterstützt das Versenden von Mails. Ich habe eine eigene Klasse erstellt, um die Mailklasse von CodeIgniter an meine Wünsche anzupassen. Diese Mails brauche ich für folgendes:

Nach einer erfolgreichen Registrierung erhält der User ein Mail. Nach der Art eines Double Opt-in<sup>39</sup> kann er den Account aktivieren oder direkt löschen. Gelöscht werden kann ein Account nur, wenn er nicht aktiviert ist. Ansonsten kann dies nur über die Settings oder einen Superuser geschehen.

Mit einer Library Erweiterung **Ashirra\_Random\_string**, welche von mir implementiert wurde, wird ein 36-stelliger String erstellt. Dieser dient als activation code Generator.

<sup>39</sup> [http://de.wikipedia.org/wiki/Double\\_Opt-in#Double\\_Opt-in](http://de.wikipedia.org/wiki/Double_Opt-in#Double_Opt-in) (Stand 30.07.2012)

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');
class Ashirra_Random_string{

    private $chars;

    private $str;

    public function __construct(){

        $this->chars = array("a","b","c","d","e","f","g","h","i","j","k","l","m","n",
            "o","p","q","r","s","t","u","v","w","x","y","z","A",
            "B","C","D","E","F","G","H","I","J","K","L","M","N",
            "O","P","Q","R","S","T","U","V","W","X","Y","Z","0",
            "1","2","3","4","5","6","7","8","9","!","");

        $this->str = "";

    }

    public function rnd_string($length="36"){

        $this->str = "";
        for ($index = 0; $index < $length; $index++) {
            $this->str .= $this->chars[mt_rand(0, count($this->chars)-1)];
        }
        return $this->str;
    }

}
?>
```

Abbildung 48: ashirra\_rnd\_string - CodeIgniter - BA

Aufruf im Controller

```
$this->load->library('Ashirra_Random_string');
$data = array(
    'user name' => $ POST['user name'],
    'password' => md5($_POST['password1']),
    'activation_code' => $this->ashirra_random_string->rnd_string(),
);
```

#### 4.2.2.3 HTACCESS<sup>40</sup>

Durch die Ergänzung des .htaccess kann der Zugriff auf Ordner und das Neuschreiben von URLs gesteuert werden.

```
DirectoryIndex index.php
RewriteEngine on
RewriteCond $1 !^(index\.php|clientdata|images|assets|robots\.txt|favicon\.ico)
#RewriteCond %{REQUEST_FILENAME} !-f
#RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ ./index.php/$1 [L,QSA]
```

Abbildung 49: htaccess - CodeIgniter – BA

#### 4.2.2.4 Fake Klasse

Da die Applikation und das System getrennt voneinander aufgebaut sind, funktioniert die Autovervollständigung nicht wie gewohnt. Um dieses Defizit zu beheben, habe ich bei den eigenen Controllern einen **fake\_controller** erstellt.

Durch diese einfache Ergänzung kann in jedem Controller mit **ctrl + space** die Autovervollständigung gestartet werden.

<sup>40</sup> <http://de.wikipedia.org/wiki/Htaccess> (Stand 27.07.2012)

```
<?php
class CI_Controller {
    /**
     *
     * @var CI_DB_active_record
     */
    public $db;

    /**
     *
     * @var CI_Loader
     */
    public $load;
```

Abbildung 50: Beispiel fake\_controller - CodeIgniter - BA

#### 4.2.2.5 Sicherheit

In der heutigen Zeit mit den ganzen „Zombie-Computern“, Bots und anderen Missbräuchen im Internet ist der Sicherheitsaspekt nicht zu vernachlässigen.

Herr Dr. Henning Müller und ich haben am Anfang besprochen, wie wichtig dieser Teil sein soll. Es wurde entschieden, dass Sicherheitsaspekte integriert werden sollen, aber [https](#) für diesen Prototyp nicht implementiert werden soll.

In den nachfolgenden Zeilen werde ich auf die getroffenen Sicherheitsaspekte eingehen.

#### Captcha<sup>41</sup>

Bei der Registrierung wurde die Captcha-Funktion von Google, reCaptcha integriert. Diese ist dazu da, das Leben der Maschinen zu erschweren und zwischen Menschen und diesen zu unterscheiden. Damit wird eine Überflutung von Registrationen, ausgelöst durch Maschinen, vermieden.



Abbildung 51: Beispiel Captcha - prototype - BA

<sup>41</sup> <http://de.wikipedia.org/wiki/CAPTCHA> (Stand 27.07.2012)



## Rechte

Im Backend kommen zwei Rechte zum Einsatz:

- Superuser,
- Administration.

Der Superuser hat (wie meistens) alle Rechte, im Gegensatz zur Administration kann er Frontuser, Backenduser und die Verbindungen zwischen Frontuser und Patienten erstellen und verwalten.

## Login

Front- sowie Backend sind mit einem Login gesichert. Mit der Validation `callback_check_database_backuser` wird das Feld `password` an eine Methode übergeben. Diese Methode ruft den `user_name` von den `$_POST` Variablen ab und generiert eine Abfrage in der Datenbank. Bei erfolgreicher Anmeldung am System werden Session-Variablen erstellt. Diese werden später bei jeder Seite zum Einsatz kommen, diese werden untenstehend erwähnt:

- `portal` zeigt an, um welches System es sich handelt (Back oder Front),
- `id` speichert die ID des Benutzers,
- `user_name` speichert den Namen des Benutzers,
- `right` steht für das Recht.

In jedem Controller, der vor fremden Zugriffen geschützt werden muss, kommen diverse Abfragen zum Einsatz.

Unten aufgeführt ist ein Beispiel, welches die Rechte nicht überprüfen muss.

```
if($this->session->userdata('logged_in_back') )
{
    if($this->session->userdata['portal']=='back'){

        //code

    }
    else {
        //If portal ist not back, redirect to login page
        redirect('loginback/login', 'refresh');
    }
}
else
{
    //If no session, redirect to login page
    redirect('loginback/login', 'refresh');
}
```

Wer also versucht sich mit einer validen URL ins System zu schmuggeln, ohne sich zuvor angemeldet zu haben, wird auf die Login Seite verwiesen.

## Verschlüsselung wichtiger Daten

Wichtige Daten, in diesem Prototyp vor allem das Passwort, werden mit **MD5 ()** in einen Hash verwandelt und so in der Datenbank gespeichert.

Ein Passwort mit dem Wert **Hello1** wird somit zu einem MD5 Hash mit einem Wert von **7a6d1b13498fb5b3085b2fd887933575**.

Sollten kriminelle Organe in die Datenbank einbrechen, sehen diese nur den MD5 Hash und können damit in erster Linie nichts anfangen.

In der Schule haben wir gelernt, dass es keine vollkommene Sicherheit gibt. Denn einen MD5 Hash kann mit Aufwand entschlüsselt werden. Dies geschieht mit sogenannten Regenbogentabellen.<sup>42</sup>

Je nach eingesetztem Hash Typ und der Länge des Kennwortes steigt der Aufwand den Hash wieder in die ursprüngliche Form zu verwandeln enorm an.

Eine weitere Gegenmassnahme ist das Beistreuen von einem Zusatz, auch „Salt“<sup>43</sup> genannt. Dieser Zusatz, im besten Fall ein Zufallswert, wird vor dem eigentlich Passwort dazugegeben und in der Datenbank mitgespeichert. Um den Hash nun zu knacken, müssen beide Werte ermittelt werden, was Regenbogentabellen unwirtschaftlich macht. Trotzdem schützt dieses „Salt“ nicht vor schlechten Passwörtern, welche per „Brute Force“ ermittelt werden können. Beispiele von schlechten Passwörtern finden wir im nächsten Kapitel „Validation“.

## Validation

Wie oben erwähnt, schützt das beste System nicht, wenn die Passwörter schlecht gewählt sind. Passwörter wie:

- adminadmin,
- hello,
- liebe / love,
- gott / god,

werden oft gebraucht, aber sind nicht sicher. Auch Textpassagen aus Büchern sind nicht geeignet, da Wörter in Datenbanken gespeichert werden können. Gute Passwörter setzen sich aus mehreren unterschiedlichen Kriterien zusammen:

- schlecht zu erraten,
- je länger desto sicherer (min. 8 Zeichen),
- Mix von Grossbuchstaben, Kleinbuchstaben, Zahlen und Sonderzeichen.

---

<sup>42</sup> [http://de.wikipedia.org/wiki/Rainbow\\_Table](http://de.wikipedia.org/wiki/Rainbow_Table) (Stand 27.07.2012)

<sup>43</sup> [http://de.wikipedia.org/wiki/Salt\\_%28Kryptologie%29](http://de.wikipedia.org/wiki/Salt_%28Kryptologie%29) (Stand 27.07.2012)

Mit der Validation wird der Benutzer an die Hand genommen und er wird so gezwungen das Passwort einigermaßen sicherer zu machen.

Zum Beispiel kann mit der Funktion `password_check[1,1,1]` der Benutzer gezwungen werden, mindestens einen Gross-, Kleinbuchstaben und eine Nummer im Passwort zu gebrauchen. Solche Funktionen können später auch mit regulären Ausdrücken und mit einer minimalen Anzahl von Zeichen ergänzt werden, damit gewisse Reihenfolgen der Zeichen nicht möglich sind oder zwingendermassen vorkommen müssen.

#### XSS Filter<sup>44</sup>

CodeIgniter liefert uns schon eine hausgemachte Lösung, um gegen das Cross Site Scripting vorzugehen. Damit diese Funktion alle `$_POST`, `$_GET` und Cookie-Daten überprüft, müssen wir eine einzige Einstellung in den Konfigurationsdateien vornehmen.

```
$config['global_xss_filtering'] = TRUE;
```

#### CSRF<sup>45</sup>

CodeIgniter liefert auch hier eine Lösung gegen Cross Site Request Forgery. Dieses Problem tritt bei HTTP Anwendungen auf, da der Browser aufgrund der Statuslosigkeit die Sitzungsdaten jedes Mal an den Server sendet.

Mit einer einfachen Änderung in der Konfigurationsdatei könnten wir uns vor dieser Art von Übergriffen schützen.

```
$config['csrf_protection'] = TRUE;  
$config['csrf_token_name'] = 'csrf_test_name';  
$config['csrf_cookie_name'] = 'csrf_cookie_name';  
$config['csrf_expire'] = 7200;
```

Wir müssten nur den Wert in der ersten Zeile von `FALSE` auf `TRUE` setzen.

Da wir im Wizard laufend `$_POST` Arrays hin- und herschicken, erkennt das System dies als CSRF und wir erhalten einen Fehler. Darum wurde dieses Feature nicht aktiviert.

---

<sup>44</sup> [http://codeigniter.com/user\\_guide/libraries/security.html](http://codeigniter.com/user_guide/libraries/security.html) (Stand 27.07.2012)

<sup>45</sup> [http://de.wikipedia.org/wiki/Cross-Site\\_Request\\_Forgery](http://de.wikipedia.org/wiki/Cross-Site_Request_Forgery) (Stand 27.07.2012)

## 4.3 gasORM

CodeIgniter stellt uns in Sachen Models nur grundlegende Möglichkeiten zur Verfügung. Update, Delete und Insert Queries müssen selber erstellt werden. Theoretisch müsste man für jedes Model ein CRUD erstellen und dieses auch testen. Da dies einen Mehraufwand bedeutet und ausserdem die Fehlerquellen stark erhöhen würde, habe ich mich für gasORM entschieden.

Zugute kommt uns, dass gasORM für CodeIgniter entwickelt wurde. Leider existiert keine API-Dokumentation, sondern nur Beispiele.

### 4.3.1 Erstellung

Die Grundeinstellungen können in einer Konfigurationsdatei vorgenommen werden. Diese befindet sich im Ordner `config`. Um alle Einstellungen nutzen zu können, müssen wir auch Anpassungen in der Konfigurationsdatei von CodeIgniter vornehmen. Mit den Einstellungen haben wir folgende Möglichkeiten:

```
$config['auto_create_tables'] = FALSE;
$config['auto_create_models'] = FALSE;
$config['cache_request'] = TRUE;
$config['models_path'] = array('Model' => APPPATH.'models');
```

gasORM besitzt die Fähigkeit direkt von der Datenbank Modelle oder über den umgekehrten Weg, die Datenbanktabellen von den Modellen generieren zu lassen.

Da ich die Datenbank mit MySQL Workbench bereits modelliert und über phpMyAdmin integriert hatte, habe ich die Funktion `$config['auto_create_models'] = TRUE;` benutzt. Damit diese Funktion wahrgenommen werden kann, muss die Migrations Library von CodeIgniter eingeschaltet werden. Nach dem Erstellen der Modelle sollte dies wieder rückgängig gemacht werden.

### 4.3.2 Aufbau des Models

```
namespace Model;
/* This basic model has been auto-generated by the gasORM */
use \Gas\Core;
use \Gas\ORM;

class Frontuser extends ORM {

    public $primary_key = 'id';

    function _init()
    {
        self::$relationships = array (
            'client' => ORM::has_many
                ('\\Model\\Client\\Frontuser => \\Model\\Client'),
            'logfront' => ORM::has_many('\\Model\\LogFront'),
        );
        self::$fields = array(
```

```

'id' => ORM::field('auto[11]'),
'user_name' => ORM::field('char[50]'),
'password' => ORM::field('char[50]'),
'activated' => ORM::field('numeric[4]'),
'activation_code' => ORM::field('char[36]'),
);
}
}

```

Der Aufbau des Models ist ähnlich wie bei anderen Programmiersprachen. Automatisch werden nur alle `ORM::field()` mit den dazugehörigen Werten erstellt. Der Primary Key ist standardmässig auf den Wert `id` festgelegt, dies kann später geändert werden. Alle Verbindungen müssen manuell erstellt werden. Hierzu haben wir drei Verbindungsmöglichkeiten:

- `has_one`,
- `has_many`,
- `belongs_to`.

Bei einer n:n Beziehung müssen ausserdem Hilfstabellen erstellt werden. Eine Hilfstabelle mit dem Namen `client_frontuser` wird in der Ordnerstruktur folgendermassen abgelegt.

```
../models/client/frontuser.php
```

Somit können ganze Ordnerstrukturen anhand des Tabellennamens erstellt werden. Diese Verbindungen sollten aber so einfach wie möglich gehalten werden, damit man den Überblick nicht verliert.

### 4.3.3 Einsatz im Controller

Durch den Einsatz des Namenraums `namespace Model\[Ordner];` oder `namespace Model;` ist der Einsatz im Controller einfacher und übersichtlicher. Durch die gute Integration in CodeIgniter können auch die meisten Queries vom Framework gebraucht werden. Die Rückgabewerte sind alles Objekte.

```

$logback = Model\LogBack::find($id);
$client = Model\Client::where_in('id',$client_frontuserArray)->all();
$result = Model\Client\Anamnese::make()->client_anamnese($clientid);

```

Oben aufgeführt sind ein paar Beispiele wie gasORM gebraucht werden kann. Beim ersten Beispiel wird `find()` eingesetzt, standardmässig können wir nach einem Primary Key suchen. Mit der Erweiterung dazu `find_by_[column]` können wir auch andere Spalten durchsuchen lassen.

Bei der zweiten Abfrage kommt die Funktion `where_in` zum Einsatz. Diese durchsucht eine Spalte, hier `id`, nach den Daten in einem Array `client_anamnese`, dies entspricht einer SQL Abfrage `SELECT * FROM Client WHERE id in (1,3,5)`.

Es ist auch möglich eigene, spezielle Abfragen zu kreieren. Diese Abfragen werden im Model direkt erstellt und können mit der Funktion `make()` gebunden werden. Im obigen Beispiel ist `client_anamnese` der Name der `function()` im Model.

Es ist auch möglich Abfragen zu verketteten.

```
$logback = Model\Logback::limit(5)->order_by('time','desc')->all();
```

Weitere `finder()`:

- `first()` und `last()`
- `max()`, `min()`, `sum()` und `avg()`

## 4.4 groceryCRUD

Grundsätzlich unterstützt CodeIgniter die Funktion zur Tabellenerstellung und der Paginierung. Diese ist aber sehr rudimentär und hat einige Probleme aufgeworfen. Vor allem in der Zusammenarbeit beider Funktionen. Ich habe mich dazu entschlossen, eine Library zu finden, welche mir Vorteile bringt bei:

- der Tabellenerstellung,
- einem integrierten CRUD,
- dem Design.

Ich wurde fündig mit groceryCRUD. Dieses Projekt wird im Moment von einer einzigen Person verwaltet. Darum ist diese Library eher nicht für kommerzielle Projekte geeignet. Trotzdem habe ich die Chance wahrgenommen, diese Library in meiner BA zu testen.

Auch hier gilt, man erhält schnell gute Resultate. Werden aber spezielle Funktionen gebraucht, ist man auf sich selbst angewiesen. Es existiert eine kleine Community. Somit ist es von Vorteil gute Programmierkenntnisse zu haben.

Ausserdem funktioniert die Suche in Tabellen mit `search all` nicht, sollte diese Tabelle Kolonnen aufweisen, welche nicht in der Datenbank vorkommen.

### 4.4.1 Controller

Einzelne Tabellen mit CRUD Eigenschaften können mit nur wenigen Zeilen erstellt werden. Mit vier Zeilen kann eine solche Tabelle generiert werden.

```
$this->load->library('grocery_CRUD');
$this->grocery_crud->set_table('allergy');
$output = $this->grocery_crud->render();
$this->load->view('back/allergy',$output);
```

Für den Gebrauch mit 19 Tabellen, welche miteinander verbunden sind, reichen diese vier Zeilen Code aber bei weitem nicht. Nachstehend wird erklärt, was alles möglich ist. So sollte auch klar werden, warum vier Zeilen für meine Applikation nicht ausreichen.

#### 4.4.1.1 Columns

Nur Kolonnen in einer Tabelle anzuzeigen, welche in der Datenbank selber vorkommen, reicht in manchen Fällen nicht aus. Dazu stellt uns groceryCRUD eine einfache Möglichkeit zur Verfügung.

```
$this->grocery_crud->columns('client','sex','Weight');
```

In diesem Fall ist `'Weight'` eine Kolonne, welche so in der Tabelle nicht existiert.

#### 4.4.1.2 Relations

Im Prinzip kann man in groceryCRUD sehr einfach Relations setzen. In meiner Arbeit kommen diese Relations aber nur in vier Tabellen zum Einsatz.

- `client` <=> `allergy`
- `logback` <=> `logcategory`
- `logfront` <=> `logcategory`
- `image` <=> `category`

Dies hat vor allem den Grund, dass meine Arbeit aus vielen n:n Beziehungen besteht. Diese werden durch die Rechte oder zugeordnete `clients` <=> `frontuser` eingeschränkt. Weil mit groceryCRUD eine Abfrage über mehrere Tabellen noch nicht möglich ist, kommt bei mir groceryCRUD zusammen mit gasORM zum Einsatz. Ohne Restriktionen funktionieren die Verbindungen aber ordentlich. Bei der Tabelle `allergy`, welche von allen uneingeschränkt benutzt werden kann, sieht dies so aus:

```
$this->grocery_crud->
set_relation_n_n('Allergy','client_allergy','allergy','client_id','allergy_
id','name');
```

Für einfache Relationen 1:n oder 1:1 sieht der Code wie folgt aus:

```
$this->grocery_crud->set_relation('category','logcategory','event');
```

#### 4.4.1.3 Change Header

Normalerweise wird die Tabellenbezeichnung aus dem Namen der Spalte (DB) generiert. Dies ist aber nicht immer schön und praktisch. GroceryCRUD stellt uns hier eine elegante Möglichkeit zur Verfügung.

```
$this->grocery_crud->display_as('date_of_birth','Birthday');
```

Im oben genannten Beispiel ändern wir `date_of_birth`, welches der Name und zugleich auch der Kolonnenbezeichner in der DB ist, in „Birthday“ um.

#### 4.4.1.4 Field type

Normalerweise werden die Feldtypen direkt von der Datenbank, sprich dem Model übernommen. Ab und zu kann es aber vorkommen, dass man den Feldtypen manuell ändern muss oder will. Dies

kommt vor allem bei eigenen Kolonnen zum Einsatz. Oder auch bei Feldern, welche nicht ersichtlich sein sollen.

```
$this->grocery_crud->change_field_type('activation_code', 'invisible');
```

#### 4.4.1.5 Fields

GroceryCRUD ist so aufgebaut, dass die Columns, welche geladen werden grundsätzlich bei der Add-Funktion und der Edit-Funktion schon vorhanden sind. Sie können aber angepasst werden.

```
$this->grocery_crud->add_fields  
('user_name', 'activation_code', 'password', 'activated_by_superuser');
```

Diese Felder sind später im `$_POST` Array vorhanden und können weiter verarbeitet werden. Sollen Felder im Array enthalten sein, werden aber vom System berechnet oder sollen einfach vom Benutzer nicht gesehen werden, können diese auf `invisible` gestellt werden, wie im Beispiel Field type gezeigt wurde.

Dasselbe ist auch mit den Edit-Feldern möglich.

```
$this->grocery_crud->edit_fields('date_of_birth', 'sex', 'height', 'Allergy');
```

#### 4.4.1.6 Callbacks<sup>46</sup>

Sehr wichtige Funktionen sind die sogenannten Callbacks oder auf Deutsch Rückruffunktionen. GroceryCRUD stellt uns diverse Möglichkeiten für solche Rückruffunktionen zur Verfügung.

##### Felder

Wir haben die Möglichkeit Felder zu verändern, sei dies beim Laden der Tabelle,

```
..callback_column('log', array($this, 'linkToLog'));
```

beim Aufruf der Edit-Funktion,

```
..callback_edit_field('client', array($this, 'add_client'));
```

oder der Add-Funktion.

```
..callback_add_field('client', array($this, 'add_client'));
```

Mit diesen Aufrufen können wir durch eine Funktion das Feld ändern lassen, wie wir möchten.

```
function linkToLog($value, $row) {  
    return "<a href='".site_url('back/logback/index/'. $row->id). "'>log</a>";  
}
```

##### CRUD

Wir haben die Möglichkeit vor oder nach einer CRUD-Funktion einen Callback ausführen zu lassen. Diese Möglichkeiten kommen vor allem bei den Logbüchern zum Einsatz, aber auch zum Generieren

<sup>46</sup> <http://de.wikipedia.org/wiki/R%C3%BCckrufffunktion> (Stand 30.07.2012)



eines Aktivierungscode, versenden der Mails oder zum Verschlüsseln des Passwortes mit einem MD5 Hash.

```
..callback_before_delete(array($this,'before_delete'));
```

Dieser Rückruf führt dann vor einem Delete zum Aufruf der folgenden Funktion:

```
function before_delete($primary_key) {
    $this->get_before($primary_key);
    $this->write_log(1);
}
```

#### 4.4.1.7 Validation

Für die Validation können wieder die Grundfunktion von CodeIgniter genutzt werden. Damit die eigene Erweiterung **Ashirra\_Form\_validation** funktioniert, müssen wir bei groceryCRUD folgendes ändern.

```
class grocery_CRUD_Form_validation extends Ashirra_Form_validation{
```

Damit die Hierarchie stimmt, leiten wir die eigene Klasse von CodeIgniter ab und die groceryCRUD Klasse von der eigenen.

#### 4.4.2 View

GroceryCRUD stellt uns direkt zwei Themen für die Tabellen zur Verfügung **datatables** und **flexigrid**. Mit den folgenden Funktionen rendert groceryCRUD die Tabelle vor und übergibt in einer Variablen **\$output** alle benötigten **css\_files** und **js\_files**.

```
$output = $this->grocery_crud->render();

$this->load->view('inc/header_backend',$output);

$this->load->view('inc/nav_back');

$this->load->view('back/client',$output);

$this->load->view('inc/footer');
```

Der Aufruf der Views erfolgt wieder wie gewohnt im Stile von CodeIgniter.

Time	Identifier	Category	Backend User	Detail	Actions
2012-07-30 - 10:25:22	relation	Insert	phillip.oggier@gmail.com	<a href="#">detail</a>	
2012-07-27 - 15:45:57	relation	Insert	phillip.oggier@gmail.com	<a href="#">detail</a>	
2012-07-27 - 15:33:45	frontuser	Insert	phillip.oggier@gmail.com	<a href="#">detail</a>	

Abbildung 52: groceryCRUD View - prototype - BA

## 4.4.3 Queries

Abfragen können nur über die gesetzte Tabelle gemacht werden. Die Abfragen sind eingeschränkt und es werden nicht alle Funktionen von Codelgniter unterstützt.

```
$this->grocery_crud->where('frontuser',$frontuser);  
$this->grocery_crud->set_table('client');
```

Da die **where\_in** und die **or\_where\_in** Funktion nicht unterstützt wurden, habe ich diese selber eingebunden. Dies erforderte aber verschiedene und tiefe Eingriffe in die Library von groceryCRUD. Diese Ergänzung hat sich aber gelohnt. So kann man die Funktionen direkt für groceryCRUD nutzen und muss nicht den Umweg über gasORM oder die Modelle machen.

## 4.5 CSS

Damit eine Software auch bedienbar ist, benötigt man ein Design. Natürlich könnte man alle Programme von einer Kommandozeile aus bedienen, aber für den nicht versierten Benutzer stellt dies eine grosse Hürde dar. Spätestens wenn man eine Software nicht nur für den Eigengebrauch entwickelt, sondern auch verkaufen möchte, ist ein ergonomisches User Interface extrem wichtig. Da das Design in den letzten Jahren immer wichtiger geworden ist, beschäftigen Firmen ganze Design-Abteilungen. Bei vielen Geräten ist das Design inzwischen das „Killerkriterium“, wenn die Funktionen identisch sind. Diese Design-Abteilungen kümmern sich um Ergonomie, Konsistenz, Schriften, Farbmanagement, etc.

Damit ich meine Funktionalitäten präsentieren kann, habe ich mich für ein Design entschieden, das mit Hilfe von CSS3 realisiert wurde. CSS bedeutet „Cascade Style Sheet“.

### 4.5.1 Front

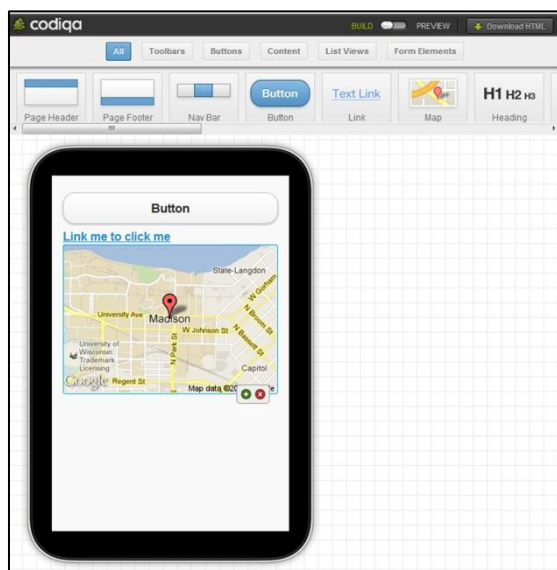


Abbildung 53: jquerymobile - Mobile Builder

Um eine Seite für mobile Geräte zu optimieren, benötigt man entweder viel Zeit und viele „Fallback“-Lösungen oder man verwendet ein Framework, wie zum Beispiel „jQuery Mobile“ (Abb. 53). Dank „jQuery Mobile“ ist es relativ einfach, bestehende HTML-Tags mit CSS für mobile Geräte fit zu trimmen. Die Seite selber wird mit normalem HTML5 erstellt. Erst das „jQuery Mobile“-CSS, in Verbindung mit JScript, ermöglicht es, die Seite auf einem Smartphone entsprechend anzuzeigen und

bedienen zu können. Der grosse Vorteil: Das Framework passt sich an die Grösse des Bildschirms an und ist immer optimal bedienbar.

Um das „jQuery“-CSS ansprechend gestalten zu können, gibt es den „Themeroller“ (Abb. 54). Mit diesem kann man die Farben, Schriftarten und –grössen und andere Elemente an die eigenen Bedürfnisse anpassen. Das Resultat kann über eine URL mit anderen geteilt oder heruntergeladen und im eigenen Projekt weiterverwendet werden.

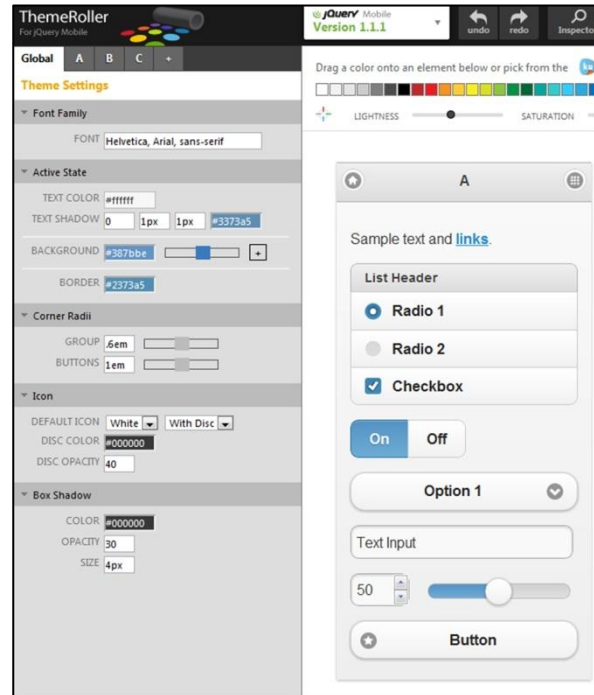


Abbildung 54: jquerymobile - Themeroller

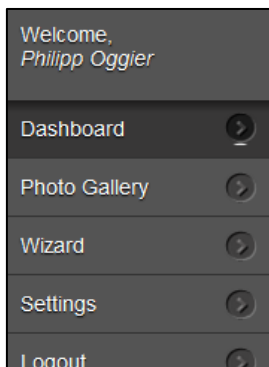


Abbildung 55: Menu frontend

Die Darstellung des Menus (mobile Version) vom sozialen Netzwerk „Facebook“ hat mich für mein Mobilemenu inspiriert. Dieses kann mit einem Wisch heraus- oder eingeklappt werden. Fündig wurde ich bei ALDOMATIC.<sup>47</sup>

<sup>47</sup> <http://blog.aldomatic.com/facebook-style-slide-out-menu-in-jquery-mobile/> Stand 08.08.2012

## 4.5.2 Backend

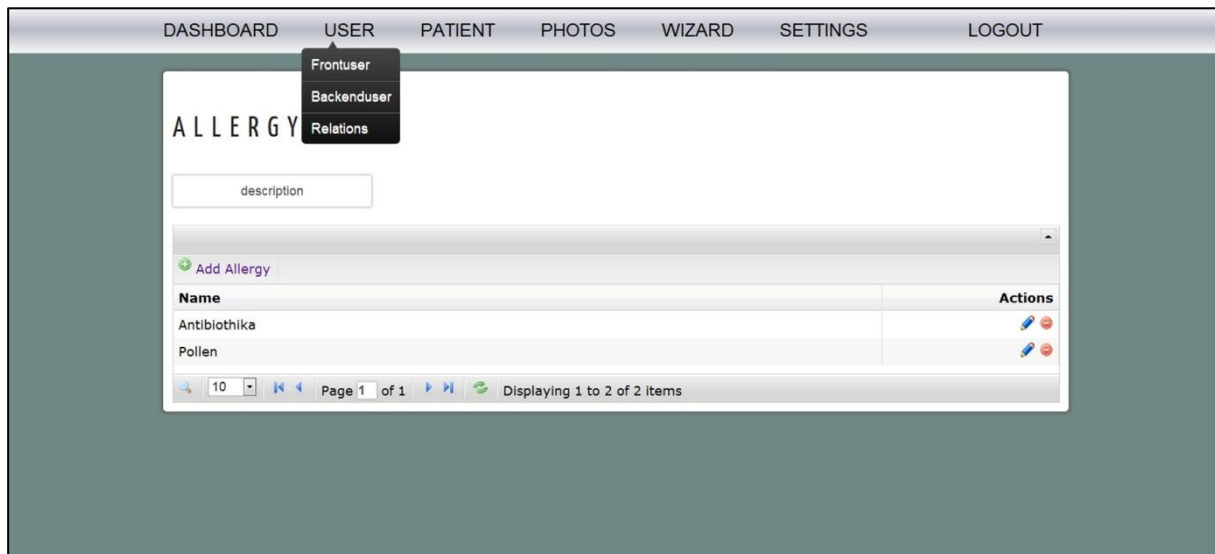


Abbildung 56: Design backend

Das persönliche Ziel für das Backend (Abb. 56) war, dieses schlicht und dennoch modern zu gestalten. So gibt es für die ganze Seite keine Grafiken: Alles wurde soweit wie möglich mit CSS3 realisiert. Dies gilt nicht für die Tabellen, welche von groceryCRUD erstellt werden. Dieses Design ist bereits in der Library integriert.

Die Farben habe ich bewusst gewählt: Sie sollten neutral, der klinischen Umgebung angepasst sein. Das Backend (Administration) sollte im Design schlicht gehalten werden.

Das Menu besteht aus einer ungeordneten Liste `<ul>`, welche mit Listenelementen `<li>` gefüllt ist. Diese werden mit CSS3 formatiert.

Der Menuhintergrund hat einen metallenen Farbverlauf. Diese CSS3-Möglichkeit wird jedoch nicht von allen Browsern unterstützt und wir bieten für alte Browser eine „Fallback“-Lösung. In dieser wird lediglich eine Hintergrundfarbe angezeigt.

```
#navibar {
background: #f5f6f6; /* Old browsers */
background: -moz-linear-gradient(top, #f5f6f6 0%, #dbdce2 21%, #b8bac6 49%, #dddf3 80%, #f5f6f6 100%); /* FF3.6+ */
background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#f5f6f6), color-stop(21%,#dbdce2), color-stop(49%,#b8bac6), color-stop(80%,#dddf3), color-stop(100%,#f5f6f6)); /* Chrome,Safari4+ */
background: -webkit-linear-gradient(top, #f5f6f6 0%,#dbdce2 21%,#b8bac6 49%,#dddf3 80%,#f5f6f6 100%); /* Chrome10+,Safari5.1+ */
background: -o-linear-gradient(top, #f5f6f6 0%,#dbdce2 21%,#b8bac6 49%,#dddf3 80%,#f5f6f6 100%); /* Opera 11.10+ */
background: -ms-linear-gradient(top, #f5f6f6 0%,#dbdce2 21%,#b8bac6 49%,#dddf3 80%,#f5f6f6 100%); /* IE10+ */
background: linear-gradient(to bottom, #f5f6f6 0%,#dbdce2 21%,#b8bac6 49%,#dddf3 80%,#f5f6f6 100%); /* W3C */
filter: progid:DXImageTransform.Microsoft.gradient(
startColorstr='#f5f6f6', endColorstr='#f5f6f6',GradientType=0 ); /* IE6-9 */
}
```

```
width: 100%; top: 0; position: absolute; height: 50px;
```

```
}
```

Erklärung zu den Präfixen der Anbieter.<sup>48</sup>

Tabelle 6: CSS3 / HTML5 - Präfixe der Anbieter

-o-	Opera
-moz-	Mozilla Firefox
-ms-	Microsoft Internet Explorer
-webkit-	Google Chrome, Chromium, Apple Safari

Der Content befindet sich in einem **wrapper**, dieser hat Attribute, die den Inhalt in der Mitte anzeigen lässt.

Als „Special“ habe ich keine Standard-Schriften gewählt. Normalerweise wird, wenn man eine Webseite entwickelt, immer auf gewisse Schriftarten zurückgegriffen, die auf allen Computern installiert sind. Hier habe ich mich für die „Yanone Kaffeesatz“ und die „Droid Sans“ entschieden, wobei erstere für die Übertitel und letztere für die Inhalte zuständig ist.

```
@import url(http://fonts.googleapis.com/css?family=Yanone+Kaffeesatz:400,700);
@import url(http://fonts.googleapis.com/css?family=Droid+Sans);
```

Diese Webschriften werden im CSS mittels @import-Statement eingebunden. Der Vorteil liegt darin, dass die Schriften so von allen Computern angezeigt werden können. Wenn die Schrift-Quelle, in diesem Fall Google, down ist, werden die Schriftarten verwendet, welche im **<font-family>** Tag definiert worden sind (in der Reihenfolge, wie sie angegeben sind).

## 4.6 JavaScript

In diesem Teil möchte ich nicht auf alle gebrauchten JavaScripts eingehen. Jedoch möchte ich das JavaScript **main.js** vorstellen, in welchem die Map für die Geolocation-Daten erstellt werden. Diese bauen auf Google Maps v3<sup>49</sup> auf.

Alle Standarduser Interfaces wurden deaktiviert und durch eigene ersetzt. Somit kann man diese später auch per CSS ansprechen. Dies ist bei den normalen Elementen nicht möglich, da die einzelnen **<div>** keine ID oder Namen besitzen.

Zu den normalen Buttons wie **Map** entspricht der Roadmap oder **Satellite** entspricht dem Hybrid Modus (Satellite und Bezeichnungen) wurden weitere Funktionen ergänzt.

<sup>48</sup> <http://peter.sh/experiments/vendor-prefixed-css-property-overview/> Stand 08.08.2012

<sup>49</sup> <https://developers.google.com/maps/documentation/javascript/> (Stand 30.07.2012)

Mit dem Button **zoom** kann der Benutzer direkt hinein- und wieder herauszoomen. Der Zoom Level ist dabei vorgegeben. Dies erleichtert die Bedienung, damit man für die Grobeinstellung des Zooms nicht die Fingerzeichen nutzen muss.

Mit dem Button **LocateME** wird der Benutzer aufgefordert, dass er dem Browser die Erlaubnis der Lokalisation erteilt. Über das Wireless Netzwerk oder GPS selber kann die Position des Gerätes auf ca. sechs bis zehn Meter bestimmt werden. Die ermittelte Position wird innerhalb eines Radius von 20 Metern angezeigt.

Mit einem Doppelklick kann ein Marker gesetzt werden, sollte bereits ein Marker positioniert sein, wird dieser an die neue Position gesetzt. Auch mit Drag & Drop kann dieser versetzt werden.

Eine spezielle Möglichkeit bietet uns das Eingabefeld **address**, dort haben wir die Möglichkeit eine Adresse direkt einzugeben und von Google suchen zu lassen.

```
input = document.getElementById('address');
autocomplete = new google.maps.places.Autocomplete(input);
autocomplete.bindTo('bounds', map);
```

Mit der Funktion **geocode()** werden die Daten per Callback bei Google gesucht.

```
function geocode() {
    geocoder.geocode({'latLng': marker.getPosition()}, function(results,
status) {
        for (var i = 0; i < results[0].address_components.length; i++)
        {
            var addr = results[0].address_components[i];
            if (addr.types[0] == 'postal_code')
                getZip = addr.long_name;
        }
        if (status == google.maps.GeocoderStatus.OK) {
            if (results[0]) {
                $('#address').val(results[0].formatted_address);
                $('#latitude').val(marker.getPosition().lat());
                $('#longitude').val(marker.getPosition().lng());
            }
        }
    });
}
```

## 4.7 Fotogalerie

Um effizient zu arbeiten, habe ich eine Fotogalerie gesucht, welche schon für mobile Applikationen optimiert wurde. Ich wurde mit der „Wonderwall Image Gallery“<sup>50</sup> von Codrops<sup>51</sup> fündig. Diese baut auf HTML5 und AJAX auf. Diese Gallery war in der vorgegebenen Art und Weise aber noch nicht für

<sup>50</sup> <http://tympanus.net/codrops/2010/05/27/awesome-mobile-image-gallery-web-app/> (Stand 09.08.2012)

<sup>51</sup> <http://tympanus.net/codrops/> (Stand 09.08.2012)

mein Projekt zu gebrauchen. Diese Galerie holt die Daten aus einem bestimmten Ordner oder Ordnerstruktur und zeigt die Ordner als Alben an. Durch meine Entscheidung eine [Imageablage](#) zu benutzen, konnte ich hier einen Vorteil geniessen. Die Galerie überprüft ausserdem, ob Thumbnails bestehen und wenn nicht, werden diese erstellt. Die Thumbs werden auch auf die Grösse geprüft und gegebenenfalls von der Galerie selbst überarbeitet.

Mit meiner Applikation wollte ich aber nicht, dass alle Frontuser alle Patientenbilder betrachten können. Das Ziel war, dass jeder Frontuser nur seine Patienten sieht. Somit musste ich Änderungen am Code vornehmen.

Über die Seite [gallery\\_client](#) kann jeder User einen seiner Patienten über ein Drop-Down-Menu auslesen. Mit einem Klick auf [continue](#) wird er auf die Galerie weitergeleitet. Diese wird geöffnet und es werden vom `$_POST` Array die gewünschten Daten in `hidden fields` gespeichert. Im JavaScript werden diese Felder dann herausgelesen und für die Abfragen weiter verwendet. Für die AJAX-Abfragen, werden diese über einen Link weiter gereicht. Unten sehen wir ein Beispiel mit welchem die Alben erstellt werden.

#### jquery.gallery.js

```
function loadAlbums ()
{
    var client = $('#client').val();
    var $loader = $('#albums_container').find('.loader');
    $loader.show();
    var url =
    '../../../../../magicpictures/assets/codrops_gallery/ajax/albums.php?client='+
    client;
    $.get(url, function(data) {
        $loader.hide();
        $('#albums').html(data).show();
    }, 'html');
}
```

#### AJAX

```
$client = $ GET['client'];
if(file_exists('../../../../../magicpictures/clientdata/'.$client)){
    $files =
    array_slice(scandir('../../../../../magicpictures/clientdata/'.$client), 2);
    if(count($files))
    {
        natcasesort($files);
        foreach($files as $file){
            if($file != '.' && $file != '..'){
                echo '<li id="'.$file.'" class="arrow"><a
href="#thumbs_container">'.$file.'</a></li>';
            }
        }
    }
}
```

## Ordnerstruktur

```
../client_nr/id-name-episode/images|thumbs/datei
```

Client Nr.: PA1010

Episode of Care: 7-care1-2012-07-30

Unter Episode of Care existieren zwei Unterordner **images** und **thumbs**.

Die Datei kann eine .gif, .png, .jpg oder .jpeg Bilddatei sein.

Zusammengesetzt sieht die Struktur dann so aus:

```
../PA1010/7-care1-2012-07-30/images/test.jpg.
```



# Schluss teil

---

## 5 Fazit

Tief durchatmen und zurück blicken. Die letzten Tage gingen schneller vorbei, als man sich das vorstellen konnte. Im Februar habe ich mich für das Thema: „Medical Photography management and clinical integration (MePhoCI)“ entschieden. Zusammen mit meinem Betreuer Herr Dr. Henning Müller haben wir uns für einen Weg entschieden, welcher mir erlaubte bekannte Methoden und Technologien mit mir noch unbekannten zu verknüpfen.

Bisher habe ich in jedem PHP Projekt von Grund auf programmiert. Dies hatte den Vorteil, dass man alle Teile bis ins letzte Detail kannte und auch die Übersicht behielt. Für diese Arbeit habe ich mich entschieden mit einem Framework und diversen externen Libraries zu arbeiten. Wie bei anderen Gegebenheiten im Leben, hat auch diese Medaille zwei Seiten. Es wird viel Arbeit abgenommen, indem schon Libraries oder Konfigurationsdateien vorbereitet sind. Aber mit einem Umfang von über 1'100 Dateien und ca. 300 Ordnern ist es nicht leicht den Überblick zu behalten. Änderungen in den Core Dateien sind auch schwieriger, sollte dies nötig sein. Wie so oft im Leben gilt auch hier das Sprichwort:

„Übung macht den Meister.“

Resultate sind schneller sichtbar bei einem Framework. Spezial-Arbeiten können aber zu einem richtigen Kampf ausarten. Ich bin Herr Dr. Henning Müller dankbar, dass ich diese Erfahrung mit dem Framework machen durfte. Ich bin zufrieden mit dem Ablauf der Arbeit und freue mich, meine Resultate präsentieren zu können.

## 6 Stärken und Schwächen der Software

Tabelle 7: Stärken und Schwächen der Software

Stärken	Schwächen
<ul style="list-style-type: none"> <li>• MVC Pattern</li> <li>• Objektorientiert</li> <li>• Trennung von System und Applikation</li> <li>• Sicherheitsaspekte einfach zu integrieren</li> <li>• CodeIgniter hinterlässt einen kleinen Footprint</li> <li>• Einfache Integration von Erweiterungen</li> <li>• Einfaches Handling</li> <li>• Wizard ermöglicht fließende Erfassung</li> </ul>	<ul style="list-style-type: none"> <li>• Implementation von Ein-Mann-Entwicklungen wie groceryCRUD</li> <li>• HTML5 ist noch unsicher (Draft-Zustand)</li> <li>• HTTPS Implementierung muss vorgenommen werden (Datenschutz)</li> <li>• Traffic Optimierung</li> </ul>

## 7 Zukunft

### 7.1 Verbesserungsmöglichkeiten

Das System ist so aufgebaut, dass sämtliche Teile einfach und gut erweitert werden können. Durch das MVC Muster und die objektorientierte Programmierung ist die nötige Flexibilität gegeben.

Dem zukünftigen Entwickler sind fast keine Grenzen gesetzt. Sinnvolle Erweiterungen könnten sein:

- PDF Tool, welches dem Benutzer die Möglichkeit gibt ein PDF zu einem Patienten zu erstellen. Dieses PDF dient dann als eine Akte für die schriftliche Ablage. Bei Bedarf kann sie ausgedruckt und dem Patienten weitergegeben werden. Dieses PDF könnte alle Information und Bilder enthalten, die Bilder könnte man als Listen den Episodes of Care zuordnen.

- Mehrsprachigkeit sollte in einem nächsten Schritt implementiert werden. Dazu müssten nur einige Tabellen angepasst werden. CodeIgniter unterstützt leider nur die Mehrsprachigkeit von Labels.
- Ein Angebot diverser CSS Designs bieten, damit der Benutzer das Aussehen der Webseite seinen Wünschen anpassen kann.
- Eine Printfunktion mit welcher die Drucker des Spitals angesprochen werden könnten, um die PDFs auszudrucken.
- Im Moment werden die Bilder als Metadaten abgespeichert. Damit diese im Spitalsystem weiterverwendet werden können, müsste eine Schnittstelle für DICOM eingerichtet werden.
- Passwortwechsel wird nach drei Monaten erzwungen; die letzten vier Passwörter dürfen dabei nicht wieder verwendet werden.
- Statt einer Verschlüsselung mit MD5 Hash könnte eine AES Verschlüsselung zum Tragen kommen.

## 7.2 Weitere Verwendungszwecke

Mit einigen Anpassungen könnte diese Applikation auch in anderen Bereichen der Wirtschaft und der Verwaltung eingesetzt werden.

- Bei Versicherungen, um Schadensfälle zu dokumentieren.
- Bei der Polizei, um Unfälle oder allgemeine Tataufnahmen zu dokumentieren.
- Bei Banken, um eine Hypothek mit Bildern zu unterlegen.
- Für Handwerker, um auf den Baustellen ihre Arbeit zu raportieren.

Um es mit anderen Worten zu sagen, diese Anwendung in abgeänderter Form, könnte überall an der Front zum Einsatz kommen, wo die Arbeit mit Bildern und Daten dokumentiert werden muss.

## 8 Quellenverzeichnis

### 8.1 Geschriebene Quellen



#### **Apps mit HTML5 und CSS3**

*Für iPad, iPhone und Android*

Galileo Press GmbH, Rheinwerkallee 4, 53227 Bonn;

Florian Franke, Johannes Ippen; 2012;

ISBN 978-3-8362-1848-1;

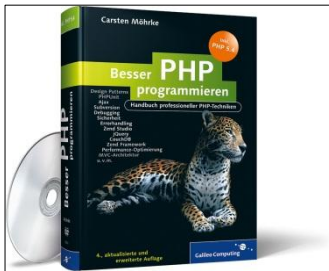
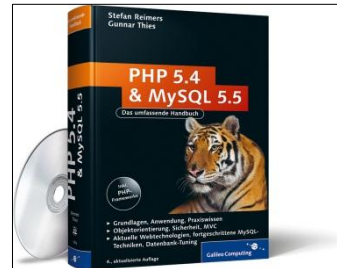
#### **PHP 5.4 und MySQL 5.5**

*Das umfassende Handbuch*

Galileo Press GmbH, Rheinwerkallee 4, 53227 Bonn;

Stefan Reimers, Gunnar Thies; aktualisierte Auflage 2012;

ISBN 978-3-8362-1876-4



#### **Besser PHP programmieren**

*Handbuch professioneller PHP-Techniken*

Galileo Press GmbH, Rheinwerkallee 4, 53227 Bonn;

Carsten Möhrke; aktualisierte und erweiterte Auflage 2012;

ISBN 978-3-8362-1741-5

### 8.2 Weitere Quellen

Alle Quellen sind direkt auf den jeweiligen Seiten als Fussnoten vermerkt.

Damit auch Leser ohne Fachwissen diese Arbeit verstehen, sind bei wichtigen Ausdrücken Fussnoten angegeben, welche auf Wikipedia verweisen. Diese Links sind als rudimentäre Information gedacht.

Das Titelbild ist von topnews.ae (<http://topnews.ae/content/24531-how-much-insurers-will-spend-medical-care-be-decided-soon> (Stand 09.08.2012))

## 9 Abkürzungsverzeichnis / Worterklärungen

• agile Methoden	Methoden zur Softwareentwicklung, wie zum Beispiel SCRUM
• AJAX	Asynchrone JavaScript und XML: Dies ermöglicht die asynchrone Datenübertragung. Seiten können so geändert werden, ohne die Seite neu zu laden.
• Apache	Apache ist ein open-source HTTP Server.
• API	Application Programming Interface / Programmierschnittstelle
• App	Applikation
• ASP.net	Active Server Pages .NET ist eine serverseitige Programmiersprache
• BA	Bachelor Arbeit
• Backend	Administrativer Teil des Softwareprogrammes
• BLOB	Binary Large Object, diese Objekte können Bild- oder Audiodateien sein.
• C#	C# (C-Sharp) ist eine Programmiersprache von Microsoft
• CodeIgniter	CodeIgniter ist ein PHP Framework
• Cross Site Request Forgery	Website übergreifende Anfragefälschung
• Cross Site Scripting	Seitenübergreifendes Scripting
• CRUD	Create, Update und Delete
• Datenbankdump	Kopie oder Auszug einer Datenbank
• Derivate	abgeleitete Systeme, zum Beispiel Unix Derivate: Linux, OS X, GNU Hurd etc.
• DICOM	Digital Imaging and Communications in Medicine
• Double Opt-in	Mailanfrage, ob der Benutzer sich wirklich anmelden will. (Activate/Delete)
• Draft Version	Entwurfsversion
• dump	Kopie oder Auszug
• Eclipse	Programmierungsumgebung für diverse Programmiersprachen
• eHealth Projects	Information und Kommunikations Technologie für Gesundheit
• episode of care	Behandlungszeitraum
• File System	Ein Dateisystem ist Ablageorganisation auf einem Datenträger. Bestandteil des Betriebssystems.
• Firebird	open-source Datenbank
• Flat Files	Datei mit strukturiertem Text. Diese Dateien stehen nicht in Zusammenhang mit anderen Dateien.
• Foreign Key	Fremdschlüssel. Referenzfeld, welche auf eine andere Tabelle verweist.
• Framework	Ein Framework unterstützt den Entwickler beim Programmieren, ist aber selber kein eigenständiges Programm.
• Frontend	Der operative Teil des Softwareprogramms
• gasORM	Eine Library für CodeIgniter, welches das object-relational mapping erleichtert
• groceryCRUD	Eine Library für groceryCRUD, welche das Erstellen von Tabellen und deren CRUD Funktionen erleichtert
• Hash	Streuwertfunktion zerhackt eine Zeichenkette einmal oder mehrmals.
• HMI	Human Machine Interface / Mensch-Maschinen-Schnittstelle
• IDE	Integrierte Entwicklungsumgebung
• InnoDB	Datenablagefunktionseinheit für MySQL, welche die Verwaltung von Beziehungen unter den Tabellen ermöglicht
• iOS	Betriebssystem von Apple

• <b>iPad</b>	Tablet Computer von Apple
• <b>iPhone</b>	Smartphone von Apple
• <b>Jailbreak</b>	Mit einem Jailbreak werden die Nutzungsbeschränkungen aufgehoben
• <b>LAMP</b>	Ein Softwarepaket, welches folgende Komponente verwendet: Linux, Apache, MySQL, PHP (Perl oder Python möglich)
• <b>Library</b>	Eine Sammlung von Unterprogrammen. (Programmbibliothek)
• <b>Linux OS</b>	Betriebssystem von Linux
• <b>MD5</b>	Message-Digest Algorithm 5 ist ein Algorithmus und erstellt einen Hash
• <b>MePhoCI</b>	Medical Photography management and clinical integration (Titel der Arbeit)
• <b>MS SQL Server</b>	Microsoft SQL Server
• <b>multiprocessing</b>	Simultanverarbeitung
• <b>multithreading</b>	Mehrere Aktionen können parallel ausgeführt werden
• <b>MySQL</b>	Datenbanksystem
• <b>MySQL Workbench</b>	Modellierungswerkzeug für MySQL Datenbanken
• <b>Namespace</b>	Mit einem Namensraum lassen sich Codeteile einfacher implementieren und finden
• <b>native</b>	systemeigen
• <b>native App</b>	systemeigene Applikation z.B. für Android oder iOS
• <b>NetBeans</b>	Programmierungsumgebung für diverse Programmiersprachen
• <b>object-relational mapping</b>	Abbildung der relationalen Datenbank ist eine Technik damit eine objektorientierte Programmiersprache seine Objekte in der Datenbank ablegen kann.
• <b>Oracle</b>	Datenbankanbieter
• <b>OS</b>	Operating System / Betriebssystem
• <b>Overhead</b>	Mehraufwand
• <b>Paginierung</b>	Nummerierung / Seitenzählung / Seitenangabe
• <b>Peripherie</b>	Umfeld, (Programmierungsumfeld) Server, DIE etc.
• <b>PHP</b>	PHP: Hypertext Preprozessor. Serverseitige Programmiersprache
• <b>phpMyAdmin</b>	Online Administrationstool für MySQL
• <b>PostgreSQL</b>	Datenbanksystem
• <b>Primary Key</b>	Primärschlüssel
• <b>Printscreen</b>	Momentaufnahme des Bildschirms
• <b>Product Backlog</b>	Eine Art Auflistung aller anstehenden Aufgaben, beschrieben aus Sicht des Kunden
• <b>Proxy</b>	Stellvertreter
• <b>readonly</b>	Lesezugriff / schreibgeschützt
• <b>reCaptcha</b>	completely automated public Turing test to tell computers and humans apart. Mit dieser Vorsichtsmassnahme kann man sich vor Bots schützen. Diese können das Captcha-Feld nicht lesen und sich somit auch nicht anmelden.
• <b>regex</b>	regulärer Ausdruck, welcher Muster in einer Zeichenkette prüfen kann. Dient zur Beschreibung von Zeichenketten.
• <b>rendern</b>	vorbereiten / berechnen und ausgeben
• <b>SCRUM</b>	Modell zur Softwareentwicklung
• <b>Sendepuffer</b>	Zwischenspeicher beim Versenden von Daten
• <b>Smartphone</b>	Ein modernes Mobiltelefon, welches als kleiner transportabler Computer bezeichnet werden kann (z.B. iPhone)
• <b>SQL</b>	Standard Query Language ist eine Abfragesprache welche in Datenbanksystem zum Einsatz kommt
• <b>SQLite</b>	Datenbanksystem
• <b>Support</b>	Unterstützung, Hilfestellung

- |                            |                                                                                                                                    |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| • <b>Symfony Framework</b> | Symfony Framework ist ein PHP Framework                                                                                            |
| • <b>Tabbing</b>           | Register                                                                                                                           |
| • <b>Tablet</b>            | Flacher und tragbarer Computer ohne Tastatur, aber mit Touchscreen (leichte Ausführung)                                            |
| • <b>Thumbnail</b>         | Vorschaubild                                                                                                                       |
| • <b>Trigger</b>           | Einen Event in der Datenbank löst einen Trigger (Auslöser) aus, welcher eine oder mehrere Aktionen veranlassen kann.               |
| • <b>Upload</b>            | Hochladen von Daten                                                                                                                |
| • <b>URL</b>               | Uniform Resource Locators / einheitlicher Quellenanzeiger<br>Beispiel: <a href="http://www.example.com">http://www.example.com</a> |
| • <b>VM</b>                | Virtuelle Maschine                                                                                                                 |
| • <b>XAMP</b>              | X (diverse Betriebssysteme), Apache, MySQL, PHP (Perl oder Python möglich)                                                         |
| • <b>Zend Framework</b>    | Zend Framework ist ein PHP Framework                                                                                               |

## 10 Index

### A

Android ..... 13, 22, 26, 83, 85

### B

Bachelor Arbeit 2, 4, 11, 17, 29, 53, 60, 61, 62, 63, 69, 72,  
..... 84  
Backend ... 4, 20, 31, 33, 34, 37, 44, 54, 56, 58, 64, 72, 75,  
..... 84  
BLOB ..... 11, 28, 29, 84

### C

Captcha ..... 33, 34, 46, 63, 85  
CodeIgniter .. 17, 18, 20, 27, 57, 58, 60, 61, 62, 63, 66, 67,  
..... 68, 69, 72, 73, 81, 82, 84  
Cross Site Request Forgery ..... 66, 84  
Cross Site Scripting ..... 66, 84  
CSS(3) ..... 24, 25, 26, 33, 73, 74, 75, 76, 82, 83

### D

Draft ..... 11, 12, 21, 81, 84

### F

Flat Files ..... 27, 84  
Foreign Key ..... 16, 36, 55, 84  
Fotogalerie ..... 39, 41, 47, 48, 77  
Frontend ..... 31, 33, 34, 36, 41, 44, 46, 48, 84

### G

gasORM ..... 17, 18, 19, 67, 68, 70, 73, 84  
groceryCRUD ..... 17, 20, 69, 70, 72, 73, 75, 81, 84

### H

Hash ..... 35, 56, 65, 72, 82, 84, 85  
HTML(5) .. 11, 12, 13, 20, 22, 24, 25, 26, 73, 76, 77, 81, 83

### I

IDE ..... 11, 16, 17, 84  
Insert, Update, Delete .. 18, 20, 22, 35, 36, 38, 39, 44, 45,  
..... 57, 67, 69, 71, 72, 84  
iPad ..... 23, 24, 26, 83, 85  
iPhone ..... 23, 24, 26, 83, 85

### J

Jailbreak ..... 22, 23, 24, 85  
JavaScript Derivate ... 11, 12, 14, 16, 33, 73, 76, 77, 78, 84

### L

LAMP ..... 16, 85  
Libraries . 16, 18, 20, 57, 58, 59, 61, 67, 69, 73, 75, 80, 84,  
..... 85

### M

MePhoCI ..... 4, 80, 85  
MySQL & Tools ..... 11, 16, 17, 28, 53, 67, 83, 84, 85, 86

### N

Namespace ..... 85  
NetBeans ..... 17, 85

### O

Object-relational mapping ..... 18, 84, 85



---

**P**

PHP... 11, 13, 14, 15, 16, 17, 28, 57, 59, 80, 83, 84, 85, 86  
 Primary Key..... 18, 68, 85  
 Product Backlog .....11, 29, 30, 31, 32, 85, 90

---

**R**

readonly..... 38, 41, 49, 85  
 Reguläre Ausdrücke..... 60, 85

---

**S**

Sicherheit.....24, 63, 65  
 Smartphone ..... 2, 31, 73, 85  
 SQL ..... 16, 68, 85  
 Symfony Framework..... 17, 18, 86

---

**T**

Tablet ..... 85, 86

Thumbnail.....40, 47, 55, 78, 86

---

**U**

URL.....59, 62, 64, 74, 86

---

**V**

Virtuelle Maschine..... 17, 31, 86

---

**X**

XAMP..... 86

---

**Z**

Zend Framework ..... 17, 18, 86

## 11 Ehrenwörtliche Erklärung

Ich bestätige hiermit, dass ich die vorliegende Bachelorarbeit alleine und nur mit den angegebenen Hilfsmitteln realisiert habe und dass ich ausschliesslich die erwähnten Quellen benutzt habe. Ohne Einverständnis des Leiters des Studienganges und des für die Bachelorarbeit verantwortlichen Dozenten Hr. Dr. Henning Müller werde ich dieses Dokument an niemanden verteilen, ausser an die Personen, welche mir die wichtigsten Informationen für die Verfassung dieser Arbeit geliefert haben.

Susten, den 09. August 2012

.....  
Philipp Oggier

## 12 Anhang

- Daten der Bachelorarbeit (S. 90)
- Stunden (S. 91 – S. 95)
- Product Backlog (S. 96 – S. 98)
- Manual (S. 99 – S. 118)
- Datenbankmodell A3 zum Herausklappen (S. 119)