

Seminararbeit 2014
Homepage "hochsicherheit.ch"



Verfasser	Severin Müller
Dozent	Matthias Bachmann
Modul	Programmieren
Seminar	"Webprojekt in PHP/MySQL"
Erscheinungsjahr	2014

Danksagung

Ich möchte mich an dieser Stelle bei allen bedanken, die mich bei der Erstellung dieser Arbeit in irgendeiner Weise unterstützt haben. Besonders hervorheben möchte ich:

Bettina Hegenauer, für die bedingungslose Unterstützung während der stressigen Zeit
Christian Vetterlein, Prozesskoordinator, Schaffhausen für das Korrekturlesen der vorliegenden Arbeit.

Inhalt

1. Themenwahl und Aufgabenstellung	5
1.1 Eigenmotivation	5
1.2 Abgrenzung der Aufgabenstellung	5
2. Projektablauf	5
2.1. Vorbesprechung mit dem Kunden	5
2.2 Erfassung inhaltliche Wünsche	6
2.3. Erfassung gestalterische Wünsche	6
2.4. Erfassung funktionale Wünsche	6
2.5. Vorlegen von Design Beispielen	6
2.6. Abschluss der Spezifikation	6
2.7. Implementierung des Projekts	6
2.8. Testen des Projekts	6
2.9. Abnahme des Projekts durch den Kunden / Projektabschluss	6
3. Spezifikation	7
3.1 Requirement Engineering - Einführung	7
3.2 Entwurf	7
3.3 Umgebung	7
4. Lerninhalte	8
4.1 MVC	8
4.1.1 Einführung	8
4.1.2 Model	8
4.1.2 View	8
4.1.3 Controller	8
4.2 PHP Interfaces und Klassen	9
4.3. PHP und MySQL	9
4.3.1 MySql Befehle	9
4.4 Captcha	11
4.5 PHPUnit	11
5. Fertiger Prototyp	12
5.1 Kopfzeile	12
5.2 Navigationsleiste	12
5.3 Startseite	12
5.5 Über uns	12
5.5 Dienstleistungen	13
5.6. Allgemeine Geschäftsbedingungen	13
5.7. Referenzen	13

5.8 Links	14
5.9 Intern	14
6. Schlusswort	16
6.1 Rückblick	16
6.2 Fazit	16
7. Anhang.....	17
7.1 Anhang A: Bilderverzeichnis	17
7.2 Anhang B: Literaturverzeichnis	17

1. Themenwahl und Aufgabenstellung

1.1 Eigenmotivation

In der heutigen Zeit spielt das Internet eine zentrale Rolle im Privat- und Geschäftsleben. Schon früh habe ich mich mit dem Design und der Entwicklung von Homepages auseinandergesetzt weil ich diesen Trend erkannte und mich die Materie sehr interessierte. Nicht nur am Desktop, sondern auch unterwegs ist man heute meistens irgendwie mit dem Internet verbunden und man möchte Informationen jederzeit abrufen können.

Da aber längst nicht jeder sich mit dem Gestalten von Homepages auseinandersetzen benötigt man Fachleute, die sich dieser Thematik annehmen. Die Hochsicherheit GmbH ist ein junges Startup welches Sicherheits-Dienste für kleinere und mittelgrosse Veranstaltungen anbietet. Da dieses noch sehr junge Unternehmen einen Kundenstamm aufbauen möchte liegt die Vermutung nahe, dass ein ansprechender Internetauftritt dabei helfen kann. Da es sich beim Firmengründer um einen Freund von mir handelt, habe ich ihm angeboten, diesen Auftritt für ihn zu erstellen.

1.2 Abgrenzung der Aufgabenstellung

Das Erstellen einer Homepage ist technisch nicht besonders komplex. Einer der grössten Fehler die im Webdesign immer wieder gemacht werden ist das "drauflos Entwickeln" ohne mit dem Kunden Rücksprache oder ausreichend Rücksicht auf seine Anforderungen zu nehmen. Da der Kunde sich aber meist nicht bewusst ist, worauf es beim Gestalten ankommt ist die bilaterale Kommunikation von grosser Bedeutung. Daher möchte ich den Fokus in dieser Arbeit auf das Requirement Engineering legen.

Das Ziel dieser Arbeit ist es, eine Homepage zu erstellen die funktionsfähig ist und den Anforderungen des Kunden entspricht. Insbesondere die folgenden drei Teilziele sollen erreicht werden:

- Sauberes Requirement Engineering
- Test Driven Development
- Erfolgreiche Abnahme durch den Kunden

2. Projektablauf

Die nachfolgenden Abschnitte sollen eine kurze Zusammenfassung darüber sein, wie das Projekt ablaufen soll.

2.1. Vorbesprechung mit dem Kunden

Um das Projekt überhaupt starten zu können war eine Vorbesprechung mit dem Kunden nötig. Er hat folgende Wünsche geäussert:

"Ich möchte einen Internetauftritt für meine neue Firma haben die für Interessenten Informationen über unsere Dienstleistungen bereitstellt. Der Kunde soll Einsätze online buchen und mit uns über ein Formular Kontakt aufnehmen können. Ich habe eine Vorstellung davon, wie der Auftritt ungefähr aussehen soll und meine Partnerin hat bereit Logos kreiert."

Es wurde vereinbart, dass ich die Logos per E-Mail erhalte und mit die farblichen Wünsche schriftlich mitgeteilt werden.

2.2 Erfassung inhaltliche Wünsche

In einer zweiten Vorbesprechung hat mir der Kunde Entwürfe der einzelnen Seiteninhalte vorgelegt. Dieser werden direkt so übernommen.

2.3. Erfassung gestalterische Wünsche

Wie der Kunde bereits in der Vorbesprechung erwähnte, hat er klare Vorstellungen darüber wie der Internetauftritt gestaltet werden soll. Er wurde vereinbart, dass ich einen ersten Entwurf kreieren und vorlegen werde.

2.4. Erfassung funktionale Wünsche

Der Kunde hat zunächst gewünscht, dass die Einsätze online gebucht werden können. Im Verlaufe der Projekts hat er aber den Wunsch geäußert, von dieser Funktion abzusehen, weil er seine Termine lieber selber koordinieren möchte und so keine Kundenerwartungen wecken muss, die dann nicht erfüllt werden können. Das Kontaktformular soll aber wie gewünscht erstellt werden.

2.5. Vorlegen von Design Beispielen

Wie bereits erwähnt, wurde mit dem Kunden vereinbart, dass vor der eigentlichen Entwicklung ein Designbeispiel vorgelegt wird, dass bei Bedarf angepasst wird.

2.6. Abschluss der Spezifikation

Hier findet das letzte Treffen mit dem Kunden vor der Implementierung statt. Das Design und die Inhalte werden abgenommen, kleinere Anpassung können bei Bedarf aber natürlich vorgenommen werden.

2.7. Implementierung des Projekts

In einer ersten Phase wird aufgrund der Spezifikation ein Requirement Engineering durchgeführt. Dessen Zweck ist es, den Problemraum zu erfassen und zu dokumentieren und eine Verbindung zwischen Problem- und Lösungsraum herzustellen¹.

Sobald das Requirement Engineering abgeschlossen ist, kann die Implementierung beginnen.

2.8. Testen des Projekts

Wenn die Implementierung abgeschlossen ist, muss das Projekt getestet werden. Unit Tests ersetzen keinen finalen Test, in dem alle Funktionen ausreichend getestet werden.

2.9. Abnahme des Projekts durch den Kunden / Projektabschluss

Sobald das Projekt fertig implementiert und getestet wurde kann es dem Kunden zur Abnahme vorgelegt werden.

3. Spezifikation

3.1 Requirement Engineering - Einführung

Für die Abgrenzung der Anforderungen existieren verschiedene Modelle. Die Unterschiede sind aber gering. Gemäss IEEE wird das Requirement Engineering in folgende Schritte unterteilt²:

- Anforderungserhebung
- Anforderungsanalyse
- Anforderungsspezifikation
- Anforderungsbewertung

Wir wollen diese Schritte nun in unser Projekt einfliessen lassen. Die detaillierte Anforderungsbeschreibung finden Sie im Dokument "Anforderungsdokument".

3.2 Entwurf

Nachdem ich die gesamten Anforderungen mit dem Kunden besprochen und abgeschlossen hatte machte ich mir Gedanken zum Design. Mir schwebte eine einfach wartbare Applikation vor, bei der die Daten und die Präsentation sauber getrennt werden. Deshalb habe ich mich für ein MVC Pattern entschieden, welches Good Practice in der Webentwicklung ist.

Im Kontaktformular sollte die Eingabe validiert werden. Die Javascript Funktionalität lieferte Jaime Oberle im Kurs „Webprogrammierung mit PHP/MySQL“ in diesem Modul. Besten Dank dafür an dieser Stelle.

Die Umsetzung erfolgte mittels PHP, HTML, CSS und etwas Javascript.

3.3 Umgebung

Da für die gewünschte Homepage bestimmte Komponenten zur Verfügung stehen müssen, habe ich meinem Kunden angeboten, die Applikation auf dem Webserver bei meinem Hoster zu betreiben. Da weiss ich, dass ich sämtliche benötigten Teilsysteme verfügbar habe.

4. Lerninhalte

4.1 MVC

4.1.1 Einführung

Model View Controller (kurz MVC) ist ein Muster zur Strukturierung von Software-Entwicklung in die drei Einheiten Datenmodell (engl. model), Präsentation (engl. view) und Programmsteuerung (engl. controller)³

Das Ziel ist dabei die Trennung von der Programmlogik und der Darstellung. Dies hat den Vorteil, dass der Designer der Darstellung nicht mehr zwingend der Programmierer sein muss, sondern sich wirklich nur um das Design kümmern muss.

Wie MVC in diesem Projekt implementiert wurde ist im beiliegenden Entwurfsdokument nachzulesen.

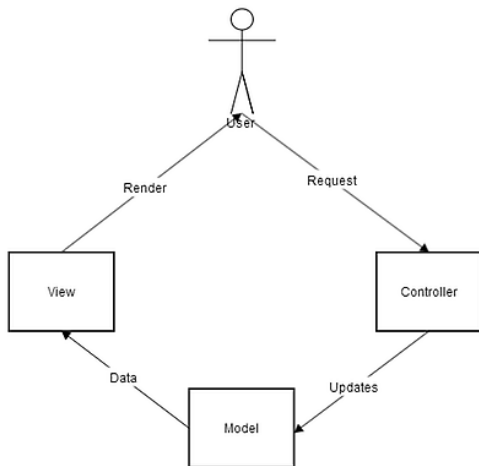


Abb. 1: MVC Modell

4.1.2 Model

Das Modell bearbeitet die Daten, die für die Darstellung notwendig sind. Sie werden vom Controller angefordert.

4.1.2 View

Die View ist das, was der User am Ende effektiv sieht. Sie erhält vom Model die Daten und wird typischerweise mit einem Template umgesetzt.

4.1.3 Controller

Der Controller ist für die Steuerung der Applikationen zuständig. Er nimmt Benutzeraktionen entgegen und leitet sie an die zuständige Stelle weiter.

4.2 PHP Interfaces und Klassen

Die Grundlagen der objektorientierten Programmierung dürfen wir für dieses Seminar als bekannt voraussetzen. Auf die allgemeine Theorie gehe ich daher an dieser Stelle nicht näher ein.

Seit Version 3 unterstützt PHP eine objektorientierte Syntax⁴. Genau wie in Java kann man mittels interfaces eine Schnittstelle schaffen und Methoden definieren, welche von einer Klasse implementiert werden müssen, ohne dass ein Implementierungsdetail preisgegeben werden muss. Eine Klasse kann ein Interface implementieren und dessen Methoden müssen überschrieben werden. Eine weitere Abstraktionsstufe kann mittels abstrakten Klassen erreicht werden.

4.3. PHP und MySQL

PHP liefert eine vollständige Unterstützung für MySQL. Somit kann eine Verbindung zu einem MySQL Server aufgebaut werden und mit dem Server interagiert werden.

4.3.1 MySql Befehle

Mit den folgenden Befehlen konnte bis anhin mit dem MySQL Server kommuniziert werden (Auszug)⁵:

- **mysql_close** — Schließt eine Verbindung zu MySQL
- **mysql_create_db** — Anlegen einer MySQL-Datenbank
- **mysql_error** — Liefert den Fehlertext der zuvor ausgeführten MySQL Operation
- **mysql_fetch_assoc** — Liefert einen Datensatz als assoziatives Array
- **mysql_num_rows** — Liefert die Anzahl der Zeilen im Ergebnis
- **mysql_query** — Sendet eine Anfrage an MySQL
- **mysql_select_db** — Auswahl einer MySQL Datenbank
- **mysql_result** — Liefert Ergebnis

Mit PHP 5.5 wurden diese Befehle allerdings für „deprecated“ also überholt erklärt. Der Grund dafür ist, dass mit diesen Befehlen sogenannte SQL Injection möglich war. SQL Injection ist eine Technik, wo ein Angreifer SQL Kommandos erstellt oder existierende verändert, um Schaden anzurichten⁶.

Um dieses Risiko zu verkleinern wird empfohlen mit PDO zu arbeiten. PDO steht für PHP Data Objects. Es handelt sich dabei um eine Schnittstelle, mit der Datenbanktransaktionen vorbereitet und durchgeführt werden können.

Ich möchte die Arbeitsweise von PDO an einem Beispiel erläutern. In meinem Design habe ich eine Klasse „MySQL“ erstellt, die mein Interface „Database“ implementiert. Ich habe darin eine Funktion connect() definiert, damit ich nicht jedes Mal wenn ich eine Transaktion auf der Datenbank ausführen möchte, alle Parameter neu setzen muss. Das ist Fehleranfällig und verursacht einen Mehraufwand der nicht notwendig ist.

```

public function connect() {

    try {
        $hoststr = "mysql:host=".$this->db_host.";dbname=".$this->db;
        return new PDO($hoststr, $this->db_user, $this->db_pass);
    }
    catch(PDOException $e) {
        echo $e->getMessage();
    }
}

```

Abb. 2: Funktion connect()

Die Funktion liefert ein neues PDO Objekt zurück, das als Datenbank-Handle dienen soll. Um ein PDO Objekt zu instanziiieren benötige ich den Host der Datenbank, den Benutzer und das Passwort des DB-Servers. Wie wir sehen können, müssen wir eine allfällige Exception abfangen.

Nun haben wir eine Verbindung zum MySQL-Server hergestellt. Wenn wir nun Abfragen ausführen wollen müssen wir diese entsprechend vorbereiten.

Wir bereiten einen Query-String vor und ersetzen konkrete Werte mit Platzhaltern. Dies tun wir, um SQL Injection zu vermeiden.

Mit PDO::Prepare können wir eine Statement Variable vorbereiten. Wenn wir das Objekt haben müssen wir die Platzhalter mit den „echten“ werten ersetzen. Die geschieht mittels der Funktion PDOStatement::bindParam().

Mit PDOStatement::Execute() wird dann die fertige Query ausgeführt. Beispiel:

```

public function add_user($user,$pass) {

    $sql = "INSERT INTO USERS (username,password) VALUES (:u, :p)";
    printf("%s\n",$sql);
    $dbh = $this->connect();
    $sth = $dbh->prepare($sql);
    $sth->bindParam(':u', $user, PDO::PARAM_STR);
    $sth->bindParam(':p', $pass, PDO::PARAM_STR);
    $sth->execute();
}

```

Abb. 3: Beispiel PDO Query

Wenn Daten mit SELECT abgefragt werden kann dann mit PDOStatement::fetchAll() das Resultat der Query zurückgegeben werden.

4.4 Captcha

Bereits 1978 wurde die erste SPAM-Mail versendet⁷. Mit der weltweiten Verbreitung des Internets stieg auch der Anteil an Spam E-Mail rasant an. Es wurden Roboter entwickelt, die Kontaktformulare auf Interseiten für Werbe- oder Schadmails missbrauchen.

Um die Flut an unerwünschten Mails einzudämmen wurden verschiedene Massnahmen entwickelt. Eine davon nennt sich Captcha. Mit Captcha muss der Benutzer auf einem Formular beweisen dass er menschlich ist, indem er ein oder zwei Wörter die durch eine Grafik dargestellt ist eintippen muss. Da diese Algorithmen verschlüsselt sind kann ein Roboter diese nicht interpretieren und das Formular so nicht absenden.

Ich habe auf dem Kontaktformular vom Auftraggeber Captcha eingebunden damit seine Mailbox einigermaßen sauber bleibt.

Die Implementierung ist sehr simpel. Man muss lediglich die Domäne, die geschützt werden soll bei Google eintragen und die Bibliothek in seinem Projekt einbinden. Sobald man sich auf Google eingetragen hat wird ein Satz von sogenannten Keys generiert; ein private key, der zur Kommunikation zwischen dem Dienst von Google und der Homepage dient und ein public key für die Kommunikation zwischen den Benutzern und dem Server der Homepage.

Wenn das Kontaktformular abgesendet wird, wird mit der Methode `recaptcha_check_answer()` die Eingabe überprüft und entsprechend reagiert.

4.5 PHPUnit

Da ich grundsätzlich Test Driven entwickeln möchte habe ich mir Gedanken über Unit Tests gemacht. Relativ schnell kam ich bei den Recherchen dazu auf PHPUnit. PHPUnit ist einfach zu installieren und bietet mir die Möglichkeit echt Unit Tests zu schreiben. Ich bevorzuge dies, weil ich so die Methoden testen kann bevor ich sie effektiv benutze.

Es macht aus meiner Sicht wenig Sinn, die Darstellungen zu testen. Jedoch gibt es die eine oder andere Datenbank-Transaktion von denen ich sicher sein möchte, dass sie funktionieren bevor ich diese in meinen Code einbaue.

Ein PHPUnit Test ist folgendermassen aufgebaut:

Man erstellt eine Klasse die die Tests enthalten soll. Diese erweitert die PHPUnit Klasse `PHPUnit_Framework_TestCase`. Dabei müssen die abstrakten Methoden „`setUp()`“ und „`tearDown()`“ überschrieben werden. Wenn es nichts aufzusetzen gibt, kann man diese leer lassen. Es gilt die Faustregel: für jeden Wert, den wir testen wollen wird ein Unit Test erstellt.

Zu beachten ist dabei, dass eine Testfunktion immer mit „`test`“ beginnen muss. Gross- und Kleinschreibung wird dabei beachtet.

Wenn man seine Testklasse fertig erstellt hat, kann man sie auf der Kommandozeile mit dem Befehl `phpunit <TestKlasse.php>` ausführen. Das Resultat wird dann ausgegeben.

Welche Unit Tests bei diesem Projekt zum Einsatz kamen ist dem Testdokument zu entnehmen.

5. Fertiger Prototyp

5.1 Kopfzeile

rsicherheit/index.php?page=home

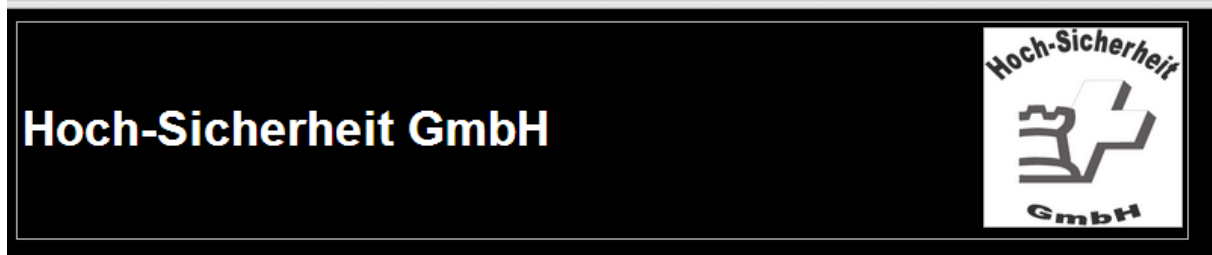


Abb. 4: Kopfzeile Homepage

5.2 Navigationsleiste



Abb. 5: Navigationsleiste

5.3 Startseite



Abb. 6: Startseite

5.5 Über uns

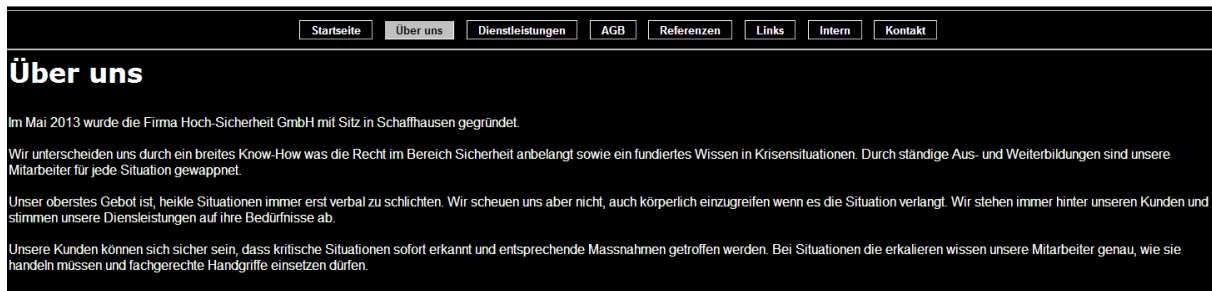


Abb. 7: Über uns

5.5 Dienstleistungen

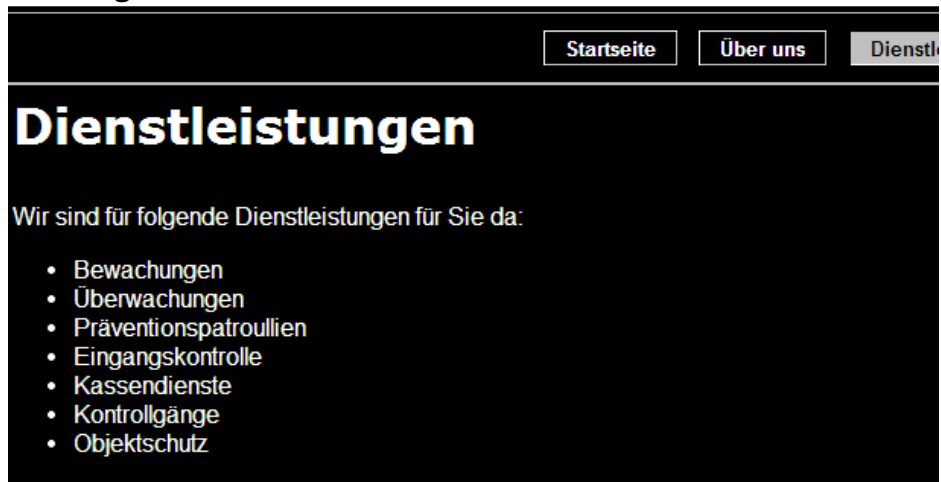


Abb. 8: Dienstleistungen

5.6. Allgemeine Geschäftsbedingungen

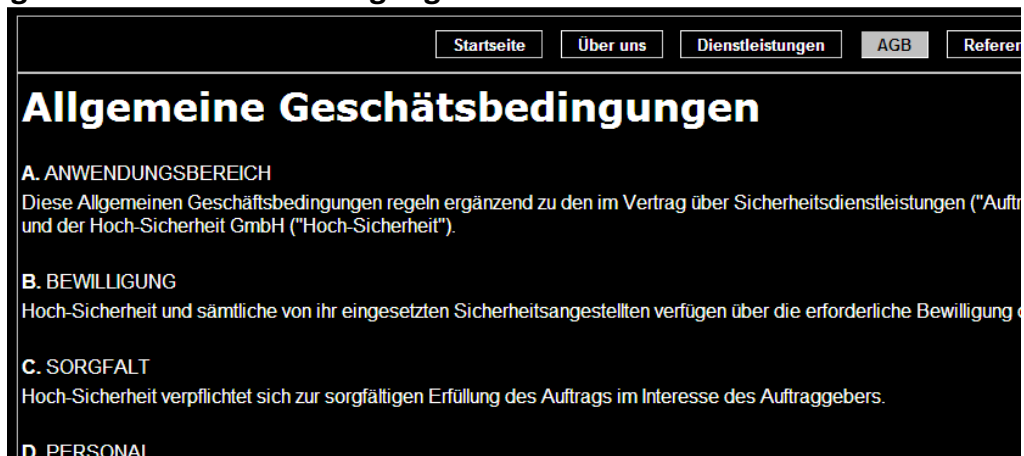


Abb. 9: Allgemeine Geschäftsbedingungen

5.7. Referenzen



Abb. 10: Referenzen

5.8 Links

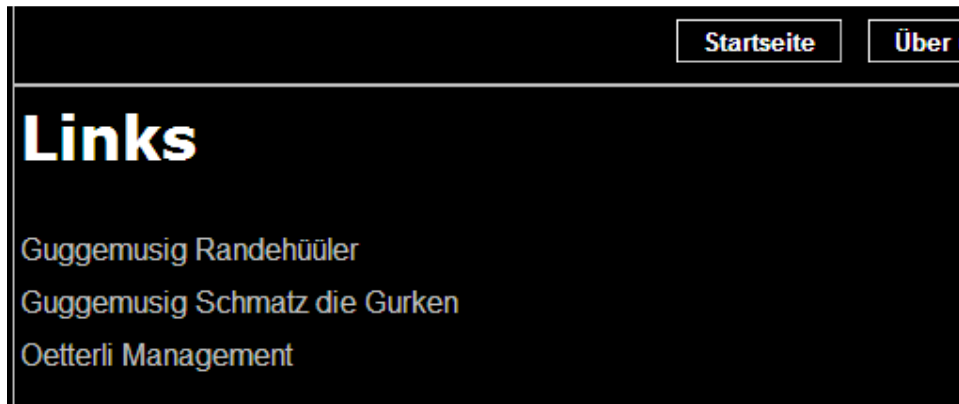


Abb. 11: Links

5.9 Intern

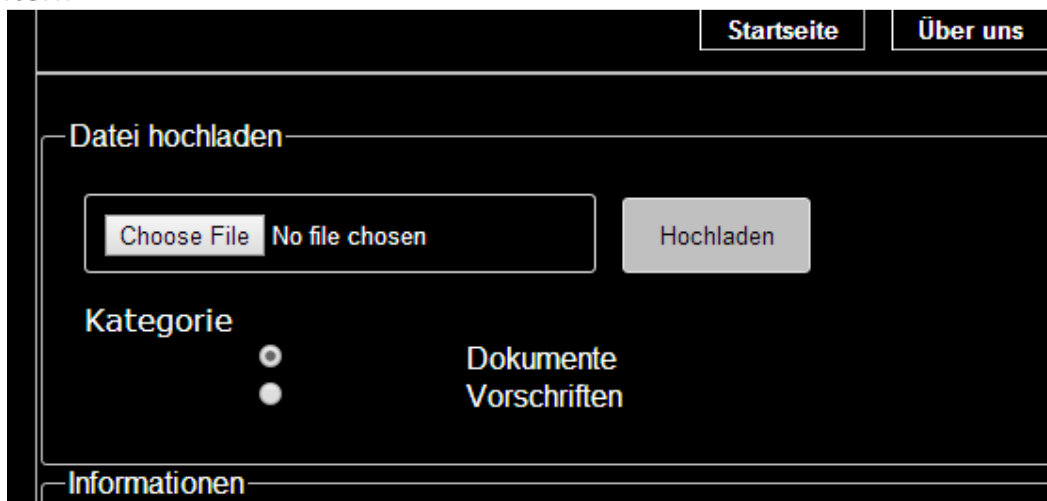


Abb. 12: Interner Bereich

Mit Dokumenten:



Abb. 13: Interner Bereich mit Dokumenten

5.10 Kontakt

[Startseite](#)
[Über uns](#)
[Dienstleistungen](#)
[AGB](#)
[Referenzen](#)
[Links](#)
[Intern](#)
[Kontakt](#)

Kontaktinformationen

Vorname

Vorname

Nachname

Nachname

mail@beispiel.ch

Email-Adresse

+41 52 123 45 67

Telefon

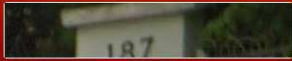
+41 79 123 45 67

Mobil


Ihre Anfrage

Ihre Anfrage

Anti-Spam - Bitte geben Sie die angezeigten Zeichen ein:



Geben Sie den angezeigten



reCAPTCHA™
stop spam,
keep people

Senden

Abb. 14: Kontaktformular mit CAPTCHA

6. Schlusswort

6.1 Rückblick

Wir erinnern uns an die Zielsetzungen in Kapitel 1.2:

- Sauberes Requirement Engineering
- Test Driven Development
- Erfolgreiche Abnahme durch den Kunden

Ich glaube, das Requirement Engineering wurde sorgfältig durchgeführt, zumal der Kunde die Homepage so abgenommen hat. Das Deployment erfolgt voraussichtlich im Juli 2014, für den Fall, dass er noch weitere Unterseiten möchte. Damit sind aus meiner Sicht die Ziele 1 und 3 erfüllt.

Ausserdem habe ich für die MySQL Operationen jeweils einen Unit Test geschrieben und durchgeführt, bis diese korrekt funktionierten bevor ich diese effektiv verwendet habe. Somit glaub ich auch das Ziel 2 erfüllt zu haben.

6.2 Fazit

Das Projekt hat mich persönlich eine grosse Freude bereitet zumal ich neu erlerntes anwenden konnte und zum ersten Mal PHPUnit eingesetzt habe. Vom Aufwand her war es für eine Seminararbeit grenzwertig zumal das Requirement Engineering viel Zeit in Anspruch genommen hat. Aber schlussendlich war es möglich das Projekt in der geforderten Zeit umzusetzen, was mich sehr freut.

Abschliessend bleibt zu sagen, dass das ganze Thema PHP und MySQL sehr interessant ist zumal Webseiten so sehr dynamisch gestaltet werden können.

Zukünftig würde ich jedoch eher auf ein Framework setzen, da viel Basis-Code so automatisch generiert werden kann.

Alles in allem: Das nächste Projekt kann kommen....

7. Anhang

7.1 Anhang A: Bilderverzeichnis

Abb. 1: MVC Modell	8
Abb. 2: Funktion connect()	10
Abb. 3: Beispiel PDO Query	10
Abb. 4: Kopfzeile Homepage	12
Abb. 5: Navigationsleiste	12
Abb. 6: Startseite	12
Abb. 7: Über uns	12
Abb. 8: Dienstleistungen	13
Abb. 9: Allgemeine Geschäftsbedingungen	13
Abb. 10: Referenzen	13
Abb. 11: Links	14
Abb. 12: Interner Bereich	14
Abb. 13: Interner Bereich mit Dokumenten	14
Abb. 14: Kontaktformular mit CAPTCHA	15

7.2 Anhang B: Literaturverzeichnis

- [1]: <http://www11.informatik.uni-erlangen.de/Lehre/SS2010/PR-SWE/Folien/fa1.pdf>, Folie 5, abgerufen am 29.03.2014
- [2]: [http://de.wikipedia.org/wiki/Anforderungsanalyse_\(Informatik\)](http://de.wikipedia.org/wiki/Anforderungsanalyse_(Informatik)), abgerufen am 29.03.2014
- [3]: https://de.wikipedia.org/wiki/Model_View_Controller, abgerufen am 29.03.2014
- [4]: <http://www.php.net/manual/de/history.php.php>, abgerufen am 12.04.2014
- [5]: <http://ch2.php.net/manual/de/ref.mysql.php>, abgerufen am 21.04.2014
- [6]: <http://www.php.net/manual/de/security.database.sql-injection.php>, abgerufen am 30.04.2014
- [7]: <http://blog.wiwo.de/look-at-it/2013/01/31/infografik-die-geschichte-des-internets-1969-bis-2012/>, abgerufen am 14.05.2014

Weitere Quellen, die bei der Erstellung dieser Arbeit hinzugezogen wurden

<http://www.php.net>

<http://www.sitepoint.com/tutorial-introduction-to-unit-testing-in-php-with-phpunit/>