**Entwurfsdokument**

**Homepage “hochsicherheit.ch”**

****

|  |  |
| --- | --- |
| Verfasser | Severin Müller |
| Dozent | Matthias Bachmann |
| Modul | Programmieren |
| Seminar | “Webprojekt in PHP/MySQL” |
| Erscheinungsjahr | 2014 |

Table of Contents

[1. Einleitung 3](#_Toc390041068)

[2. Modelle 4](#_Toc390041069)

[2.1 MVC 4](#_Toc390041070)

[2.2 Klassendiagramm 4](#_Toc390041071)

[2.3. Architekturmodell 5](#_Toc390041072)

[3. Implementierung 6](#_Toc390041073)

[3.1 Grundgerüst 6](#_Toc390041074)

[3.1.1 Allgemein 6](#_Toc390041075)

[3.1.2 Unterseiten 6](#_Toc390041076)

[3.2 Kontaktformular 7](#_Toc390041077)

[3.3 Interner Bereich 7](#_Toc390041078)

[3.3.1 Allgemein 7](#_Toc390041079)

[3.3.2 Dokumentenverwaltung 7](#_Toc390041080)

[3.3.3 Detailimplementierung Dokumentenverwaltung 8](#_Toc390041081)

[4. Anhang 10](#_Toc390041082)

[4.1 Anhang A: Bilderverzeichnis 10](#_Toc390041083)

# 1. Einleitung

Aufbauend auf der Anforderungsanalyse soll in dieses Dokument festlegen, wie die Anforderungen umgesetzt werden.

Die Anforderungsanalyse wurde komplett durchgeführt und ist im entsprechenden Dokument nachzulesen.

# 2. Modelle

## 2.1 MVC

Als Pattern wurde das MVC Pattern gewählt. Dieses Pattern ermöglicht eine saubere Trennung zwischen Daten und Ansichten.

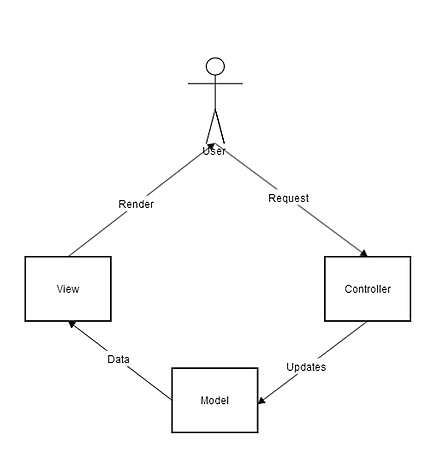


Abb. 1: MVC Modell

# 2.2 Klassendiagramm

Das nachfolgende Diagramm zeigt ein vereinfachtes Klassen Diagramm wie das Projekt implementiert wurde

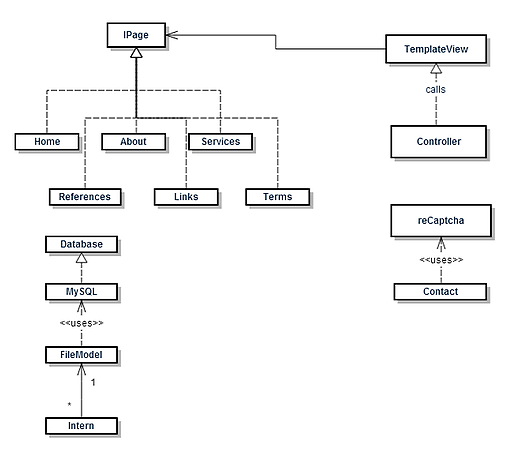


Abb. 2: Objektdiagramm

## 2.3. Architekturmodell

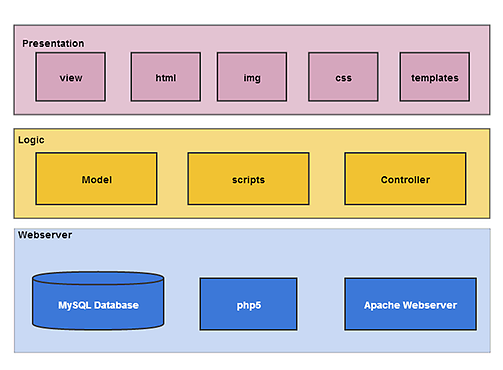


Abb. 3: Architekturmodell

# 3. Implementierung

## 3.1 Grundgerüst

### 3.1.1 Allgemein

Die Hauptseite stellt den Eingangspunkt der Homepage dar. Das grundlegende Design ist hier festgelegt und der Inhalt wird über die Unterseiten eingefügt.

Mit einem Klick auf eine Unterseite wird der Controller angerufen welcher den Request an die Klasse TemplateView weiterleitet. In der Methode

indexAction()

wird die Unterseite generiert und ausgegeben.

Das Grundgerüst soll aus einem Interface bestehen, welches von den meisten Unterseiten implementiert wird. Im Sinne von MVC soll die Ansicht von der Logik getrennt werden. So ist es auch einfach, weitere Unterseiten ohne grossen Aufwand zu erstellen.

Das Interface besteht aus den Methoden

public function getPageTitle()

public function getPageContent()

So kann ein Grunddesign mit einem Titel und einem Inhalt erstellt werden.

### 3.1.2 Unterseiten

Eine Unterseite soll im Order “view” abgelegt werden und eine Klasse instanziieren, die das interface IPage implementiert.

Eine Unterseite enthält nur das nötigste an Code. Beispiel Unterseite Home:

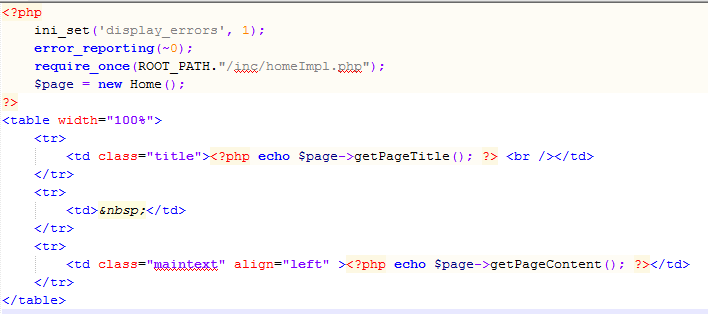


Abb. 4: Beispiel Unterseite

Ein Objekt der Klasse Home wird instanziiert und mit den beiden Methoden wird der Inhalt generiert.

## 3.2 Kontaktformular

Das Kontaktformular wird implementiert das interface IPage nicht, da die Methoden nicht benötigt werden. Das Design wird im Ordner View abgelegt. Mittels Javascript können Validierungen der Formularfelder eingebaut werden.

Beim Absenden des Formulars soll eine E-Mail generiert werden und eine Mail wird an die E-Mail Adresse des Kunden verschickt.

Als Antispam Massnahme soll die Open-Source library “reCaptcha” von Google eingebunden werden.

Wird das Formular abgeschickt wird der Captcha Code geprüft und das E-Mail entweder verschickt oder es wird dem Benutzer ein entsprechender Inhalt ausgegeben.

## 3.3 Interner Bereich

### 3.3.1 Allgemein

Der interne Bereich soll den Benutzern nicht öffentlich zugänglich gemacht werden, sondern mit einem Passwort zu schützen. Die Umsetzung erfolgt mittels einem Login das MySQL verwendet.

Um einen Wechsel der Datenbank zu vereinfachten ist ein interface “Database” zu erstellen, welche einige Methoden bereitstellt:

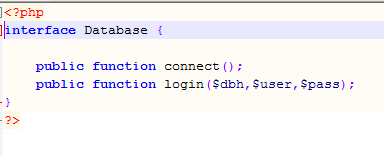


Abb. 5: Interface "Database"

Für die MySQL Implementierung erstellen wir eine Klasse MySQL die das Interface implementiert.

### 3.3.2 Dokumentenverwaltung

Die Dokumente sollen in der Datenbank abgelegt werden. Für diesen Zweck wurde eine Tabelle in der Datenbank angelegt die die Dateinformationen enthält.

Folgende Informationen sind in der Tabelle enthalten:

* Dateiname
* Kategorie
* Dateigrösse
* Dateityp
* Dateiinhalt

Die Kategorie soll dabei dynamisch gestaltet werden, daher wird eine zweite Tabelle erstellt, in der die Kategorie referenziert werden kann.

Das nachfolgende Diagramm zeigt die Tabellenimplementierung für die Dokumentenverwaltung:



Abb. 6: Tabellen für Dokumente

### 3.3.3 Detailimplementierung Dokumentenverwaltung

Die Dokumentenverwaltung soll so dynamisch wie möglich gestaltet werden. Um dem MVC Pattern true zu bleiben wurde ein **Filemodel** erstellt, welches sich um die Aufbereitung der Daten kümmert. Sobald der Benutzer eine Datei hochlädt wird dem Filemodel das Superglobal “$\_FILES” mit allen Fileinformationen und die ausgewählte Dateikategorie übergeben.

Das Filemodel bereitet die Informationen auf und übergibt die benötigten Werte an die Funktion add\_document():

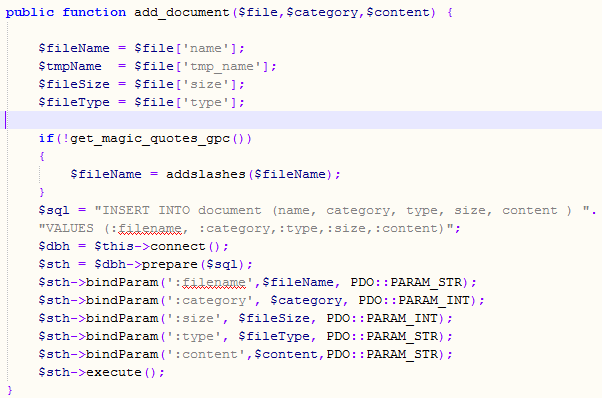


Abb. 7: Funktion add\_document()

Durch den Aufruf von add\_document() wird das Dokument in der Datenbank gespeichert.

Ebenso enthält das Filemodel eine Funktion get\_files, das alle Dateien mit der angegebenen Kategorie zurückliefert:

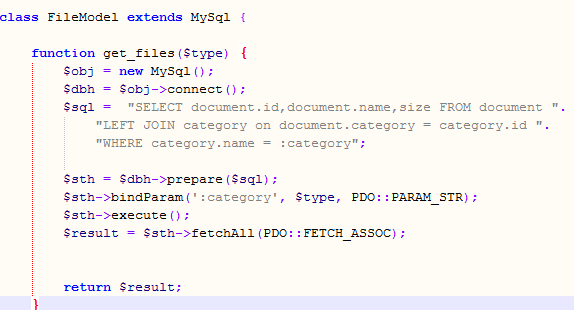


Abb. 8: Funktion get\_files()

Da der Kunde die Dateien auch löschen möchte wird beim Rendern ein Link eingefügt, der auf ein PHP Script verweist, dass das Dokument löscht.

# 4. Anhang

## 4.1 Anhang A: Bilderverzeichnis

[Abb. 1: MVC Modell 4](#_Toc390040055)

[Abb. 2: Objektdiagramm 4](#_Toc390040056)

[Abb. 3: Architekturmodell 5](#_Toc390040057)

[Abb. 4: Beispiel Unterseite 6](#_Toc390040058)

[Abb. 5: Interface "Database" 7](#_Toc390040059)

[Abb. 6: Tabellen für Dokumente 8](#_Toc390040060)

[Abb. 7: Funktion add\_document() 9](#_Toc390040061)

[Abb. 8: Funktion get\_files() 9](#_Toc390040062)