Entwurfsdokument
Homepage “hochsicherheit.ch”

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

Table of Contents

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.

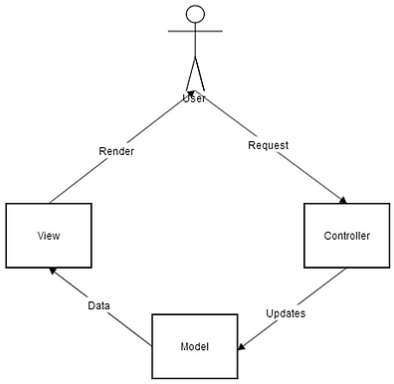

Abb. 1: MVC Modell

2.2 Klassendiagramm

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

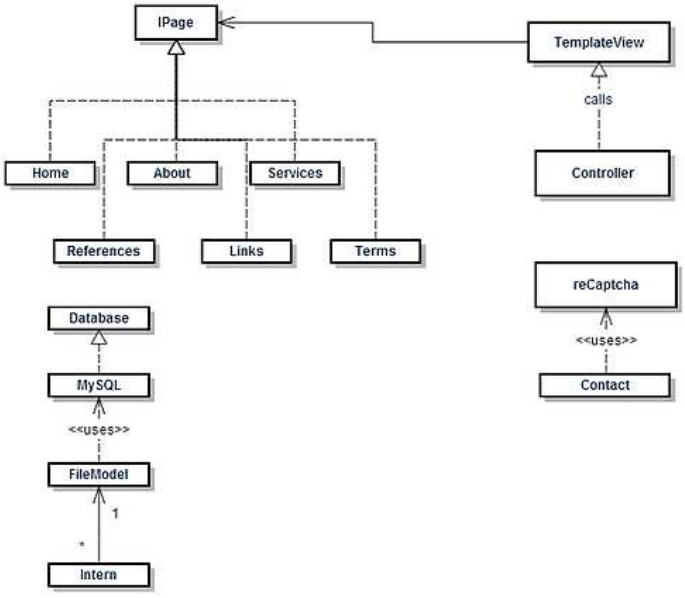

Abb. 2: Objektdiagramm

2.3. Architekturmodell

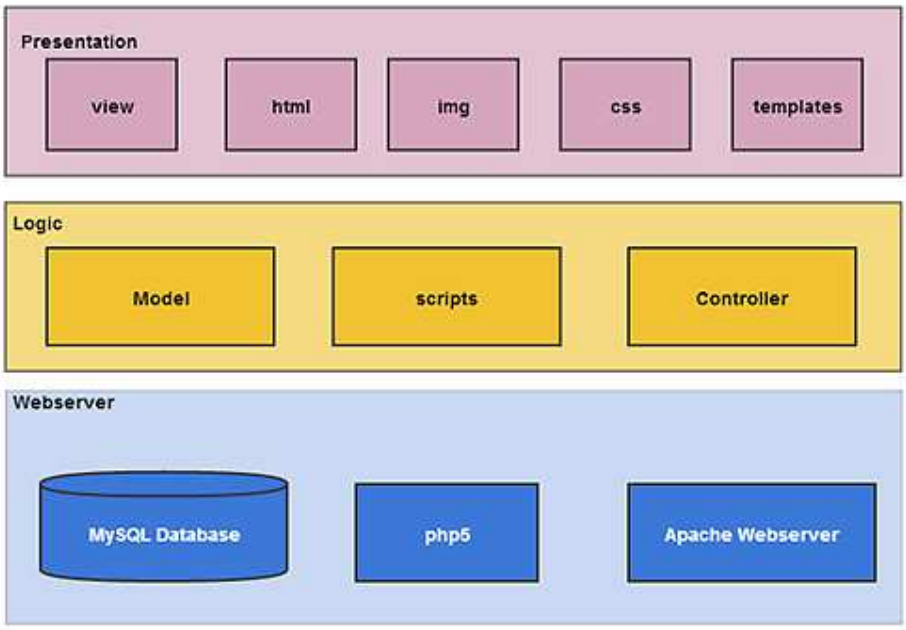

Abb. 3: Architekturmodell

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:

```
<?php
    ini_set('display_errors', 1);
    error_reporting(~0);
    require_once(ROOT_PATH."/inc/homeImpl.php");
    $page = new Home();
?>
<table width="100%">
    <tr>
        <td class="title"><?php echo $page->getPageTitle(); ?> <br /></td>
    </tr>
    <tr>
        <td> </td>
    </tr>
    <tr>
        <td class="maintext" align="left" ><?php echo $page->getPageContent(); ?></td>
    </tr>
</table>
```

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:

```php
<?php
interface Database {

    public function connect();
    public function login($dbh,$user,$pass);
}
?>
```

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():

```php
public function add_document($file,$category,$content) {

    $fileName = $file['name'];
    $tmpName  = $file['tmp_name'];
    $fileSize = $file['size'];
    $fileType = $file['type'];

    if(!get_magic_quotes_gpc())
    {
        $fileName = addslashes($fileName);
    }
    $sql = "INSERT INTO document (name, category, type, size, content ) ".
    "VALUES (:filename, :category,:type,:size,:content)";
    $dbh = $this->connect();
    $sth = $dbh->prepare($sql);
    $sth->bindParam(':filename',$fileName, PDO::PARAM_STR);
    $sth->bindParam(':category', $category, PDO::PARAM_INT);
    $sth->bindParam(':size', $fileSize, PDO::PARAM_INT);
    $sth->bindParam(':type', $fileType, PDO::PARAM_STR);
    $sth->bindParam(':content',$content,PDO::PARAM_STR);
    $sth->execute();
}
```

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:

```php
class FileModel extends MySql {

    function get_files($type) {
        $obj = new MySql();
        $dbh = $obj->connect();
        $sql =  "SELECT document.id,document.name,size FROM document ".
            "LEFT JOIN category on document.category = category.id ".
            "WHERE category.name = :category";

        $sth = $dbh->prepare($sql);
        $sth->bindParam(':category', $type, PDO::PARAM_STR);
        $sth->execute();
        $result = $sth->fetchAll(PDO::FETCH_ASSOC);


        return $result;
    }
```

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