|  |
| --- |
| Modul 151: Projekt Medien Center |
| Planen und Erarbeiten einer Web Applikation mit integrierter Datenbank |
|  |
| **Dueen Brunner, IN15-19C** |
| **15.12.2018** |

|  |
| --- |
|  |

Inhalt

[1 Informieren 3](#_Toc533368970)

[1.1 Auftrag 3](#_Toc533368971)

[1.2 Projektvorgaben 3](#_Toc533368972)

[1.3 Technische Vorgaben 3](#_Toc533368973)

[2 Planen 4](#_Toc533368974)

[2.1 Grundidee Web-Applikation 4](#_Toc533368975)

[2.2 Werkzeuge 4](#_Toc533368976)

[2.2.1 Webserver / Datenbankserver 4](#_Toc533368977)

[2.2.2 IDE 4](#_Toc533368978)

[2.2.3 Code Verwaltung 4](#_Toc533368979)

[2.3 Pflichtenheft 5](#_Toc533368980)

[2.4 Zeitplan 6](#_Toc533368981)

[2.4.1 Meilenstein 1: MVC Grundgerüst mit Datenbank Verbindung 6](#_Toc533368982)

[2.4.2 Meilenstein 2: LoginPage, Datei Upload und Dashboard 6](#_Toc533368983)

[2.4.3 Meilenstein 3: Detailansicht für ein Medium mit Kommentarfunktion 6](#_Toc533368984)

[2.4.4 Abgabe 6](#_Toc533368985)

[3 Entscheiden 7](#_Toc533368986)

[3.1 Template Engine 7](#_Toc533368987)

[3.2 Design 7](#_Toc533368988)

[3.3 JavaScript 7](#_Toc533368989)

[4 Realisierung 8](#_Toc533368990)

[4.1 Klassendiagramm 8](#_Toc533368991)

[4.1.1 Model 8](#_Toc533368992)

[4.1.2 Controller 8](#_Toc533368993)

[4.1.3 View 9](#_Toc533368994)

[4.1.4 Repository Pattern 9](#_Toc533368995)

[4.1.5 Application 10](#_Toc533368996)

[4.2 Programmablaufplan Routing 11](#_Toc533368997)

[4.3 Framework 12](#_Toc533368998)

[4.4 Datenmodell 13](#_Toc533368999)

[4.5 Sicherheit 13](#_Toc533369000)

[4.5.1 Passwort Richtlinien 13](#_Toc533369001)

[4.5.2 Einschleusen von HTML Code 13](#_Toc533369002)

[4.5.3 SQL Injection 14](#_Toc533369003)

[4.5.4 Brute Force Attacken 14](#_Toc533369004)

[5 Kontrolle 15](#_Toc533369005)

[5.1 Überblick Testfälle 15](#_Toc533369006)

[5.2 Testfall Nr.1: Login über Loginpage 16](#_Toc533369007)

[5.2.1 Testschritte 16](#_Toc533369008)

[5.3 Testfall Nr.2: Hochladen von Bilder 17](#_Toc533369009)

[5.3.1 Testschritte 17](#_Toc533369010)

[5.4 Testfall Nr.3: Kommentarfunktion auf Detailansicht 19](#_Toc533369011)

[5.4.1 Testschritte 19](#_Toc533369012)

[6 Fazit 20](#_Toc533369013)

[7 Quellenangabe 20](#_Toc533369014)

# Informieren

## Auftrag

Im Rahmen der LBV Modul 151-1 ist der Lehrling zu einer Projektarbeit verpflichtet, nämlich das Entwerfen und Realisieren einer Web Applikation mit integrierter Datenbank nach einem selbst definierten Pflichtenheft.

Terminiert und abgegeben, mitsamt Dokumentation und Quellcode, wird das Projekt spätestens am **23.Dezember 2018 um 23:55**.

## Projektvorgaben

Die formellen Vorgaben des Projekts wurden durch den Lehrer Alexander Schenkel erteilt und lauten wie folgt:

* Die Projektarbeit wird in Einzelarbeit umgesetzt.
* Sie definieren selber ein Pflichtenheft zu Ihrer Applikation. Dies kann z.B. ein BlogSystem, ein Web-Forum, eine Medien- resp. Bilderverwaltung oder ein RSS-FeedReader sein. Eigene Vorschläge sind willkommen, werden aber mit der Lehrperson abgesprochen.
* Wählen Sie die Applikation nicht zu umfangreich: Das Projekt muss zum Schluss des Semesters abgeschlossen sein. Die Applikation sollte ca. aus 5-10 Haupt-Modellen (z.B. Benutzer, Blog-Eintrag, Kategorien, Tags etc.) bestehen.
* Die Applikation muss mind. verschiedene Benutzer via Login-Mechanismus unterscheiden.
* Sie haben während des Unterrichts jeweils mind. 1 Lektion pro Woche Zeit, an Ihrem Projekt zu arbeiten.
* Sie dürfen alle Informationen verwenden: Unterrichtsunterlagen, Internet. Bei Unklarheiten oder grösseren Problemen wenden Sie sich an die Lehrperson.

## Technische Vorgaben

Neben den formellen Vorgaben des Projekts, sind auch bestimmte technische Vorgaben für die Web-Applikation gegeben:

* Als Server-Programmiersprache wird PHP 7 auf einem Apache Web Server eingesetzt.
* Für das Frontend wählen Sie HTML/CSS/JavaScript.
* Als Datenbank wird eine MySQL-Datenbank eingesetzt.
* Sie setzen kein PHP-Web-Framework ein: Sie entwerfen ein eigenes Web-MVCFramework. Die Grundlagen dazu werden in den ersten Unterrichtslektionen erarbeitet.
* Zusätzliche Bibliotheken oder Paket-Manager (z.B. php composer, Template-Engine) dürfen eingesetzt werden.
* Frontend-Bibliotheken / Frameworks (z.B. Bootstrap CSS Framework) sind erlaubt

# Planen

## Grundidee Web-Applikation

Für die Projektarbeit des Moduls erarbeite ich ein Web-Applikation, bei der sich Benutzer registrieren können und Medien, hauptsächlich Bilder, veröffentlichen können. Es wird ein öffentliches Dashboard zur Verfügung stehen, auf dem alle Benutzer die veröffentlichen Bilder, welche mit einem kurzen Titel und einer Beschreibung versehen werden, einsehen können. Zusätzlich kann ein einzelnes Bild aufgerufen werden, um eine Detailansicht zu öffnen, bei der es auch für die Benutzer möglich ist Kommentare zu hinterlassen.

## Werkzeuge

Für die Realisierung des Projekts werden einige Tools benutzt.

### Webserver / Datenbankserver

Bereits bei früheren Projekten wurde das Programm Xampp benutzt um einen Web- und Datenbankdienst lokal zur Verfügung zu stellen. Auch in diesem Projekt wird erneut auf Xampp zurückgegriffen.

Die Xampp Distribution kommt mit PHP Version 7.2.13 und PhpMyAdmin Version 4.7.0.

### IDE

Als Programmierumgebung wird Microsoft Visual Studio benutzt mit folgenden Plugins:

* PHP Debug
* PHP IntelliSense
* Phpfmt

### Code Verwaltung

Da auf mehreren Geräten codiert wird, muss eine Online Code Verwaltung eingerichtet werden. Für diesen Zweck wird GitKraken zur Hilfe genommen.

## Pflichtenheft

Teil des Projektes ist die Selbstbestimmung eines Pflichtenhefts. Ich habe mir für die gesamte Web-Applikation folgende Punkte zusammengefasst, welche es zu realisieren gibt.

|  |  |  |
| --- | --- | --- |
| Nummer | Funktion | Beschreibung |
| 1 | Login Mechanismus | Ein Benutzer muss in der Lage sein einen persönlichen Account einrichten zu können. |
| 2 | Medien Upload | Ein eingeloggter Benutzer kann ein Bild, dass ein bestimmtes Dateiformat enthält, veröffentlichen. |
| 3 | Dashboard | Auf der Hauptseite wird ein Dashboard gezeigt, welches alle veröffentlichte Bilder in einer bestimmten Darstellungsweise anzeigt. |
| 4 | Persönliche Medien Ansicht | Ein Benutzer kann das Dashboard auf seine eigenen Medien beschränken. |
| 5 | Detailansicht | Es ist möglich ein Medium in einer Detailansicht zu öffnen |
| 6 | Kommentarfunktion | Auf einer Detailansicht können Benutzer Kommentare hinterlassen |

## Zeitplan

Siehe Zeitplan.xlsx

### Meilenstein 1: MVC Grundgerüst mit Datenbank Verbindung

Der Erste Meilenstein ist erreicht, sobald das Routing implementiert wurde, welches den entsprechenden Controller mit der passenden View aufrufen kann. Ausserdem muss die Datenbank Verbindung funktionieren und das Erstellen wird vollautomatisch über ein php Skript durchgeführt.

### Meilenstein 2: LoginPage, Datei Upload und Dashboard

Meilenstein 2 sollte bis zum 07. November erreicht werden. Bis dahin sollte eine Registrierung für den User funktionieren und er sollte sich damit einloggen können. Das daraufhin angezeigte Dashboard zeigt alle veröffentlichten Medien der Benutzer an. Ausserdem kann ein User über ein Formular selber Dateien hochladen, dabei werden akzeptable Daten in der Datenbank gespeichert.

### Meilenstein 3: Detailansicht für ein Medium mit Kommentarfunktion

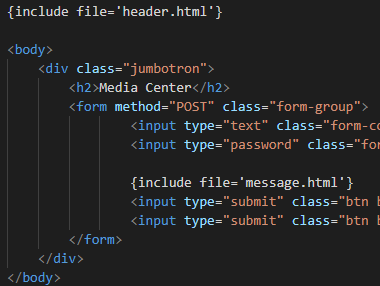
Der dritte und letzte Meilenstein sollte bis Ende November erreicht werden. Über einen Link kann ein Eintrag im Dashboard in einer Einzelansicht geöffnet werden. In der Detailansicht werden alle assoziierten Kommentare abgerufen und angezeigt. Ausserdem können eigene Kommentare verfasst werden unter dem Bild

### Abgabe

Abgegeben wird das Projekt bis spätestens **23. Dezember um 23:55.**

# Entscheiden

## Template Engine

Um die HTML Seiten einfach zusammensetzen zu können und einzelne Ausschnitte wiederverwendbar zu machen muss eine Template Engine eingesetzt werden. Ich habe mich in diesem Fall für Smarty entschieden, da es eine sehr simple Syntax besitzt.

## Design

Da das Design der Web-Applikation eher im Hintergrund steht und nur soweit ausgebaut werden muss, damit das Benutzen möglich ist, wird hier auf normales Bootstrap zurückgegriffen.

## JavaScript

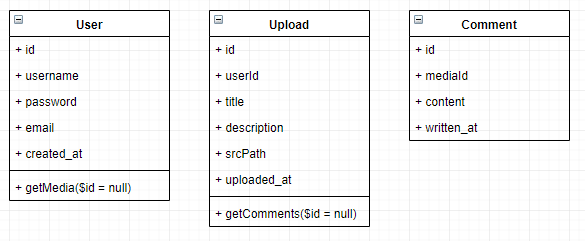
Auf den Einsatz von JavaScript wird in diesem Projekt komplett verzichtet, da es zum einen keine Anforderung ist, und zum anderen nur unnötige Komplexität reinbringt.

# Realisierung

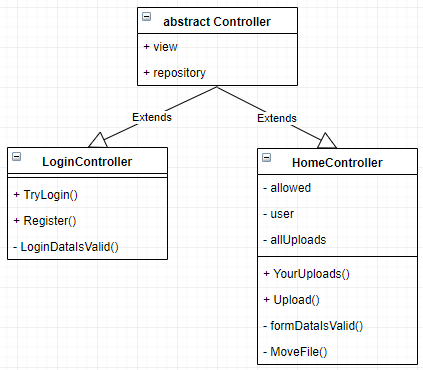
## Klassendiagramm

In diesem Abschnitt werden die eingesetzten Klassen in UML Form dargestellt und anschliessend deren Funktion erklärt.

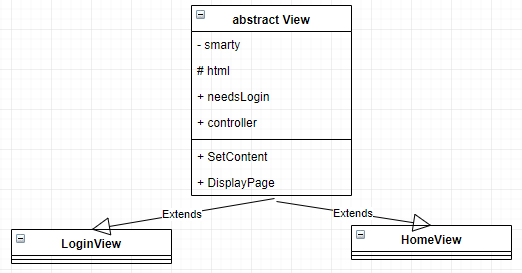
### Model

Diese drei Model Klassen stellen die Datenbank Entities und den Inhalt der Web-Applikation dar.

### Controller

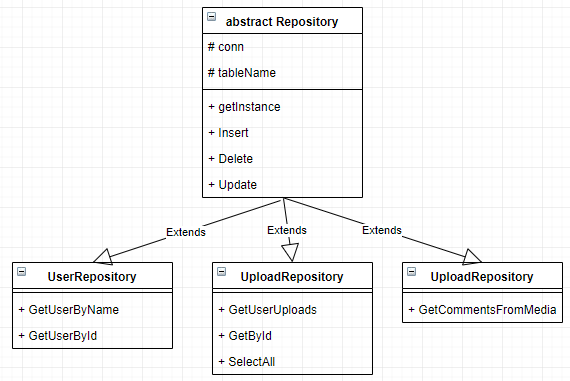
In der Superklasse Controller befinden sich zwei Properties View und Repository welche jeweils im Konstruktor des entsprechenden Controllers definiert werden. Die ableitenden Controller beinhalten spezifische Methoden, welche über den Router abgerufen werden.

### View

Die Superklasse View beinhaltet alle Information für das Front End der Applikation. Über die Smarty Variable wird, zusammen mit den Funktionen SetContent und DisplayPage, die gesamte Anzeige verwaltet. Die html Variable wird im Konstruktor der jeweiligen View definiert. Mit der Property needsLogin kann beim Routing überprüft werden, ob ein User zugreifen darf auf die Seite oder nicht.

### Repository Pattern

Das Repository Pattern übernimmt die ganze Arbeit für Datenbank Zugriffe, damit dies zentral an einem Ort statt findet. Die Superklasse Repository definiert hier die Funktionen Insert, delete und Update, da diese in allen Tables gleich funktionieren. Die Ableitenden Repositories definieren dann erweiterte Datenbank Abfragen.

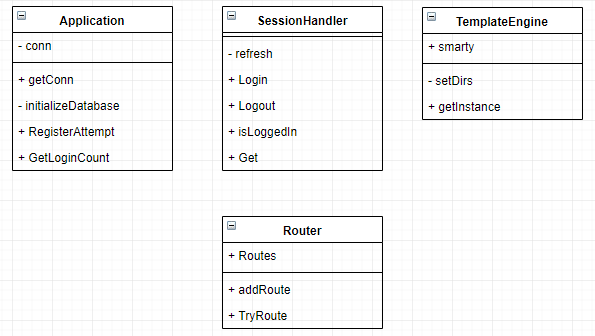


### Application

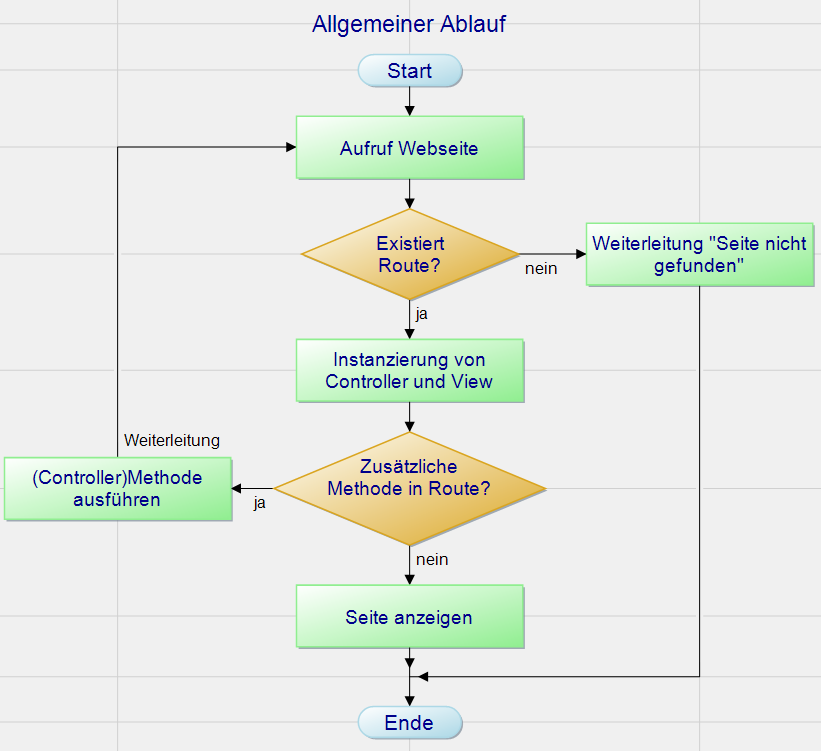
Diese Klassen sind für den Ablauf der Medienseite zuständig. Die Application Klasse stellt die Datenbank Verbindung zur Verfügung und baut die Datenbank auf, falls sie nicht vorhanden ist.

Über die SessionHandler Klase wird der gesamte Prozess der Session abgewickelt, d.h. über ihr wird das Login und Logout (session starten und schliessen) gesteuert, wie auch das holen von Session Variablen abgesichert.

In der TemplateEngine befindet sich das Smarty Objekt, welche alle nötigen Pfade mit setDirs zugewiesen bekommt. Die Singleton Instanz der Template Engine wird allen Views zugewiesen.

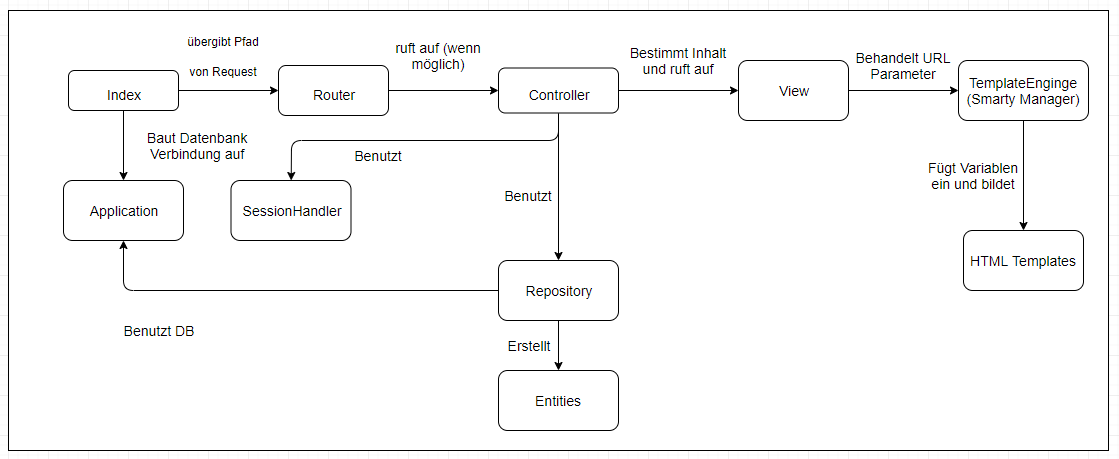
Der Router wird bei Aufruf mit allen möglichen Pfaden gefüllt und anschliessend wird über die abgefragte Route der Controller und die View aufgerufen. Bei einer ungültigen Route wird der Benutzer auf eine Fehlerseite weitergeleitet.

## Programmablaufplan Routing

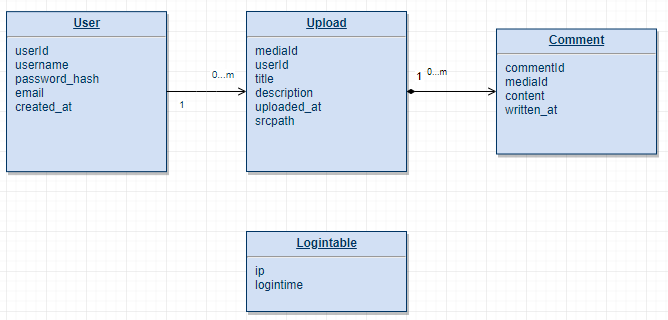
Hier wird ein grober Programmablauf für das Aufrufen der Webseite dargestellt. In der Router Klasse wird ein Dictionary gehandhabt, welches alle möglichen Routen mit der zuständigen Controller Klasse und einer optionalen Methode beinhaltet.



## Framework

In dieser Skizze wird grob beschrieben wie der gesamte Ablauf vom Request bis zur Darstellung des Front Ends durchläuft.

## Datenmodell

Das Datenbank Modell besteht aus insgesamt vier Tables. Die Tabellen User, Upload und Comment speichern die Entities des Projekts ab. Die Tabelle Logintable speichert Loginversuche mit der IP Adresse des Sendegeräts und der Zugriffszeit.

## Sicherheit

Im Unterricht wurden diverse Sicherheitsprobleme besprochen und erläutert, wie diese verhindert werden können. Dieses Kapitel befasst sich mit den Sicherheitsproblemen, die in dem Projekt abgefangen wurden.

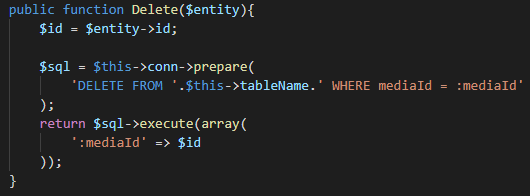
### Passwort Richtlinien

Bei Registrierung muss ein Passwort mindestens 8 Zeichen, davon ein Grossbuchstabe und ein Sonderzeichen enthalten. Bei Nichterfüllung wird der User auf die Loginseite mit entsprechender Meldung weitergeleitet.

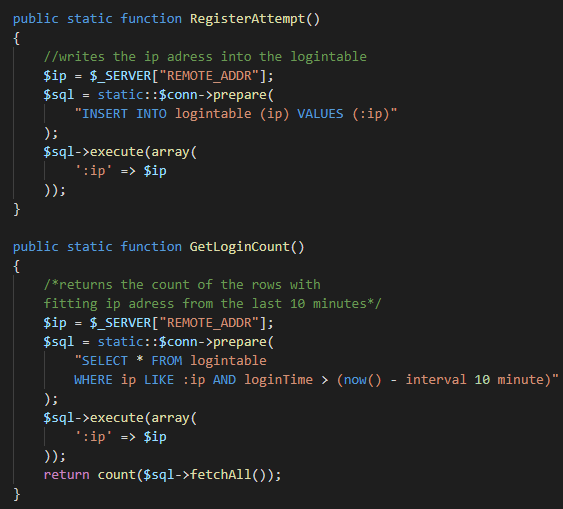
### Einschleusen von HTML Code

Das Einschleusen von schädlichem HTML Code über Input Felder ist generell in der Web-Applikation nicht möglich. Jedes Input Feld, dass Text übermittelt wird bei Aufnahme mit htmlspecialchars() bearbeitet. Werte, welche HTML Zeichen beinhalten können zwar in der Datenbank abgespeichert werden, jedoch werden sie bei der Ausgabe auf der Webpage von der Smarty Engine bearbeitet und nicht als HTML Code behandelt.

### SQL Injection

Durch das Verwenden von Prepared statements in den Repository Klassen, wird eine SQL Injection verhindert, weil alle Parameter zwischen Gänsefüsschen gesetzt werden und somit als ganzer String eingefügt werden in die Query.

### Brute Force Attacken

Die Applikation fügt bei jedem Login Fehlversuch einen Datensatz in die Login Tabelle ein. Bei jedem erneuten Login Versuch werden die Anzahl Datensätze vom Sendergerät mit der IP überprüft und sollten es drei oder mehr sein, dann wird jeder neue Versuch abgelehnt. Nach zehn Minuten ist ein Login Versuch nicht mehr relevant.

# Kontrolle

Dieses Kapitel befasst sich mit dem Testen der Applikation. Für das Durchführen der Testfälle wurde die Vorlage aus dem Unterricht benutzt. Es werden drei Hauptfunktionen des Programms getestet, nämlich folgende:

* Login Mechanismus
* Bilder Upload
* Kommentar unter einem Bild posten

## Testfall Nr.1: Login über Loginpage

|  |  |
| --- | --- |
| Use-Case/Szeario: | Login über Startseite der Web-App |
| Beschreibung: | Testen des Formulars, ob die Daten richtig validiert werden und das Login funktioniert. |
| Vorbedingungen: | - |
| Datum: | 22.12.2018 |
| Tester: | Dueen Brunner |
| Erfolg: | ja, vollständig |

### Testschritte

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Nr. | 1 | Test-Erfolg: | OK |
| Titel / Kurzbeschreibung | Login mit einem bestehenden Benutzer über das Formular | | |
| Vorbedingung, Eingabeparameter | Ein User muss vorher über Registrierung erstellt werden, damit dieser auf Login überprüft werden kann | | |
| Testablauf | 1. Aufruf „localhost“, Anzeige der Login / Register Page 2. Eingabe von Benutzername und Passwort des bestehenden Users 3. Absenden der Daten über Knopfdruck auf Login | | |
| erwartetes Ergebnis | Weiterleitung auf localhost/home, Der User sollte ein Dashboard sehen | | |
| tatsächliches Ergebnis | User wird weitergeleitet und kann Dashboard einsehen | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Nr. | 2 | Test-Erfolg: | OK |
| Titel / Kurzbeschreibung | Login mit falschem Passwort | | |
| Vorbedingung, Eingabeparameter | Ein User muss vorher über Registrierung erstellt werden, damit dieser auf Login überprüft werden kann | | |
| Testablauf | 1. Aufruf „localhost“, Anzeige der Login / Register Page 2. Eingabe von Benutzername und einem falschen Passwort 3. Absenden der Daten über Knopfdruck auf Login | | |
| erwartetes Ergebnis | Die Applikation bleibt bei der LoginPage und gibt eine Meldung aus, dass das Login fehlschlug | | |
| tatsächliches Ergebnis | Die Applikation bleibt bei der LoginPage und gibt eine Meldung aus, dass das Login fehlschlug | | |

## Testfall Nr.2: Hochladen von Bilder

|  |  |
| --- | --- |
| Use-Case/Szeario: | Hochladen eines Bildes auf der Dashboard Page |
| Beschreibung: | Testen des Upload Formulars, ob die Daten richtig validiert werden und das Bild gespeichert wird |
| Vorbedingungen: | Der Benutzer muss eingeloggt sein und ein Bild besitzen |
| Datum: | 22.12.2018 |
| Tester: | Dueen Brunner |
| Erfolg: | ja, vollständig |

### Testschritte

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Nr. | 1 | Test-Erfolg: | OK |
| Titel / Kurzbeschreibung | Hochladen eines Bildes auf dem Upload Formular | | |
| Vorbedingung, Eingabeparameter | Der Benutzer muss eingeloggt sein und ein Bild besitzen | | |
| Testablauf | 1. Auswählen des Bildes 2. Eingabe von Titel und Beschreibung zum Bild 3. Formular absenden über Upload Knopf | | |
| erwartetes Ergebnis | Es wird erneut das Dashboard abgerufen. Zuoberst sollte das hochgeladene Bild erscheinen | | |
| tatsächliches Ergebnis | Es wird erneut das Dashboard abgerufen. Zuoberst erscheint das ausgewählte Bild | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Nr. | 2 | Test-Erfolg: | OK |
| Titel / Kurzbeschreibung | Hochladen eines Bildes grösser > 2MB | | |
| Vorbedingung, Eingabeparameter | Der Benutzer muss eingeloggt sein und ein Bild besitzen, dass grösser als 2MB ist. | | |
| Testablauf | 1. Auswählen des zu grossen Bildes 2. Eingabe von Titel und Beschreibung zum Bild 3. Formular absenden über Upload Knopf | | |
| erwartetes Ergebnis | Es wird erneut das Dashboard abgerufen, es bleibt jedoch unverändert. Unter dem Formular sollte eine Meldung bezüglich der Dateigrösse erscheinen. | | |
| tatsächliches Ergebnis | Es wird erneut das Dashboard abgerufen, es bleibt jedoch unverändert. Unter dem Formular erscheint eine Meldung bezüglich der Dateigrösse. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Nr. | 3 | Test-Erfolg: | OK |
| Titel / Kurzbeschreibung | Hochladen einer Datei, die kein Bild ist | | |
| Vorbedingung, Eingabeparameter | Der Benutzer muss eingeloggt sein und eine Datei besitzen, die kein Bild (jpg, png oder gif) ist. | | |
| Testablauf | 1. Auswählen eines Nicht-Bildes (in diesem Beispiel wurde ein PDF genommen) 2. Eingabe von Titel und Beschreibung 3. Absenden über Upload Knopf | | |
| erwartetes Ergebnis | Es wird erneut das Dashboard abgerufen, es bleibt jedoch unverändert. Unter dem Formular sollte eine Meldung bezüglich des nicht unterstützten Dateiformats erscheinen. | | |
| tatsächliches Ergebnis | Es wird erneut das Dashboard abgerufen, es bleibt jedoch unverändert. Unter dem Formular erscheint eine Meldung bezüglich des Dateiformats. | | |

## Testfall Nr.3: Kommentarfunktion auf Detailansicht

|  |  |
| --- | --- |
| Use-Case/Szeario: | Hinterlassen eines Kommentars in der Detailansicht eines Eintrags |
| Beschreibung: | Testen der Kommentar Funktion in der Detailansicht eines Medien Uploads |
| Vorbedingungen: | Der Benutzer muss eingeloggt und es muss mindestens ein Eintrag bestehen |
| Datum: | 22.12.2018 |
| Tester: | Dueen Brunner |
| Erfolg: | ja, vollständig |

### Testschritte

|  |  |  |  |
| --- | --- | --- | --- |
| ID/Nr. | 1 | Test-Erfolg: | OK |
| Titel / Kurzbeschreibung | Posten eines Kommentars unter einem Medien Upload | | |
| Vorbedingung, Eingabeparameter | Der Benutzer muss eingeloggt und es muss mindestens ein Eintrag bestehen. | | |
| Testablauf | 1. Detailansicht eines Uploads aufrufen 2. In der Detailansicht in das Kommentarfeld Text eingeben 3. Posten des Kommentars über Post Knopf | | |
| erwartetes Ergebnis | Es wird erneut die Detailansicht des Mediums aufgerufen. Unter dem Bild sollte ein nicht beschreibbares Feld mit dem abgesendetem Inhalt erscheinen. | | |
| tatsächliches Ergebnis | Es wird erneut die Detailansicht des Mediums aufgerufen. Unter dem Bild erscheint ein nicht beschreibbares Feld mit dem abgesendetem Inhalt, | | |

# Schlusswort

Das letzte Kapitel dieses Dokumentes beinhaltet eine Reflexion meinerseits über das Modul 151 und das erarbeitete Projekt.

## Reflexion Projektarbeit Medien Center

Zuerst würde ich mit einer Stellungnahme zum Endprodukt meiner Arbeit beginnen. Das Realisieren einer Web-Applikation, beginnend beim Erarbeiten eines einfachen Konzepts, über das Codieren des gesamten Frameworks, bis hin zum Testen hat mir grundsätzlich gefallen und mir vor einen grossen Einblick in die Funktionsweise der untypisierten Skriptsprache PHP gegeben.   
Dass ich leider nicht zeitgerecht alle Funktionen aus meinem Pflichtenheft für das Medien Center erfüllen konnte, finde ich schade und hat mir einmal mehr gezeigt, dass ein Projekt dieses Ausmasses viel Arbeit und Zeit auf sich nimmt, welche ich gerade zu Beginn nicht investiert habe. Eventuell werde ich mit den fehlenden Funktionen, nämlich das Bearbeiten und Löschen von Medien Einträgen und Kommentaren, in nächster Zeit noch vervollständigen, damit ich auch zufrieden sein kann mit meiner Arbeit.

Die Programmiersprache PHP hat mich während des Semesters viel Geduld gekostet, welches zum einen mit dem Syntax, welcher für mich als Desktopentwickler in C# sehr fremd wirkte, und zum anderen mit dem schwierigen Debuggen der Webseite zu hat, was es mir manchmal schwer machte weiterzukommen. Aus diesem Grund mag wahrscheinlich mein Code und insgesamt mein Projekt etwas zu spärlich ausfallen.

Dass während den Lektion selten Zeit für das Arbeiten an der Web-Applikation finde ich schade und hat meinen Zeitplan dann verschoben.