Arbeitsjournal

Inhalt

[Donnerstag 5. Mai 2016 3](#_Toc451426303)

[Sonntag 8. Mai 2016 4](#_Toc451426309)

[Donnerstag 12. Mai 2016 5](#_Toc451426315)

[Sonntag 15. Mai 2016 6](#_Toc451426321)

[Mittwoch 18. Mai 2016 7](#_Toc451426327)

# Donnerstag 5. Mai 2016

### Tätigkeiten

|  |  |  |
| --- | --- | --- |
| Tätigkeiten | Zeitaufwand | |
| Soll | Ist |
| MVC-Pattern von Blog Projekt anpassen | 1h | 0.5h |
| Datenbank Entwurf erstellen | 0.5h | 0.6h |

### Arbeiten

Ich habe in der ersten Lektion des neuen Projektes mein MVC-Pattern, welches ich schon beim Blog-Projekt erstellt habe auf die neue Aufgabenstellung angepasst. Zudem habe ich geschaut, welche Teile des Blogs ich auch für dieses Projekt verwenden könnte.

Ich habe auch noch einen Datenbank-Entwurf für das Projekt gemacht und anhand der Spezifikation verschiedene Tabellen erstellt.

### Hilfestellungen

* [www.draw.io](http://www.draw.io) zum Erstellen des Datenbankdiagramm

### Reflexion

Ich konnte das erreichen was ich wollte. Bis jetzt funktioniert alles.

### Nächste Schritte

1. Model Klassen auf die Datenbank anpassen
2. Erster Prototyp erstellen

# Sonntag 8. Mai 2016

### Tätigkeiten

|  |  |  |
| --- | --- | --- |
| Tätigkeiten | Zeitaufwand | |
| Soll | Ist |
| Datenbank Schnittstelle (Model) neu entwerfen | 2h | 2h |
| Grunddesign erstellen | 2h | 2.5h |

### Arbeiten

Nach dem ich die Datenbank gehabt habe, habe angefangen die Model Klassen zu schreiben, um die ersten Zugriffe auf die Datenbank zu machen.

Neu brauche ich für das Design nicht mehr Bootstrap sondern Materializecss ([www.materializecss.com](http://www.materializecss.com)). Das ist auch ein Front-End Framework, welches Bootstrap extrem ähnlich ist und einfach das Design von [Google’s Material Design](https://www.google.com/design/spec/material-design/introduction.html) einbringt. Die Funktion sind fast exakt die gleichen wie bei Bootstrap.

### Hilfestellungen

* Materialize CSS Dokumentation
* Blog Projekt

### Reflexion

Ich kann jetzt die ersten Statements gegen die Datenbank ausführen und bin soweit gut im Zeitplan, da ich jetzt die Login Funktionalität implementieren kann. Das Design steht soweit auch schon.

### Nächste Schritte

1. Login und Registration implementieren
2. Verbleibende Model-Klassen schreiben

# Donnerstag 12. Mai 2016

### Tätigkeiten

|  |  |  |
| --- | --- | --- |
| Tätigkeiten | Zeitaufwand | |
| Soll | Ist |
| Arbeitsjournal schreiben | 0.5h | 0.7h |
| User-Model fertigstellen | 1h | 0.8h |

### Arbeiten

Ich habe am heutigen Donnerstag noch das Arbeitsjournal vom Wochenende nachgeschrieben

Im Rest der Lektion habe ich noch die Usermodel-Klasse fertiggeschrieben. Da ich in meinem Blog-Projekt aus Faulheit die Datenbank nicht vor SQL-Injection geschützt habe musste ich noch recht viel neuen Code schreiben. Jetzt ist die Datenbank durch prepared-Statements geschützt.

### Hilfestellungen

* Internet

### Reflexion

Ich konnte heute endlich die Usermodel Klasse fertig schreiben und daher kann ich das nächste Mal mit dem Login und der Registration starten.

### Nächste Schritte

* Login
* Registration

# Sonntag 15. Mai 2016

### Tätigkeiten

|  |  |  |
| --- | --- | --- |
| Tätigkeiten | Zeitaufwand | |
| Soll | Ist |
| Login Formular erstellen und Logik programmieren | 2h | 1.5h |
| Registration implementiert | 1h | 1h |

### Arbeiten

Da das Login ja grundsätzlich immer gleich abläuft konnte ich den LoginController vom Blog Projekt übernehmen. Im folgenden Text zitiere ich das Arbeitsjournal von meinem Blog Projekt.

Zuerst habe ich eine LoginController erstellt und drei verschiedene Actions (Methoden) geschrieben (index, login, add). Index 🡪 Anzeige des Formulars; login 🡪 Logik zum Einloggen; add 🡪 Registrationslogik

Als ich das Design des Formulars fertiggestellt habe, habe ich angefangen mit der Registration. Innerhalb einer Stunde konnte man sich registrieren und die Eingaben wurden korrekt validiert.  
Das Passwort wird Serverseitig mit sha512 verschleiert.

Als nächstes machte ich das Login möglich. Das ging auch nicht mehr so lange. Dafür implementierte ich auf dem UserModel eine neue Methode, welche die E-Mail und das Passwort des Objektes mit der Email und dem Passwort des übergebenen Objektes überprüft.

Jedoch haben sich dennoch ein paar kleine Sachen geändert. Das Passwort jetzt nicht mehr wie vorher mit der hash()-Funktion gehasht sondern jetzt mit password\_hash(). Dazu gehört auch das überprüfen mit password\_verify(). Die Methoden, welche zum Vergleichen von Benutzer und Passwort haben sich minim geändert.

### Hilfestellungen

* Blog Projekt
* Internet

### Reflexion

Ich kann jetzt Benutzer erstellen und einloggen und habe damit den ersten Meilenstein erreicht. Ich jetzt endlich damit anfangen Galerien zu erstellen und ein grundlegendes Design für den internen Bereich anzufertigen.

### Nächste Schritte

* Galerien erstellen
* Bilder zu den Galerien hinzufügen

# Mittwoch 18. Mai 2016

### Tätigkeiten

|  |  |  |
| --- | --- | --- |
| Tätigkeiten | Zeitaufwand | |
| Soll | Ist |
| Autoloader und Namespaces | 3h | 2h |

### Arbeiten

Um das „Classloading“ zu vereinfachen arbeite ich in PHP mit Namespaces und einem Autoloader. Der Autoloader ist dafür da, dass ich sobald ich eine bestimmte Klasse benötige nicht selber das File „includen“ oder „requiren“ muss, sondern dass er das für mich automatisch macht.

Ich habe jetzt ein PHP-File „autoloader.php“, welches ein class-Array enthält wo alle Klassen als Key drin sind und der dazugehörige Wert dem Pfad zum File wo sich die Klasse drin befindet entspricht.

In der Funktion \_\_autoload() wird jetzt immer geschaut ob die Klasse als Key besteht, wenn ja dann wird das File geladen sonst wird eine Exception geworfen.

### Hilfestellungen

* Internet
* Joel Messerli

### Reflexion

Ich habe mit dem Teil der Arbeit mit mehr Arbeitszeit gerechnet, jedoch kam ich sehr gut und schnell voran und konnte so gut einsparen.

### Nächste Schritte

* Galerien erstellen
* Bilder zu den Galerien hinzufügen

# Donnerstag 19. Mai 2016

## Tätigkeiten

|  |  |  |
| --- | --- | --- |
| Tätigkeiten | Zeitaufwand | |
| Soll | Ist |
| Gallery Model erstellen | 0.8h | 0.7h |
| Formular für Galerieerstellung | 0.4h | 0.4h |
| Troubleshooting bei Namespaces | 0.3h | 0.5h |

### Arbeiten

Heute habe ich das Gallery Model soweit fertiggestellt, dass ich schon komplette Objekte erzeugen kann und ein Gallery-Objekt in der Datenbank speichern kann.  
Zudem startete ich damit ein Formular für die Galerie Erstellung auf der Hauptseite zu machen.

Da ich beim letzten Arbeitsschritt etwas vergessen habe zu implementieren, musste ich das noch nachholen. Genauer gesagt habe ich vergessen einige Klassen dem richtigen Namespace zuzuordnen und daher konnte der Autoloader die Klassen nicht finden und laden.

### Hilfestellungen

* Internet

### Reflexion

Heute ist es gut gelaufen und ich konnte noch einen Fehler finden und ihn beheben. Dank dem ich den Fehler noch jetzt gefunden habe konnte ich ihn auch dementsprechend schnell beheben und werde später keine Probleme deswegen mehr haben.

### Nächste Schritte

* Galerie Formular fertigstellen
* Galerieerstellung im Back End ermöglichen

# Samstag 21. Mai 2016

## Tätigkeiten

|  |  |  |
| --- | --- | --- |
| Tätigkeiten | Zeitaufwand | |
| Soll | Ist |
| Galerie Übersicht | 1.4h | 1.5h |
| http-Error Seiten | 1h | 0.8h |
| Loginseite ist nicht mehr verfügbar wenn eingeloggt | 0.5h | 2h |
| Galerien löschen | 1h | 1.2h |

### Arbeiten

Ich kann jetzt auf der Hauptseite (nur wenn eingeloggt) alle persönlichen Galerien anzeigen. Später sollte man auch noch Galerien von anderen Usern anzeigen lassen können einfach unter einer anderen Überschrift.

Ausserdem kann ich jetzt von meinen Controllern aus extra einen anderen Controller aufrufen, welcher dann ein spezielles Bild mit einem http-Status Code bzw. Error Code anzeigt.

Wenn man als User eingeloggt ist kann man jetzt nicht mehr auf die Loginseite zugreifen, damit man als eingeloggter User keine anderen Accounts erstellen kann oder sich ohne auszuloggen mit einem anderen Benutzer anmelden kann.

Meine letzte Arbeit am heutigen Tag war, dass ich eigene Galerien löschen kann. Das ging relativ schnell, da fast alle Grundbausteine bestanden.

### Hilfestellungen

* Internet

### Reflexion

Ich habe widererwarten für die dritte Tätigkeit viel länger gebraucht. Das lag vor allem daran, dass ich mehrere Methoden versucht habe aber die ersten haben mir rein «Design»-technisch nicht gefallen und ich wollte mich nicht damit zufrieden geben.

### Nächste Schritte

* Galerie Ansicht machen
* Fotos zu einer Galerie hinzufügen
* Thumbnails für Galerien

# Sonntag 26. Mai 2016

## Tätigkeiten

|  |  |  |
| --- | --- | --- |
| Tätigkeiten | Zeitaufwand | |
| Soll | Ist |
| Fotos zu einer Galerie hinzufügen | 2h | 2.3h |
| Tags zu einem Foto hinzufügen | 1h | 1.5h |
| Fotos skalieren auf Thumbnailgrösse | 1h | 1.3h |

### Arbeiten

Am heutigen habe ich das Image-, das Tag- und das ImageTag-Model implementiert. Somit kann ich jetzt Fotos heraufladen und einer Galerie hinzufügen und den Fotos eigene Tags anhängen. Es wird in der Datenbank ein neues Tag erstellt, wenn einem Bild ein Tag angehängt wird, welches noch nicht besteht und sonst wird das schon bestehende tag benutzt.

Die Fotos werden jetzt auch auf dem Server skaliert und in einem separaten Thumbnailordner abgespeichert.

### Hilfestellungen

* Internet

### Reflexion

Ich kam grundsätzlich relativ gut vorwärts. Teilweise hatte ich noch ein bisschen mühe beim richtigen Umsetzen des geplanten und machte auch ein paar unnötige Umwege um an das Ziel zu kommen.

### Nächste Schritte

* Thumbnails in der Galerie anzeigen.

# Samstag 4. Mai 2016

## Tätigkeiten

|  |  |  |
| --- | --- | --- |
| Tätigkeiten | Zeitaufwand | |
| Soll | Ist |
| Fotos Gesamtansicht | 0.8h | 0.9h |
| Fotos Einzelansicht | 1.5h | 1.3h |
| Fehler-/Erfolgsmeldungen | 1.5h | 1.6h |
| Suche nach Galerie | 1h | 1.3h |
| Suche nach Bild | 2h | 2h |

### Arbeiten

Ich konnte heute sehr vieles erledigen und fertigstellen. Zu diesen Sachen gehören die Gesamtansicht der Bilder in einer Galerie. Dort kann ich auch die Bilder hochladen und, dann werden sie auf eine Höhe von 200px skaliert, da bei 100px die Bilder fast verschwindend klein waren.

Wenn man auf ein Bild klickt kommt man auf die Einzelansicht und dort sieht man die Tags und kann zudem zu dem nächsten und letzten Bild wechseln.

Ich habe noch Flash Funktionen eingefügt, damit ich per GET-Parameter Nachrichten auf der Seite anzeigen kann. Die Nachricht kann sich in folgenden Kategorien unterscheiden: error, warning, success, info.

In der Galerie Übersicht kann man jetzt nach seinen eigenen Galerien suchen und in der Galerie Einzelansicht ist es möglich nach Bildern zu suchen mit einem gewissen Tag.

### Hilfestellungen

* Internet
* PHP-Doc
* Stackoverflow
* Joel Messerli’s Blog-Projekt (Flash)

### Reflexion

Ich konnte heute sehr viele neue Sachen hinzufügen, musste jedoch aber auch recht viel Zeit dafür aufwenden. Ich bin jetzt ziemlich am Ende des Projektes und es geht jetzt vor allem noch darum manuelle Tests durchzuführen und zu überprüfen, ob alles so funktioniert, wie erwartet.

### Nächste Schritte

* Testen
* Bugs beheben
* Je nach dem noch Zusatzfunktionen hinzufügen