Exposee

Neingag ist die Inverse von 9gag.  
 Während sie ständig von witzigen Medien bombardiert werden, sorgt Neingag dafür,  
dass sie sich nicht „totlachen“ können.

Neingag

Projekt für serverseitige Anwendungen

Moritz Vetter

Inf2557@hs-worms.de

Matrikelnr. 671536

Björn Ammon

inf2517@hs-worms.de

Matrikelnr. 671303

Inhaltsverzeichnis

1. Einleitung2

2. Ideen3

2.1 Raumbuchung3

2.2 Neingag4

3. Vorbereitung5

4. Realisierung6

4.1 Datenbank6

4.2 Registration7

4.3 Login8

4.4 Upload9

4.5 Anzeige11

4.6 Kommentieren / Votes13

5. Zusammenfassung13

6. Ausblick14

7. Quellenverzeichnis15

# 1. Einleitung

Für das Modul „Serverseitige Anwendungen“ soll eine Anwendung erstellt werden, die mit einer Datenbank kommuniziert. Hierbei steht die Funktionalität im Vordergrund, Design ist nicht ausschlaggebend.

Diverse Funktionalitäten wie Login und verschlüsselte Passworteinträge in der Datenbank sind Voraussetzungen für ein anständiges Projekt.

# **2. Ideen**

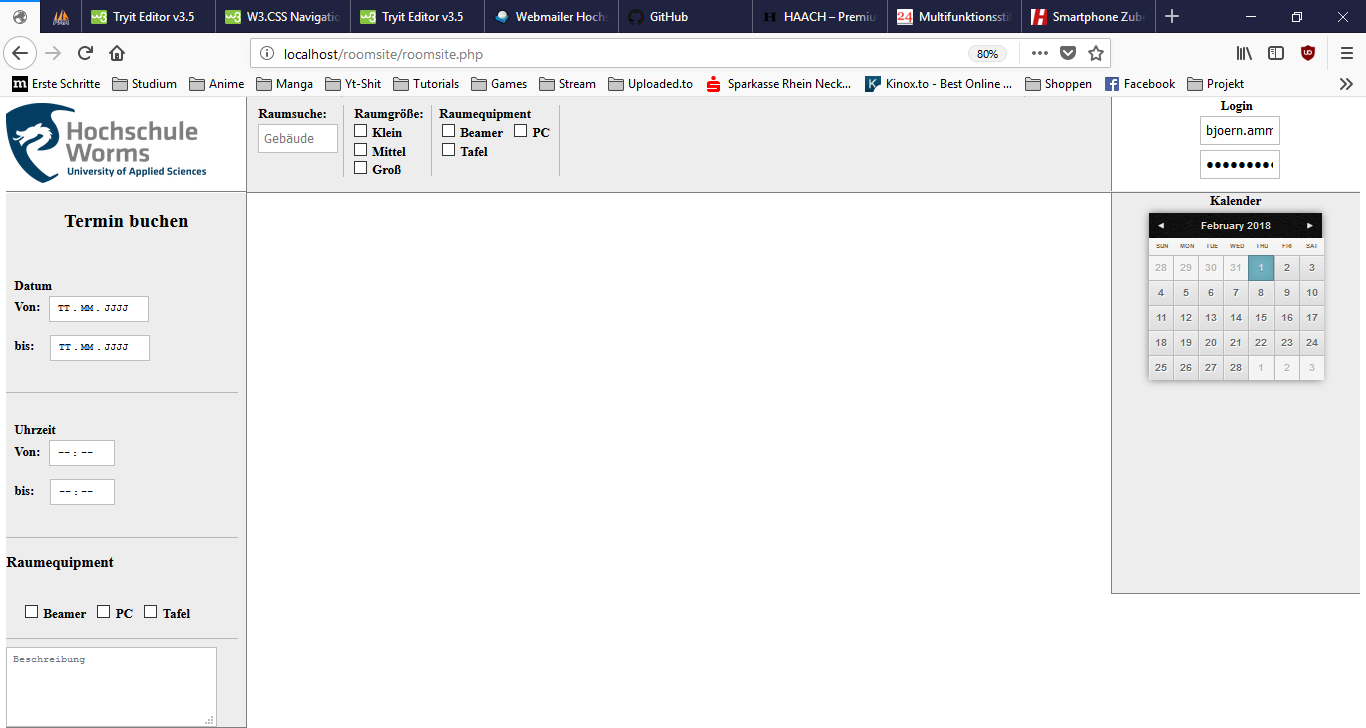
# 2.1 Raumbuchung

Die erste Idee war eine Seite zur Raumbuchung an der Hochschule Worms.   
Hier hätte man die Möglichkeit von einem Account aus, einen Raum zu buchen. Studenten der Hochschule Worms hätten hier ihren Login des Rechenzentrums verwendet, also z. B. inf2517@hs-worms.de. Organisationen sollten einen extra Account erstellen, z. B. veranstaltungen-eichbaum@gmx.de als E-Mail-Adresse. Informationen wären dann an die angegebene Adresse gesendet worden.

Zum Finden eines Raumes gibt es eine Suchmaske, welche verschiedene Optionen bietet, wie Gebäudenummer, Raumnummer, Raumgröße oder auch vorhandenes Equipment (Beamer etc.). Nach Eingabe der Gebäudenummer, wäre im Zentrum der Seite eine Blaupause des Gebäudes angezeigt worden, worüber man auch die einzelnen Räume auswählen und anzeigen lassen könnte. Nach Auswahl des Raumes werden Größe, Ausstattung und eventuell vorhandene Reservierungen angezeigt.

Um eine Buchung zu beantragen, müsste der Nutzer einen Zeitraum angeben, diese Buchung wird dann abgeschickt und kann von der Administration genehmigt werden.

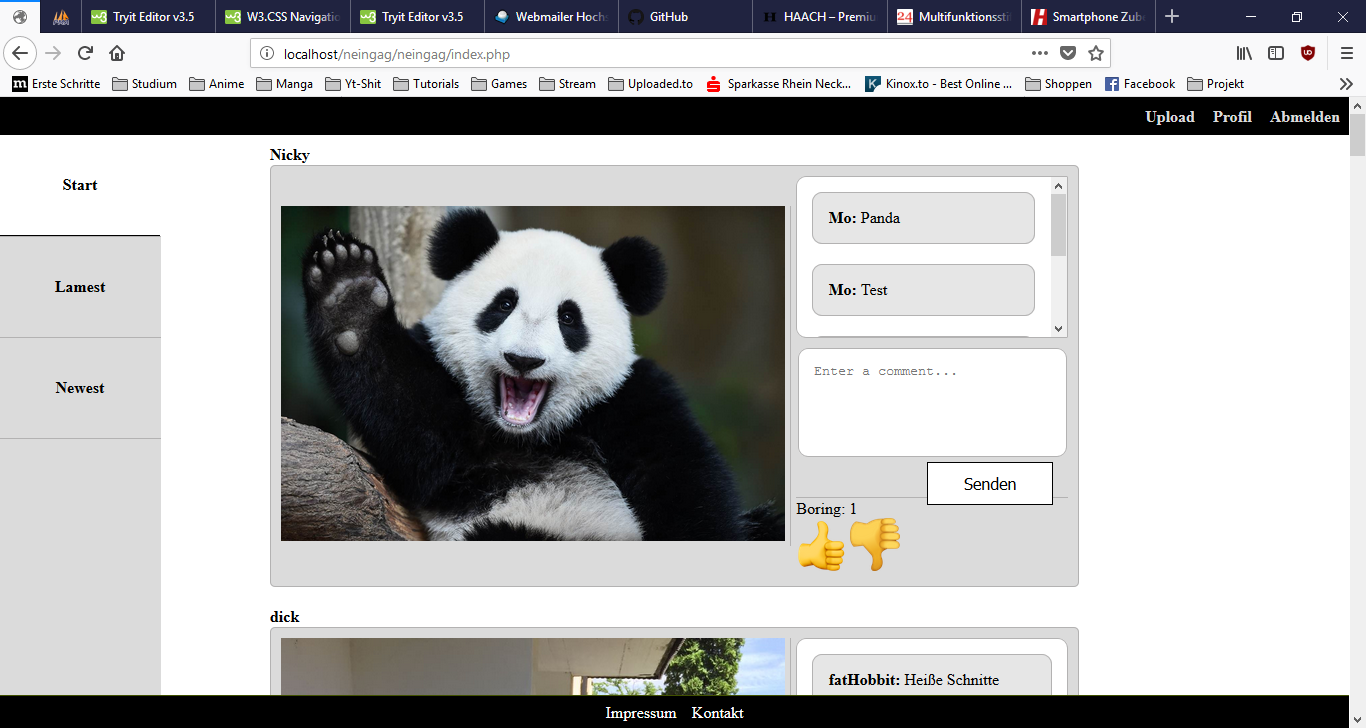
Da diese Idee unserer Meinung nach zu simpel war, haben wir sie verworfen. Diverse Funktionalitäten wie Login, Raumsuche und Raumbuchung waren bereits gegeben.



# 2.2 Neingag

Unsere nächste Idee war es dann eine Seite zum betrachten und bewerten von Bildern, welche von den Nutzern selbst hochgeladen werden, zu erstellen.

Da es solche Formate schon gibt und diese immer den Zweck der Unterhaltung in den Vordergrund stellen, nahmen wir den alternativen Ansatz eine Seite zu kreieren welche nur für die langweiligsten Bilder gedacht ist – Neingag!  
Auch hier gibt es eine Registration und das dazugehörige Login. Eingeloggte Nutzer können sämtliche Funktionen der Seite nutzen während für alle anderen nur die Betrachtung der Bilder ermöglicht wird.

Die Funktionen der Seite beinhalten natürlich das Hochladen von Bildern, welche direkt auf der Seite angezeigt werden. Das Herzstück der Seite ist die Anzeige der Bilder aller Nutzer. Hierfür gibt es verschiedene Anzeigemöglichkeiten: Start zeigt die Bilder in zufälliger Reihenfolge an, Lamest sortiert die Bilder nach Bewertung, wobei eine hohe Bewertung für ein langweiligeres Bild steht sowie Newest, welches die Bilder in der Reihenfolge des Uploads hochlädt, angefangen mit den neuesten Bildern. Bei allen Sortierungen wird die Anzeige der eigenen Bilder ausgelassen, sodass einem Nutzer jeweils nur die Bilder der anderen Nutzer gezeigt werden.

Außerdem gibt es die Option Bilder zu bewerten und zu kommentieren, sowie die Kommentare der anderen Nutzer anzusehen.

Im persönlichen Bereich eines Nutzers kann man die eigenen Profildaten bearbeiten oder auch das gesamte Profil löschen. Beim löschen eines Nutzers werden alle seine Bilder aus der Datenbank entfernt und im File System archiviert.

# 3. Vorbereitung

Zur Implementierung der Seite haben wir uns als Sprache PHP ausgesucht, dies war für uns dank der Einführung von Herrn Mahler und der hilfreichen Seite php-einfach.de sehr naheliegend und hat schnell zum Erfolg geführt. Das erleichterte uns auch die Auswahl des Datenbanksystems, da MySQL des Öfteren empfohlen wurde und wir bereits Vorkenntnisse darin hatten.

Nachdem wir diese Grundlagen festgelegt hatten erstellten wir ein Wireframe mit grundlegendsten Funktionalitäten.

WIREFRAME

Im Anschluss haben wir sämtliche Funktionalitäten der Seite genauer definiert und darauf basierend herausgearbeitet welche Daten in der Datenbank gespeichert werden müssen. Außerdem haben wir uns ein Projekt auf github.com erstellt zur erleichterten Synchronisierung unserer Fortschritte.

# 4. Realisierung

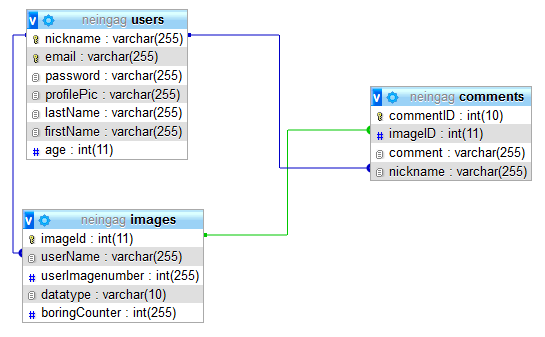
# 4.1 Datenbank

**Users:**

„Users“ speichert sämtliche Informationen, die den Nutzer betreffen. Die wichtigsten hier sind E-Mail, Nickname und Passwort. Ersteres sind „Unique Keys“, dürfen also nur einmal in der Datenbank vorhanden sein. Dadurch konnten wir den Nickname auch als „Primary Key“ verwenden, was uns die Informationsabfrage erleichtern wird, da wir das dann auch als „Foreign Key“ verwenden können. Restliche Informationen sind optional. Passwort wird verschlüsselt gespeichert.  
 **Images:**

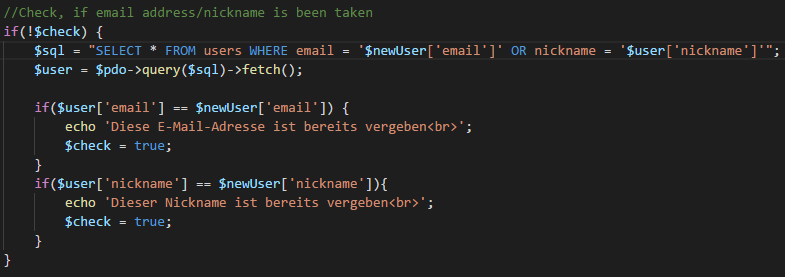
„Images“ erstellt und inkrementiert automatisch die „ImageId“ („Primary Key“) bei einem Upload. „UserName“ gibt uns die Information, wer die Bilder hochgeladen hat und die „userImagenumber“ indiziert und inkrementiert (durch Code) die Nummer des Bildes vom jeweiligen User. Hat beispielsweise „MaxMuster“ drei Bilder hochgeladen, wird das nächste Bild die „userImagenumber == 4“ haben. Außerdem dienen die Infos zum Speichern und Aufrufen der Bilder, genauso wie das Dateiformat. Die „Votes“ der Bilder werden auch hier abgespeichert. Dazu später mehr.   
 **Comments:**

„Comments“ speichert sämtliche Kommentare in der Datenbank. „CommentId“ wird automatisch mit jedem neuen Kommentar inkrementiert. Außerdem wird dazu auch der Nickname des Autors und die ID des betroffenen Bildes gespeichert.

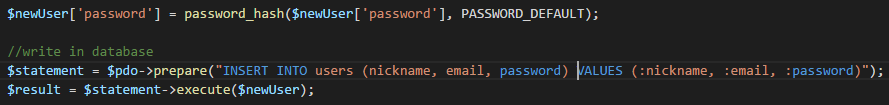


# 4.2 Registration

Die Registration war unsere erste Programmierhürde. Nachdem wir ein Formular erstellten, welches sämtliche Eingaben per POST-Methode übergibt, speicherten wir alles in ein Array, und verglichen die gegebenen Informationen mit unseren Bedingungen. Entspricht das Passwort unseren Vorstellungen? Wurde der Benutzername bereits vergeben, oder ist die E-Mail-Adresse bereits vorhanden? Diese Probleme wurden mit folgendem Codeschnipsel gelöst:



Sobald alle Bedingungen erfüllt sind, wird das Passwort verschlüsselt und ein Datenbankeintrag hinterlegt.

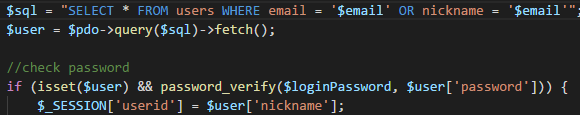


Zeitgleich wird ein Ordner für die Bilder des frisch registrierten Nutzers erstellt, worin sämtliche Uploads gespeichert werden.



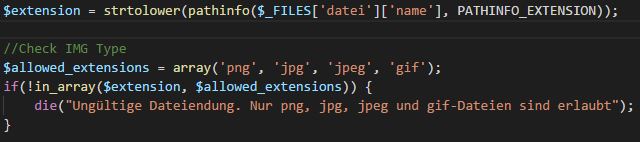
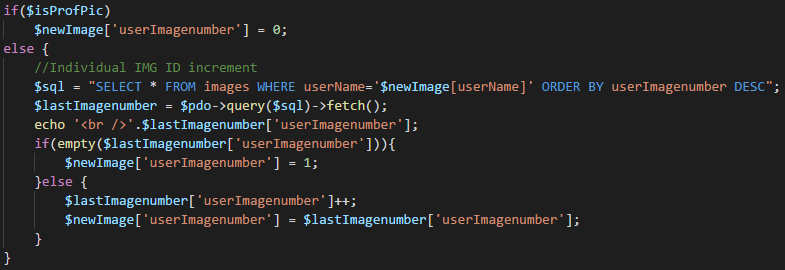
# 4.3 Login

Der nächste Schritt war die Implementierung der Login-Funktion.



Hier wird die E-Mail-Adresse, oder Nickname, so wie das Passwort per POST-Methode übergeben.   
Daraufhin wird aus der Datenbank die jeweilige Spalte mit der angegebenen E-Mail-Adresse/Nickname ausgegeben und das jeweilige Passwort über eine gegebene Funktion überprüft. Diese Funktion vergleicht einen Hash-Wert mit dem Passwort und kann damit die Richtigkeit verifizieren.   
Danach wird der Nickname für weitere Zwecke in der $\_SESSION – Variable gespeichert.

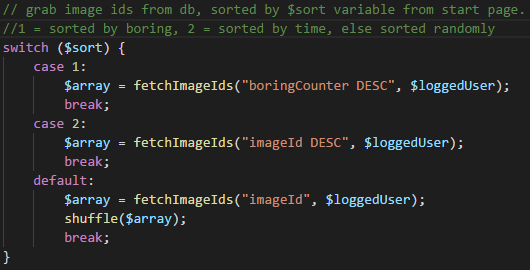
# 4.4 Upload

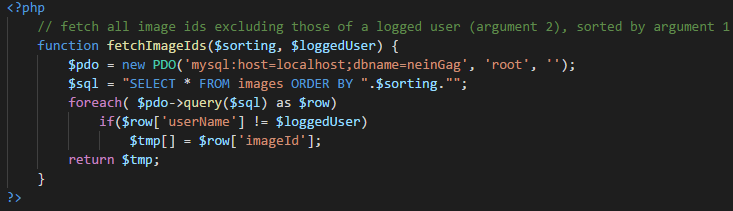
Zum Uploaden der Bilder mussten wir Fragen beantworten wie:   
„Wie begrenzen wir die Dateiformate?“, „Wie bzw. unter welchem Namen speichern wir die Bilder?“, „Wie rufen wir sie ab?“ und „Wie unterscheiden wir Profilbilder mit den nicht-Profilbildern?“.   
Dank php-einfach.de ließen sich die Fragen schnell beantworten. Mit Hilfe der Bibliotheksfunktion „pathinfo()“ konnten wir den Dateityp herausfiltern und begrenzen. Auch weitere Restriktionen haben wir eingeführt, z. B. die Bildgröße auf 500kb.   
Mit der Hilfe der Variable $\_SESSION, nehmen wir zur Dateibenennung den Nickname, fügen am Ende ein „\_“, die Bildnummer und den Dateityp hinzu. Die Bildnummer inkrementieren wir manuell ab dem Wert 1.   
Den Wert 0 erhalten nur Profilbilder, was uns die Trennung erleichtert.   
Hiernach wird das Bild in den jeweiligen Nutzerordner auf dem Server gespeichert. So können wir es später für die Betrachtung nutzen.

Zur Unterscheidung während des Uploads von Profilbildern und normalen Bildern, mussten wir herausfinden, woher die Upload-Anfrage kommt. Also haben wir über die GET-Methode die Information von der Profil-Seite über die URL herausgefiltert und mit Hilfe einer Fallunterscheidung die Gründe des Uploads bestimmt.   
Falls es dann ein Profilbild war, wurde in der Datenbanktabelle „users“ die jeweilige Spalte „profilePic“ verändert.

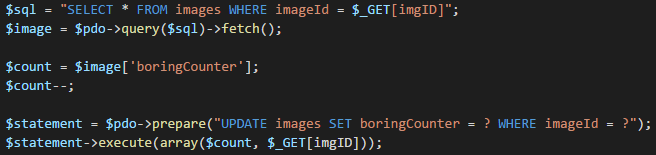
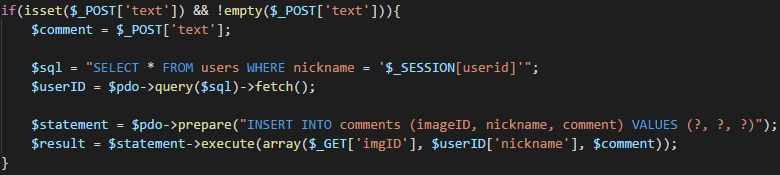


# 4.5 Anzeige

Um zu erkennen, welche Sortiermethode der Nutzer wünscht, wird auf der jeweiligen Seite einer Variable einen Wert zugewiesen und in ein Switch-Case übergeben. 

Je nach gewünschter Sortierung wird dann der Befehl per String als Parameter in einer Funktion übergeben, ausgeführt und das Ergebnis im Array gespeichert. Es wird auch der Nickname als Parameter übergeben, um die Anzeige eigener Bilder zu vermeiden.   
Sobald das Array gefüllt ist, wird es über eine Schleife in einem schön verpackten Div-Container ausgegeben.

# 4.6 Kommentieren / Votes

Nun fehlen nur noch die Kommentare und die Up-/ Downvotes.   
Die Votes beginnen ab dem Wert 0, so landet zumindest die erste Information darüber in der Datenbank. Danach wird bei jedem Vote abgefragt, welchen Wert der „boringCounter“ des jeweiligen Bildes hat und dementsprechend addiert, oder wie im folgendem Beispiel subtrahiert.   
Beim Absenden wird der Kommentar in eine Variable gespeichert und mit ein paar anderen Informationen der Datenbank hinzugefügt.   
Durch diese Informationen können wir dafür sorgen, dass die jeweiligen Bilder ihre Kommentare vom richtigen Autor angezeigt bekommen. 

# 5. Zusammenfassung

Im Großem und Ganzem sind wir stolz auf unser Werk. Die Arbeit war es Wert um die Erfahrung zu machen, wie das Ganze von der serverseitigen Position aus funktioniert. Dennoch haben wir noch nicht genügend an Erfahrung gesammelt. Viele weitere Funktionen werden wir zukünftig einbauen, um zu sehen wie die Dinge funktionieren.

# 6. Ausblick

Geplante Features für zukünftige Releases beinhalten:

* Social Features
* Erweiterte Profile
* Upload von Videos
* App
* Anzeige der eigenen Bilder eines Users

# 7. Quellenverzeichnis

1. php-einfach.de: Upload, Login, Registration, Datenbankabfrage, Datenbankaktualisierung und weitere Hilfen.
2. stackoverflow.com: Nachschlagen bei Fehlermeldungen
3. w3schools.com: CSS-Styles
4. self-html.org: Hilfestellung bei Funktionen/Algorithmen
5. b11.webdesign-fingerübungen.de (Vorlesung Auszeichnungssprachen b. Herrn Westermann) : CSS-Styles
6. php.net/rmdir: Funktion zum rekursiven Löschen von Verzeichnissen