Dokumentation zum Mello-Projekt von Tobias Hölzer

Überblick

Wir bekamen den Auftrag zum Ende des Schuljahres eine Software zu entwickeln, von der Idee bis zur Veröffentlichung. Dabei würde unser Lehrer Herr Pelz uns unterstützen.

Nach einigem Hin- und Herüberlegen haben wir uns entschlossen eine Social Media Seite zu programmieren, welche auf einer modularen Programmierung basiert. Nutzer können Dienste abonnieren und deabonnieren, die Basis macht das Profil des Nutzers aus, die ganz nach Belieben des Nutzers designt werden kann.

Da alle aus unserer Gruppe Erfahrungen mit JavaScript, HTML und CSS haben, wollten wir unsere kleine Social Media Seite mit diesen Programmier- und Auszeichnungssprachen programmieren. Dazu benötigten wir noch einige zusätzliche Programme und Tools:

- MongoDB
- Node.js
- Npm

Eine große Hilfe war auch die NextCloud von Robert, auf der wir unsere Dateien speichern konnten, ebenso hat sie bei der Planung geholfen. Ebenso konnten wir auf einem seiner Server Git installieren. Mithilfe von Git war es uns möglich verschiedene Versionen unseres Programms zu speichern.

Nach dem Anfertigen eines kurzen Pflichtenheftes fingen wir an eine Basis zu planen und zu implementieren. In diesem Bericht möchte ich meine Aufgaben und Tätigkeiten während des Projektes beschreiben und erklären.

Inhaltsverzeichnis

Benutzte Technologien

MongoDB

> *MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need*

-MONGODB.COM/WHAT-IS-MONGODB

MongoDB ist unser Datenbank-Management-System. Durch MongoDB kann man mit Javascript eine Datenbank erstellen und darauf zugreifen. MongoDB ist außerdem schneller als herkömmliches SQL, da es Anfragen anders behandelt. MongoDB basiert auf C++, einer der schnellsten Sprachen der Welt.

Node.js

> *Node.js® ist eine JavaScript-Laufzeitumgebung, die auf Chromes V8 JavaScript-Engine basiert. Durch ein Event-basiertes, blockierungsfreies I/O-Modell ist Node.js schlank und effizient. Mit npm hat Node.js das größte Ökosystem für Open Source Bibliotheken der Welt.*

-NODEJS.ORG

Mit Node.js ist es uns möglich mit purem Javascript einen kompletten Web-Server zu bauen. Beim Aufruf unserer Seite werden sogenannte "Requests" an unseren Server geschickt, worauf der Server "Responses" an den Browser zurückschickt. Ein Beispiel für ein Response ist ein einfaches HTML Dokument.

Außerdem ist es mit Node.js noch möglich Javascript Dateien miteinander zu verknüpfen, sowie npm-Module in seine Datei zu importieren.

NPM

> *npm is the package manager for JavaScript and the world's largest software registry. Discover packages of reusable code — and assemble them in powerful new ways.*

-NPMJS.COM

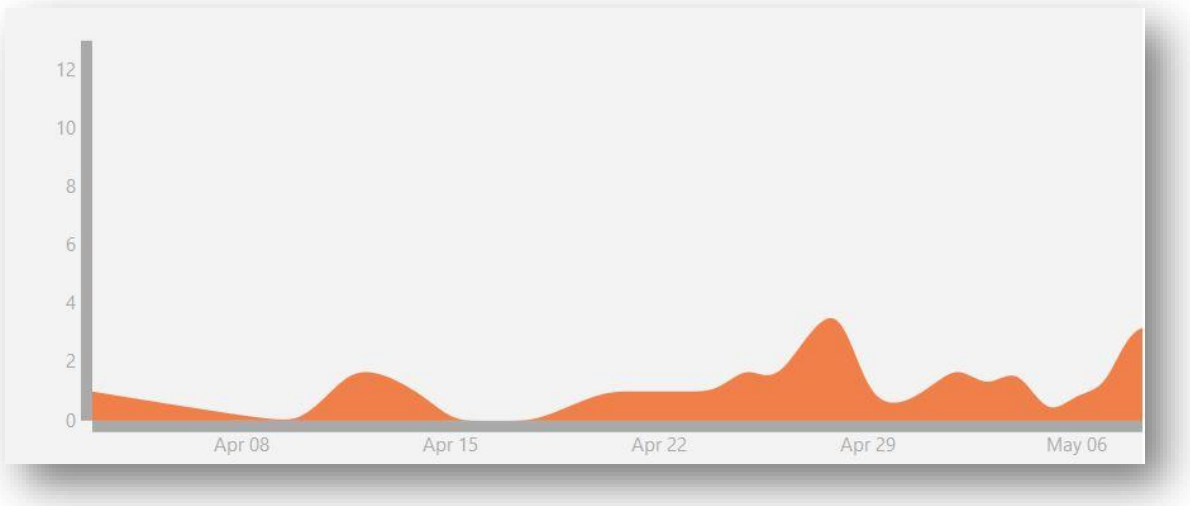
Mittels npm können wir fertige Module/Packeges herunterladen und in unser Projekt integrieren. Wir benutzen eine Vielzahl von npm-Packages, so ersparen wir uns viel Arbeit.

Mein Bericht

Vorab

Ich habe während des Projektes viel im Internet zu Dingen recherchieren müssen, wie sie funktionieren, beispielsweise MongoDB oder EJS. Dazu schaute ich regelmäßig in den Dokumentationen(Manuals, "RTFM") und auf Seiten wie StackOverflow nach und habe mir auch davon Teile meiner Lösungen genommen. ***Trotzdem sind so gut wie alle meine Implementierungen und Lösungsansätze von mir erarbeitet verstanden und implementiert!*** Ausnahmen werden in der Dokumentation vermerkt.

Meine Dokumentation ist chronologisch angeordnet, da ich versuche einen Erzählfluss zu generieren. In diesem Erzählfluss werde ich zum einen erklären was und wie ich etwas gemacht habe und jeweils dazu im Anschluss eine grobe Übersicht über die Funktionsweise dessen was ich gemacht habe geben. Nur grob, da ich sehr viel Zeit in das Projekt investiert habe und demnach auch sehr viel gemacht habe. Ich werde jedoch NICHT den Code oder die Implementierung angesprochener Lösungen erklären.



-MEINE GIT-UPLOADS

-ALLE GIT-UPLOADS

Planung

[18.03.] – [25.04.]

Nachdem die Gruppeneinteilung für das Projekt klar war, haben wir uns in meiner Gruppe überlegt, was wir überhaupt machen wollen.

Jeder hatte ein paar Ideen, ich hatte die Idee, eine Website zu machen, welche Online-Direktwahlen ermöglicht, nach einem System, welches von mir und einem Freund entwickelt wurde. Dieses hätte ich in den Politik-Unterricht einbringen können, jedoch hätten sich die Anderen aus meiner Gruppe nicht damit identifizieren können, also schob ich die Idee beiseite und suchte nach etwas Neuem. Da wir mit dem Gedanken gespielt haben ein Spiel zu entwickeln, dachte ich mir, ob wir vielleicht einen Wirtschaft Simulator programmieren könnten, jedoch wurde diese Idee direkt ausgeschlagen.

Hannah wollte eine Kunst-Website machen, ähnlich wie Instagram, nur eben für Künstler. Diese Idee gefiel uns nicht so sehr, aber da uns bis dahin nichts besseres eigefallen ist, haben wir die Idee weiterentwickelt. So wollten wir nun eine Social-Media Seite entwickeln, jedoch nicht nur für Künstler, so haben wir eine größere Zielgruppe.

Meiner Meinung nach sind Alleinstellungsmerkmale für Social Media Webseiten enorm wichtig, da es nur so möglich sei, sich von der Konkurrenz abzuheben. In meiner Gruppe war ich wohl der einzige mit dieser Ansicht, da alle anderen einfach nur die Features von Facebook, Instagram und Co. kopieren wollten. (Die Ideen, nicht den Quellcode) Trotzdem habe ich mich durchgesetzt und wir haben Alleinstellungsmerkmale für unsere Website festgelegt:

1. Verschiedene Designs:
    1.1. Der Benutzer kann zwischen verschieden Themes wählen
    1.2. Der Benutzer kann seine Profilseite so frei gestalten wie er möchte
2. Abonnierbare Module:
    2.1. Dem Benutzer soll nichts aufgedrängt werden
    2.2. Der Benutzer kann verschiedene Module abonnieren
    2.3. Der Benutzer hat ein Dashboard, auf das abonnierte Module zugreifen

Ich bestand darauf, dass wir unser Projekt modular programmieren, da wir so besser Planen konnten und Freiraum für spätere Ideen haben. Modular heißt, dass wir eine Basis haben, der wir nach und nach einzelne Module hinzufügen können, ohne dabei die Basis oder andere Module zu verändern. Gedacht war, dass am Ende jeder alleine für sich Module hinzufügen kann, ohne sich dabei auf wen Anderen aus der Gruppe verlassen zu müssen.

Also haben wir uns überlegt, was wir in unsere Basis machen müssen, um so eine Modulare Programmierung zu unterstützen. Auch haben wir Modul-Ideen gesammelt, ein Großteil davon entstammte einfach aus den einzelnen abgelehnten Ideen.

Aus den gesammelten Ideen und Konzepten erstellten wir ein grobes Lastenheft, welches nicht mehr als eine Seite besaß. In diesem setzten wir uns für das Projekt folgende Ziele:

- Eine Basis zu erstellen, welche die Möglichkeit hat durch Module erweitert zu werden.
    - Personalisierte Profile
    - Design des Benutzerprofils vom Benutzer veränderbar
    - Design der Website wählbar
    - Benutzer haben ständigen Zugriff auf ihre Daten
- Module
    - Optionale Erweiterungen für die Basis
    - Beispielmodule erstellen
        - Wetter
        - ToDo-Liste
        - Chats
    - Benutzer kann Module auswählen

So hatten wir eine sehr vage Ahnung wo die Reise hinführen sollte, aber leider auch nicht mehr.

Es war in unserem Fall schwierig die Implementierung im Vorhinein zu planen, da niemand von uns wusste, wie unser Endprodukt eigentlich nun wirklich aussehen sollte und nur jeder von uns eine Ahnung hatte, wie die Vorstellungen der Anderen aussahen. Daher war unsere "Planung" äußerst schnell und ohne große Probleme abgeschlossen, was jedoch später in der Implementierung für mehr Aufwand sorgte.

Zwischen Planung und Implementierung

[26.03.] – [06.04.]

Ich habe Hannah welche sich anfangs um das Design kümmern wollte gebeten, einen Design Vorschlag für Login und Register einzureichen. Diesen habe ich dann implementiert, jedoch ohne jegliche Funktion. Jetzt hatten wir einen Ordner mit einigen HTML-Dateien welche alle auf zwei CSS-Dateien zugreiften. Unterdessen kümmerte sich Robert um den Anfang der Funktionalitäten der Basis, sprich Login und Register.

Da ich nichts zu tun hatte, habe ich ein weiteres Design (Dieses sollte während der intensiven Entwicklung unser Standard Design sein) entwickelt. Hier traten auch erstmals Verzweigungen durch Links zwischen den verschiedenen Seiten auf. Außerdem hatte ich sporadische AGBs, Impressum und Info Seiten erstellt.

Ich habe Kevin vorgeschlagen ein weiteres Design zu erstellen um sich ein wenig mit CSS bekannt zu machen. Dadurch hatten wir nun drei Designs, welche später in unsere Basis implementiert werden konnten.

Es hat Zeit gekostet, bis ich wirklich anfangen konnte es sinnvolles zu machen, abgesehen vom Design, da Robert erst nach zwei Wochen den Anfang hochgeladen hatte. Und auch danach musste ich erst einmal verstehen, wie sein Code funktionierte. Zwar hatten wir alle Erfahrung mit JavaScript, jedoch hatte niemand von uns bisher mit Node.js programmiert, ebenso hatte noch niemand zuvor eine Serveranwendung geschrieben.

Aller Anfang ist schwer

[07.04.] – [19.04.]

Da Robert nur Register und Login am Anfang implementiert hatte, habe ich die AGBs, Impressum und Info Seiten hinzugefügt, welche Links vorher zwar im Design zu sehen, allerdings noch nicht aufrufbar waren. Generell waren aufgrund der im Design vorzufindenden Verzweigung der verschiedenen Seiten, jedoch nicht in der Implementierung angegebenen Verzweigung, viele Links und Seiten überhaupt nicht zugänglich. Daher machten Kevin und ich uns dran unser Pfadsystem zu erneuern.

Dazu veränderte Kevin erstmal alle Links so, dass sie zugänglich waren. Dann schauten wir, wie wir dieses System verbessern konnten, diese Planung habe ich dann implementiert. So hatten wir nun zwei Dateien, in der alle Pfade aufgelistet, in einem Objekt (Dictionary) gespeichert und dann als Node-Modul exportiert wird. So können nun die verschiedenen Skripte auf dieses Objekt zugreifen und verwenden. Außerdem sind so die Pfadnamen variabel geworden. (Pfadname=Das was in der URL hinter unserer Serveradresse steht) Zwei Dateien deshalb, weil einmal der Server darauf zugreifen muss und einmal der Browser. Mit EJS konnten wir diese zwei Dateien zu einer machen.

EJS

[20.04.] – [23.04.]

EJS ist ein Node-Modul und eine Server-View-Engine, welches einem erlaubt beim Serverseitigem rendern der Seite der HTML-Datei JavaScript-Variablen zu übergeben. So muss zum Beispiel der Browser nicht mehr auf den Public-Ordner zugreifen, wenn er die Pfadnamen haben möchte. Außerdem ermöglicht EJS Datenbank Einträge an den Browser zu schicken, wie zum Beispiel den Usernamen auf das Dashboard.

EJS ersetzt also das JavaScript einer HTML-Seite und ermöglicht eine einseitige Kommunikation zwischen Server und Browser. Ich habe unser gesamtes Programm auf EJS umgestellt, wodurch es mir möglich wurde Fehlermeldungen bei der Registrierung anzeigen zu lassen.

Die Anwendungsweise von EJS ist recht simpel. Der Server rendert eine EJS-Datei, welcher über ein Objekt(Dictionary) Variablen übergeben wurden. Da die Seite also bei so gut wie jedem Laden anders aussieht, rendern die EJS-Dateien nicht schon beim Serverstart, sondern erst beim Aufruf über den Browser. EJS-Dateien benutzen HTML, jedoch kann man durch bestimmte Tags EJS aufrufen, wobei man die übergebenden Variablen nutzen kann. Hier ein kleines Beispiel, in dem **Hallo Tobias, Hallo Robert, Hallo Kevin, Hallo Hannah** ausgegeben wird:

```
1 <div>
2     <% Usernames.forEach(user){ %>//user nimmt einmal jeden Username an
3     <p>Hallo <%= user %>, </p>  //Es wird 4 mal Hallo [user], ausgegeben
4     <% } %>   //beenden der Schleife wird von EJS berücksichtigt
5 </div>
```

IN VIEW.EJS

```
1 res.render("view.ejs", {
2     Usernames: ["Tobias","Robert","Kevin","Hannah" ]
3     //Hier kann man auch Dinge aus der Datenbank übergeben.
4 )};
```

IN APP.JS

Die Modul-Engine

[24.04.] – [28.04.]

Nun war es für mich an der Zeit mich um die Module zu kümmern, schließlich war es ja auch meine Idee und ich hatte schon Vorstellungen wie ich diese umsetzen könnte. Mein Ziel war es, dass es unserer Gruppe zukünftig möglich sein könne, einen Ordner mit bestimmten Dateien in einen "modules" Ordner zu machen und das so automatisch das Modul hinzugefügt wird.

Ich fing an mir einen Strukturplan für ein Beispielmodul zu erstellen, heißt welche Variablen dieses Modul übergeben muss, um zu funktionieren. Dabei habe ich zwischen drei Arten von Modulen unterschieden:

- *Dashed*: Ein Modul, welches man auf dem Dashboard hinzufügen kann.
- *Sited*: Ein Modul, welches eine neue Seite erstellt und via link besucht werden kann.
- *Both:* Ein Modul, welches sowohl die Eigenschaften eines Dashed-Modul, als auch eines Sited-Modul

Mein System sollte automatisch anhand der Variablen erkennen können, um welche Art von Modul es sich handelt.

Außerdem brauchte ich einen Store, wo der Benutzer die Module seinem Account hinzufügen und entfernen kann. Das hatte zufolge, dass der Benutzer in der Datenbank Variablen benötigte, sodass die hinzugefügten Module gespeichert werden. Schon im Vorhinein war mir klar, dass hier Fehler auftreten könnten, sollte ein Modul von uns zum Server hinzugefügt werden, der virtuelle Benutzer wisse aber noch gar nicht, dass es ein neues Modul gibt.

Also fing ich an alles zu implementieren und da alles voneinander abhängt implementierte ich alles gleichzeitig. Dies erforderte natürlich enorme Konzentrationsstärke da ich an mehreren Stellen gleichzeitig arbeitete, damit konnte ich allerdings umgehen. Um das Problem mit den "unwissenden" virtuellen Benutzern umging ich, indem bei jedem Login überprüft wird, ob der Benutzer die Daten hat.

Dazu: Diese Daten sind nichts anderen als Booleansche Objekte, welche zu jedem Modulnamen einen true oder false Wert speichern, halt eben in Abhängigkeit ob der Benutzer ein Modul hinzugefügt und gegebenfalls an das Dashboard gepinnt hat.

Im Prinzip durchsucht mein Algorithmus einen Ordner namens "modules" und versucht dort in jeden einzelnen Ordner zu gehen und jeweils eine Datei namens "use.js" zu importieren. Diese Imports werden dann verarbeitet und die Daten aus den Schnittstellen unserem Kernprogramm hinzugefügt. Um die Imports richtig zu verarbeiten, benötigt jede use.js Datei einen Export mit bestimmten Variablen (Schnittstellen).

Hier ein Beispiel, wie ein Modul implementiert werden kann, welches die Möglichkeit gibt etwas am Dashboard anzuzeigen.

```
1  var express = require('express');
2  var app = express();
3  var router = express.Router();
4
5  module.exports = {router : router,
6                    path_main : "/weather", //erster URL Teil
7                    Name : "Waether", //Anzeigename
8                    db_name : "weather_module", //Datenbankname
9                    Info : "Info about the module", //Informationen
10                   path_view : "/view"} //Pfad zur view.ejs datei
```

Pairprogramming & Debugging

[28.04.] – [24.05.]

Hin und wieder haben Kevin und ich uns gegenseitig bei unseren Aufgaben via Discord© geholfen, in dem einer den Code schreibt und der Andere sagt, wie man das Problem am besten lösen könne. Meistens ging es dabei um Probleme, welche durch meine sehr eingrenzende Modul-Engine verursacht wurden.

Zum Beispiel, dass die Wetterdaten über Socket.IO an das Dashboard gesendet werden müssen, da das Dashboard die einzelnen Views der Module importiert und nicht selbst rendert. Daher können kein Variablen über EJS an das Dokument übergeben werden und es mussten einige Elemente im Skript Teil des Views erstellt werden. Leider sind wir auf diesen Ansatz nicht sofort gekommen.

Aber auch er konnte mir helfen, die Modul-Engine zu verbessern, sowie die Einbindung verschiedener Designs/Themes zu implementieren. Außerdem haben wir zusammen sein Design aus den Ferien als unser Login&Register Design implementiert.

Abgesehen davon habe ich regelmäßig Kleinigkeiten hinzugefügt, das Design angepasst und erweitert, Bugs entfernt, sowie meine Modul-Engine verbessert. Beispielsweise habe ich um den Modulerstellern ihre Arbeit zu erleichtern ihnen die Möglichkeit gegeben Modulvariablen zu übergeben und auch dem Benutzer in der Datenbank eigene Modulvariablen zu geben ohne dabei eine neue Collection mit einer weiteren Benutzerverknüpfung hinzufügen zu müssen. So konnten Module nun individueller auf den Benutzer erstellt werden.

Datenbank Zugriff

[02.05.]

Da es immer ein wenig gedauert hat bis in der Datenbank gespeichert wurde, dass ein Benutzer ein Modul hinzugefügt oder an das Dashboard gepinnt hat, zum Teil so lange, dass die Seite schneller neu geladen hatte, als die Datenbankeinträge aktualisiert wurden und somit falschanzeigen zu Stande kamen. Daher suchte ich nach

schnelleren Datenbank-Zugriffen und bin nach einiger Recherche auf einen interessanten Lösungsansatz gestoßen. Diesen habe ich dann auch sofort übernommen und konnte so mein Problem lösen.

Vorher	Nachher
Verbindungsaufbau zur Datenbank bei jeder Änderungsanfrage	Einmaliger Verbindungsaufbau zur Datenbank bei Start des Programms
Mindestens 3 Sekunden Laufzeit um einen Datenbankeintrag zu ändern	Maximal 1 Sekunde Laufzeit um viele Datenbankeinträge einzeln zu ändern
Verbindung zur Datenbank musste nach jeder Änderung geschlossen werden	Verbindung bleibt erhalten und wird erst beim Beenden des Programms geschlossen
Bei nicht Schließung der Verbindungen: Viele Verbindungen verlangsamen die Datenbank	Eine Verbindung alleine verlangsamt Datenbank nicht

Benutzer Einstellungen

[04.05.] – [05.05.]

Da ich nun ein besseres Verständnis für Datenbankzugriffe hatte, habe ich begonnen eine Seite für Benutzer-Einstellungen zu entwerfen und implementieren. Dabei handelt es sich lediglich um Input Abfragen aus dem Client-Dokument und diese dann einem Benutzer in der Datenbank richtig hinzuzufügen. Bei den "normalen" Daten wie zum Beispiel Name oder Geburtstag gab es wenig Probleme, erst bei dem Hashen der Passwörter bekam ich Probleme. Robert bat an, dies für mich zu übernehmen, so konnte ich mich auf die nächsten Aufgaben konzentrieren.

Benutzerdefinierte Themes

[08.05.] – [10.05.]

Eines unserer Ziele war es verschiedene Designs zu erstellen und dem Benutzer die Entscheidung überlassen, welches Design er denn benutzen möchte. Da wir auch schon mehrere Designs hatten, habe ich angefangen eine Engine für ein Benutzervariables Themes zu implementieren und da es zu aufwendig gewesen wäre ein schon vorhandenes Design zu konvertieren, zugleich ein zweites Theme als Beispiel zu erstellen. Dies lief dank der "include" Funktion von EJS komplett ohne Probleme ab, so konnte ich noch ein wenig mit dem Design meines Beispiel Themes beschäftigen. Ein paar Wochen später habe ich noch ein zusätzliches erstellt, da mir mein Beispiel Design nicht so sehr gefiel.

Das WM-Modul & Gruppen Modul

[18.05.] – [23.05.]

Mein Vater und ich wollten zusammen ein Tippspiel für die kommende Weltmeisterschaft in Python schreiben, so kam ich auf die Idee auch ein Tippspiel als Modul zu erstellen. Dies war auch die Gelegenheit zu beweisen, dass unser Kurs ein besseres Tippspiel zu Stande bekommt als der zweite Kurs bei Herrn Diez.

Dieses Modul zu entwerfen und zu implementieren hat sehr viel relativ viel Zeit in Anspruch genommen, vor allem da ich bei zwei oder drei möglichen Benutzer Tools einfach alle drei gewählt habe. Zum Beispiel ob ein Benutzer Gruppenabhängig Tippt oder für alle Gruppen das gleiche Tippt: ich habe einfach beides implementiert.

Um besagte Gruppenabhängigkeit zu implementieren, entwarf und implementierte ich außerdem auch noch ein Gruppenmodul, indem der Benutzer Gruppen erstellen und beitreten kann. Das Gruppenmodul konnte Robert auch in seinem Einzel-Chat benutzen

Beide Module sind sehr weitreichend programmiert und die Logik und Probleme dahinter sehr komplex, daher werde ich sie hier weiter nicht erläutern.

Lernen

Ich habe mich schon im Vorhinein unglaublich auf dieses Projekt gefreut, da es für mich eine super Gelegenheit war die Abläufe der Softwareentwicklung, neue Technologien und die Implementierung eines Servers kennen zu lernen und zu verstehen. Vor allem war es aber auch eine Gelegenheit zu schauen, ob eine Zukunft als Informatiker für mich in Frage kommt. Abgesehen davon freute ich mich auch auf die Arbeit in einer Gruppe und darauf mich auf meine Gruppenmitglieder verlassen zu können.

Auf jeden Fall habe ich unglaublich viel aus dem Projekt mitnehmen können, ich weiß nun wie man Serveranwendungen in JavaScript schreibt, genauso kenne ich nun Alternativen zu SQL und verstehe nun viel besser den Umfang einer einfachen Webanwendung.

Genauso haben sich meine Fertigkeiten als Gruppenmitglied verbessert und ich habe auch etwas für Gruppenmanagement gelernt. Damit meine ich nicht, dass ich Leiter meiner Gruppe war, wir waren alle Gleichgestellte. Sondern denke ich, dass ich das Projekt über den kompletten Durchblick über unser Projekt hatte und auch bei Problemen als Vermittler dienen konnte.

Rückblickend haben wir viele Fehler in der Entwicklung gemacht und waren auch relativ ineffizient. Der größte Fehler den wir gemacht haben war, dass wir im Vorhinein unser Projekt nicht durchgeplant hatten und auch keinen Zeitplan erstellt haben. So habe ich zum Beispiel enorm viel Zeit in kleinere Dinge gesteckt, welche auch nicht weiter relevant waren.

Aus dem gleichen Grund konnten wir auch unser Projekt nicht zu unserer vollen Zufriedenheit abschließen. Geplant war, unsere "Basis" komplett fertigzustellen. Die komplette Profilsektion konnte nicht implementiert werden, da anfangs dem Beauftragtem keine genaueren Fristen dazu festgelegt wurden und die Aufgabe ein wenig unterschätzt wurde.

Dafür ist es dem Benutzer möglich seine Daten jederzeit zu verändern (bis auf den username, der wird in der Datenbank unter anderem als Primärschlüssel verwendet und daher nicht veränderbar ist.). Außerdem kann der Benutzer zwischen 2 (und einem schlechtem) Design/Theme wählen. Die Modul-Engine erfüllt ihren Zweck, darüber hinaus ist sie nun auch zu einem Alleinstellungsmerkmal geworden und funktioniert bei richtiger Anwendung der Modul-Programmierer Fehlerfrei. Ebenso ist natürlich eine Fehlerfreie Registrierung und Anmeldung möglich und sechs Module funktionieren einwandfrei.

Hierbei anzumerken ist, dass die Zahlen für die Module und Themes wenig sind, sie sollen lediglich das Potential unserer Website zeigen. Module und Themes können unglaublich einfach hinzugefügt werden.

What's next?

Es sind noch viele Dinge geplant, zum Beispiel die Umgestaltung und Verbesserung der Modul-Engine oder die Erweiterung durch neue Module. Ebenso als eines unserer nächsten Ziele ist die Fertigstellung der Profilseite und die Suche nach Profilen. Eine Idee von mir war auch noch, so bald wie möglich in die Alpha-Testphase zu gelangen, in der wir unsere Freunde und Verwandte die Website nutzen lassen und für sie Fragebögen zur Verbesserung unserer Website erstellen, die sie dann ausfüllen können.