Web- Programmierung

WS 2015/16

- Dokumentation der Semesterarbeit -

Comeet

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Thomas Krieger**  **Matrikel-Nr.: 257510** | **Marileen Stamer Matrikel-Nr.: 258234** | **Torsten Garding**  **Matrikel-Nr.: 257523** |
|  |  |  |

Inhaltsverzeichnis

[1. Einführung 1](#_Toc440754158)

[1.1 Projektbeschreibung 1](#_Toc440754159)

[1.2 Funktionen von Comeet 1](#_Toc440754160)

[1.3 Zusätzliche Ziele 2](#_Toc440754161)

[1.4 Verwendete Technologien 2](#_Toc440754162)

[1.4.1 Grunt 2](#_Toc440754163)

[1.4.2 Less / CSS 2](#_Toc440754164)

[1.4.3 Bootstrap Grid System 2](#_Toc440754165)

[1.4.4 Handlebars / HTML 3](#_Toc440754166)

[1.4.5 JavaScript Clientseitig 3](#_Toc440754167)

[1.4.6 JavaScript Serverseitig (Express) 3](#_Toc440754168)

[1.4.7 Bootstrap Datepicker 3](#_Toc440754169)

[1.4.8 MySQL 3](#_Toc440754170)

[1.4.9 PHP 4](#_Toc440754171)

[1.5 Zielumgebung 4](#_Toc440754172)

[1.6 Entwicklungssysteme 5](#_Toc440754173)

[2. Entwicklung 6](#_Toc440754174)

[2.1 Verzeichnisstruktur 7](#_Toc440754175)

[2.2 Design 7](#_Toc440754176)

[2.3 Datenbank 8](#_Toc440754177)

[2.3.1 Attendees 8](#_Toc440754178)

[2.3.2 Contacts 9](#_Toc440754179)

[2.3.3 Events 10](#_Toc440754180)

[2.3.4 Items 11](#_Toc440754181)

[2.3.5 Users 12](#_Toc440754182)

[2.4 Funktion 12](#_Toc440754183)

[2.5 Background-Engine 12](#_Toc440754184)

[3. Implementierung 14](#_Toc440754185)

[3.1 Abweichungen vom Entwurf 15](#_Toc440754186)

[3.2 Programmlogik 15](#_Toc440754187)

[3.3 Einschränkungen und Features 15](#_Toc440754188)

[3.4 Quellcode 15](#_Toc440754189)

[3.5 Installation 15](#_Toc440754190)

1. Einführung

Als Prüfungsleistung entwickeln Sie eine kleine Anwendung (Semesterprojekt). Zu Ihrem Projekt erstellen Sie eine Dokumentation. Das Semesterprojekt bearbeiten Sie in Gruppen von 3 ­ 4 Personen. Es ist gut, wenn sich Teampartner mit ähnlichen Vorkenntnissen finden. Themen zur Auswahl:

1. Control Panel eines Meß‐ und Steuerungsgeräts
   * Ziel: Weboberfläche einer existierenden webbasierten Schnittstelle ansprechend gestalten
   * Einlesen und Darstellen von Messwerten
2. Webanwendung für selbstorganisierte Eventorganisation
   * Ziel: eine kleine Anwendung konzeptionieren und prototypisch umsetzen
   * Events anlegen, bearbeiten, Nutzer zuordnen o.ä.

Die beiden Themen werden auf der ersten Präsenz nochmal genauer vorgestellt und erläutert.

1.1 Projektbeschreibung

Als Semesteraufgabe im WS2015/16 soll im Modul Webprogrammierung eine Webanwendung zur selbstorganisierten Eventplanung entwickelt werden. System sowie Inhaltsgestaltung sind für die Studenten relativ frei wählbar. Alternativ konnte man ein Interface zur Steuerung einer Maschine entwickeln. Wir wählten die Eventplanung und nannten unser Portal „Comeet“ (für ‚Come and meet‘).

1.2 Funktionen von Comeet

Comeet stellt eine Anwendung dar, mit welcher registrierte Benutzer eigene Events/Veranstaltungen anlegen können. Dazu können neue Kontakte per E-Mail oder andere registrierte Benutzer direkt eingeladen werden. Die Benutzer können ebenfalls die Events ihrer Kontakte sehen, sowie ihre eigenen Events verwalten.

* Nutzer können sich registrieren
* Registrierte Nutzer können sich einloggen
* Eingeloggte Nutzer können:
  + Events anlegen
  + Events bearbeiten
  + Ihr Profil bearbeiten
  + Ihre Kontaktliste anzeigen und verwalten
  + An Events von Freunden teilnehmen
  + Sich aktiv an der Planung beteiligen (Mitbringliste)

1.3 Zusätzliche Ziele

* Per E-Mail eingeladene Kontakte können ihre Eventteilnahme bestätigen oder absagen

1.4 Verwendete Technologien

Im Folgenden führen wir alle verwendeten Technologien auf.

1.4.1 Grunt

Grunt ist ein "Task Runner" Tool, welches auf Node.js basiert. Wir verwenden dieses Tool während der Entwicklung, da es die Arbeit am Projekt sehr erleichtert. Wir verwenden die Module "connect" um einen lokalen Server zu starten, der die statischen Dateien ausliefert. Das "watch" Modul "beobachtet" während der Entwicklung das komplette Verzeichnis und erzeugt einen live reload im Browser. Wird nur am CSS etwas geändert, so wird auch nur die entsprechende CSS Datei neu geladen, ohne dass sich deswegen die ganze Seite neu lädt.

Des Weiteren werden die Module "less" und "compile handlebars" genutzt. Diese dienen zum kompilieren von less zu CSS und handlebars Templates zu HTML. Dadurch können wir im Baukasten-Prinzip entwickeln.

1.4.2 Less / CSS

LESS ist ein Framework, welches die Fähigkeiten von CSS mit dynamischen Funktionen wie Variablen, Mixins, Operationen und Funktionen erweitert. LESS wird für die Entwicklung genutzt und von einem less-Compiler (den wir als Grunt Node Modul eingebunden haben) zu CSS kompiliert.

1.4.3 Bootstrap Grid System

Da wir die Anwendung responsive machen wollen, haben wir das Gridsystem von Bootstrap eingesetzt um schneller starten zu können und nicht erst selbst erzeugen zu müssen. Das Gridsystem bringt Klassen für Spalteneinteilung und Breakpoints mit, wir nutzen ein fluid layout (volle Breite) und das Standard 12 Spalten Grid.

1.4.4 Handlebars / HTML

Handlebars ist eine Template Engine, die recht leicht zu verwenden ist. Wir verwenden weniger die eigentliche Templating Funktionen (Expressions, Helper ..), sondern hauptsächlich nur die Basis- Funktionalität um für die einzelnen Bausteine der Anwendung einzelne Partials anzulegen, die man dann gegenseitig in anderen Templates inkludieren kann.

1.4.5 JavaScript Clientseitig

Wir versuchen, frontendseitig das meiste mit CSS zu lösen, es ist JavaScript notwendig für:

* Das Layer (Öffnen und Schließen)
* Formularvalidierung bei der Registrierung und beim Anlegen eines neuen Events
* Teilweise zum Setzen von CSS Klassen für CSS Animationen
* Ajax Post requests zum Server

1.4.6 JavaScript Serverseitig (Express)

Da wir schon mit Node und dem Grunt Connect Modul arbeiten, haben wir beschlossen backendseitig doch kein PHP einzusetzen. Stattdessen nehmen wir einen Node.js Express Server, der die POST Requests abhandelt und mit der Datenbank kommuniziert, und gleichzeitig auch die statischen Inhalte ausliefert.

Diese Entscheidung hat evtl. einen Umbau zur Folge, sodass wir Grunt rauswerfen und auch die less und handlebars Dateien vom Express Server kompilieren lassen. Einen live reload müsste man auch auf diesem Wege hinbekommen.

Hier ergaben sich während der Entwicklung Änderungen, siehe 3.1.

1.4.7 Pickaday Datepicker

Ein kleiner Datepicker, ohne Abhängikeiten (wie jQuery o.ä.), der einfach einzubinden und zu stylen ist.

Quelle: <https://github.com/dbushell/Pikaday>

Wir nutzen ihn um ein Datum für ein Event zu selektieren und die nächsten zwei Kalendermonate in der Seitenleiste anzuzeigen

1.4.8 MySQL

MySQL ist eines der weltweit verbreitetsten relationalen Datenbankverwaltungssysteme. Es ist als Open-Source-Software sowie als kommerzielle Enterpriseversion für verschiedene Betriebssysteme verfügbar und bildet die Grundlage für viele dynamische Webauftritte.

Verwendete Version: 5.0.96

1.4.9 PHP

PHP ist eine weit verbreitete Scriptsprache. Die verwendete Template-Engine ist in PHP geschrieben, ebenso wird die Template-Engine aus PHP heraus angesprochen. Zusätzlich kann man mit PHP sehr komfortabel Datenbanken ansprechen. Die Scriptsprache wird direkt auf dem Webserver ausgeführt, dieses vollkommen transparent für die Clients. Es ist später nicht mehr zu erkennen, ob der vorliegende Inhalt statisch oder dynamisch durch PHP erzeugt ist. PHP erfordert keine Installation auf Client-Rechnern.

Verwendete Version: 5.4.42

1.5 Zielumgebung

Als Zielsystem für die Blogsoftware kommen jegliche Rechner mit den benötigten Serverdiensten in Frage. Benötigt wird zwingend ein Webserver mit PHP-Unterstützung. Ebenso muss ein Datenbanksystem vorhanden sein, welches SQL beherrscht. Für die volle Funktionalität ist zudem ein funktionstüchtiger Emailserver Voraussetzung. Da die Template Engine Smarty verwendet wird, muss der Webserver ebenso für diese eingerichtet sein. Die Datenbankanbindung wird mittels PHP PEAR:DB realisiert.

Die Voraussetzungen sind bewusst so vage gehalten, damit das Endprodukt auf vielen Systemen eingesetzt werden kann. So ist es 'vollkommen' Betriebssystem und architekturunabhängig - vollkommen in dem Sinn, dass auf der Zielarchitektur natürlich ein Webserver mit PHP verfügbar sein muss.

Auf der Clientseite ist lediglich ein Internetbrowser nötig, der JavaScript beherrscht und aktiviert hat. Auch hier gibt es keine Einschränkung an Betriebssystem oder gar Hardwarearchitektur. Aus Zeitgründen ist das Design der Webseite nur an aktuelle Browser angepasst. Alte Browser wie beispielsweise Microsoft Internet Explorer 6 (und früher), Mozilla Firefox 2 und früher sowie Netscape werden explizit nicht unterstützt!

Das Layout der Seite basiert komplett auf CSS. Es werden weder browserspezifische 'Hacks' oder 'Weichen' eingesetzt

Als Scriptsprache auf dem Server kommt lediglich PHP zum Einsatz. PHP ist bei den meisten Hosting-Angeboten verfügbar, somit lässt sich das Endprodukt (hoffentlich) auf vielen dieser Angebote nutzen.

Als Scriptsprache für die Clients ist JavaScript in Verwendung. Die Auswahl hier ist nicht groß, es bleibt nur JavaScript übrig, sobald man mehrere Browser unterstützen möchte.

1.6 Entwicklungssysteme

Da unsere Gruppe verschiedene Hard- und Software einsetzte, mussten grundlegend erst einmal Wege gefunden werden, gemeinsame Arbeitsgrundlagen zu schaffen. Um weitestgehend gleiche Arbeitsumgebungen zu schaffen, wurde folgende Software eingesetzt:

* Node.JS für Bootstrap und Grunt
* Brackets und IntelliJ als Entwicklungsumgebung
* Notepad++ als Texteditor
* Chrome bzw. Iron als Web-Client zum Testen
* Git als Versionskontrolle

1.6.1 Entwicklungssystem von Thomas Krieger

Thomas setzte Windows 10 Professional als Betriebssystem ein. Zusätzlich zu dem oben genannten Softwarepacket nutzt er Eclipse als ergänzende Entwicklungsumgebung.

1.6.2 Entwicklungssystem von Marileen Stamer

Marileen setzte MacOS 10.9.5 als Betriebssystem ein. Als Entwicklungsumgebung und Zugriff auf Git benutzte sie IntelliJ, als Web-Client kam Chrome zum Einsatz.

1.6.3 Entwicklungssystem von Torsten Garding

Torsten setzte anfangs Linux als Betriebssystem ein, stieg dann aber auf Windows 10 Professional um. Zum Einsatz kamen Node.JS, Grunt, Git und Brackets, sowie SWIron als Web-Client.

2. Entwicklung

Zu Beginn haben wir uns zusammengesetzt und den Zweck und die Zielgruppe für unsere Anwendung bestimmt. Als Ergebnis kam dabei heraus, dass wir unsere Webseite zwar jung und aufgeschlossen darstellen wollen, als Zielgruppe aber grundsätzlich jeder angesprochen werden sollte. Daraus hat sich dann einerseits das Design ergeben, andererseits wurden aber Worte wie z.B. „Freunde“ durch neutraler klingende „Kontakte“ ersetzt. Während man privat sicher „Freunde“ einladen würde, würde man in einer Firma eher von Kollegen oder Mitarbeitern sprechen. Um eine Differenzierung zu vermeiden, entschieden wir uns an dieser Stelle für „Kontakte“. Dies bot die Möglichkeit, diese Webseite nicht nur privat, sondern auch geschäftlich nutzen zu können.

Anschließend haben wir uns über die Funktionen und den Umfang geeinigt. Da der Zeitrahmen recht begrenzt war, versuchten wir uns auf das notwendige zu beschränken. Auf Dinge wie einen integrierten Chat oder Nachrichtendienst wurde daher verzichtet, aber es steht die Idee Raum, die Webseite im Anschluss nach dem Semester weiter zu entwickeln. Im weiteren Verlauf entstanden der Name und das Logo. „Comeet“ ist ein kleines Wortspiel und setzt sich aus den Wörtern „Come and meet“ zusammen, was gleichzeitig als Slogan für unsere Eventplanungs-Webseite verwendet wird. Das Logo mit dem Stern in der Mitte und dem umkreisenden Pfeil symbolisiert dabei einmal die Gemeinschaft, ohne die ein Event ja gar nicht möglich ist. Andererseits stellt es auch einen Kometen dar, der einen Stern umkreist. Die ersten Entwürfe wurden auf Papier angefertigt. Diese wurden später auf den Computer übertragen und dabei verfeinert. Als nächstes besprachen wir, mit welchen Technologien wir arbeiten wollen, und entschieden uns für die unter 1.4 genannten. Das führte dazu, dass wir diese auf jedem genutzten Computer lauffähig machen mussten, was sich bei drei unterschiedlichen Systemen anfangs als nicht ganz so einfach herausstellte. Nachdem wir alle über eine annähernd gleiche Softwarekonfiguration verfügten begannen wir, die ersten Bausteine für die einzelnen Seiten der Plattform zu erstellen. Wir erstellten zu den .hbs Dateien die entsprechenden .less Dateien unter Zuhilfenahme des Grid Systems von Bootstrap. Im Groben orientierten wir uns an den erstellten Entwürfen, stellten aber schnell fest, dass sie auf der fertigen Seite nicht wie geplant wirkten und änderten sie ab. Anschließend bauten wir die einzelnen .hbs Bausteine zu den Seiten unserer Website zusammen und erstellten die Navigation. Hinterher beschäftigten wir uns mit der Funktionalität der Formulare, Buttons, etc. - allerdings erstmal nur so weit, dass wir die Funktionalität optisch darstellen konnten. Bis zu diesem Zeitpunkt hatten wir auf eine nötige Datenbank verzichtet und uns auf die Elemente beschränkt, welche wir für die Präsentation für nötig hielten. Die letzten zwei Wochen (je Mo und Do) vor der zweiten Präsenz haben wir mit eingehenden Tests der Seite und letzten Anpassungen für die Präsentation verbracht, sodass wir einen lauffähigen Dummy präsentieren konnten. Mittels Grunt konnten wir während der Präsentation praktisch live einen Dummy insoweit vorstellen, dass man durch Klicks die später zu erwartenden Reaktionen simulieren konnte.

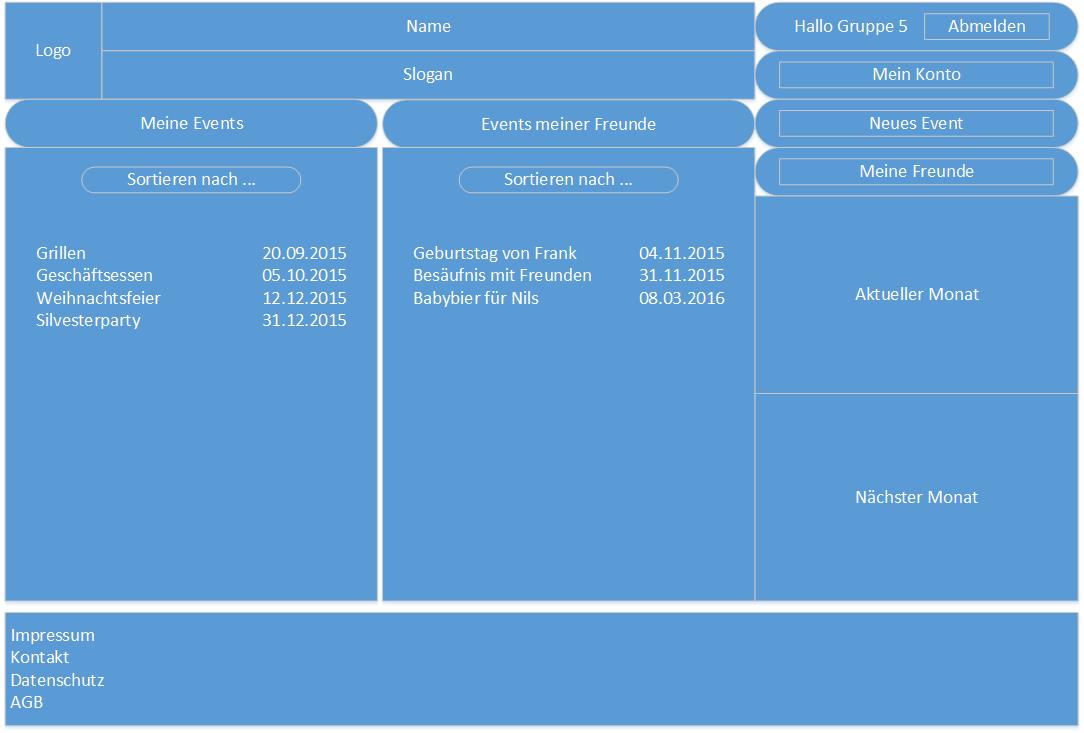
Unmittelbar nach der Präsentation begannen wir mit der Implementierung von JavaScript und PHP, welche die simulierten Funktionen durch echte ersetzen sollten. Es wurde eine MySQL Datenbank aufgesetzt, die notwendigen Tabellen angelegt und mittels erster PHP Testcodes probiert, diese Tabellen auszulesen – in einem weiteren Schritt wurden http-PostRequests mit JavaScript erzeugt, um die Interaktion zwischen JavaScript und PHP auszuprobieren. Anschließend wurden Stück für Stück die simulierten Funktionen ersetzt.

2.1 Verzeichnisstruktur

* **Comeet (Projektverzeichnis)**
  + .**tmp** (Output Verzeichnis für deploybare Webanwendung)
  + **dev** (Arbeitsverzeichnis)
    - **assets** (Bilder, favicon usw.)
    - **js**
    - **partials** (handlebars Bausteine)
    - **php**
    - **styles** (less Dateien)
* Gruntfile.js
* Package.json

2.2 Design

- folgt –



2.3 Datenbank

Benutzerdaten sowie der Inhalt des Blogs werden in einer Datenbank namens HTO01FLQZCHT\_2 gespeichert. In der Datenbankdesignphase sind fünf Tabellen angelegt worden. So existieren je eine Tabelle für

* Benutzerdaten (Users)
* Kontakte (Contacts)
* Events (Events)
* Mitbringsel usw. (Items)
* Teilnehmer (Attendees)

Die Tabellen werden nun ein wenig näher untersucht.

2.3.1 Attendees

--

-- Tabellenstruktur für Tabelle `Attendees`

--

CREATE TABLE IF NOT EXISTS `Attendees` (

`Event\_ID` int(11) NOT NULL,

`User\_ID` int(11) NOT NULL,

`status` varchar(50) default NULL,

KEY `Event\_ID` (`Event\_ID`),

KEY `User\_ID` (`User\_ID`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

2.3.2 Contacts

--

-- Tabellenstruktur für Tabelle `Contacts`

--

CREATE TABLE IF NOT EXISTS `Contacts` (

`User\_ID` int(11) NOT NULL,

`Contact\_ID` int(11) NOT NULL,

KEY `User\_ID` (`User\_ID`),

KEY `Contact\_ID` (`Contact\_ID`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

2.3.3 Events

--

-- Tabellenstruktur für Tabelle `Events`

--

CREATE TABLE IF NOT EXISTS `Events` (

`Event\_ID` int(11) NOT NULL auto\_increment,

`Title` varchar(255) NOT NULL,

`Description` varchar(5000) NOT NULL,

`Street` varchar(255) NOT NULL,

`Nr` varchar(30) NOT NULL,

`Postcode` varchar(30) NOT NULL,

`City` varchar(255) NOT NULL,

`CalendarDate` datetime NOT NULL,

`User\_ID` int(11) NOT NULL,

`MapLink` varchar(500) default NULL,

PRIMARY KEY (`Event\_ID`),

KEY `User\_ID` (`User\_ID`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO\_INCREMENT=68 ;

2.3.4 Items

--

-- Tabellenstruktur für Tabelle `Items`

--

CREATE TABLE IF NOT EXISTS `Items` (

`Item\_ID` int(11) NOT NULL auto\_increment,

`Event\_ID` int(11) NOT NULL,

`User\_ID` int(11) NOT NULL,

`Name` varchar(255) NOT NULL,

PRIMARY KEY (`Item\_ID`),

KEY `Event\_ID` (`Event\_ID`),

KEY `User\_ID` (`User\_ID`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO\_INCREMENT=510 ;

2.3.5 Users

--

-- Tabellenstruktur für Tabelle `Users`

--

CREATE TABLE IF NOT EXISTS `Users` (

`User\_ID` int(11) NOT NULL auto\_increment,

`Firstname` varchar(255) NOT NULL,

`Lastname` varchar(255) NOT NULL,

`Email` varchar(255) NOT NULL,

`Birthdate` varchar(50) NOT NULL,

`Picture` varchar(255) default NULL,

`Password` varchar(255) NOT NULL,

`Username` varchar(255) NOT NULL,

`Status` varchar(50) default NULL,

UNIQUE KEY `User\_ID` (`User\_ID`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO\_INCREMENT=81 ;

2.4 Funktion

Als zentrale Steuerdatei wird die index.html im Basisverzeichnis des Webservers verwendet. Mit HTML und CSS wird die Webseite generell konstruiert, mit JavaScript werden die Interaktionen zwischen User und Webseite durchgeführt. JavaScript kommuniziert anschließend mit PHP-Seiten, welche serverseitig die Verbindung zur MYSQL Datenbank herstellen und notwendige Datensätze aufbereiten bzw. bereitstellen. Die Ergebnisse werden anschließend mit JavaScript und HTML visualisiert.

2.5 Background-Engine

Wir haben versucht die Anwendung so einfach wie möglich zu gestalten was bedeutet, dass wir Aspekte wie Sicherheit und Angriffsschutz völlig außer Acht gelassen haben. Da wir ausschließlich mit HTML, JavaScript und PHP arbeiten, sind keine speziellen Anforderungen am Background vonnöten. Es werden PostRequests im JavaScript erzeugt, welche dann mittels PHP bearbeitet und mit einem HTTP Response beantwortet werden.

3. Implementierung

?Ich? bin bei der Implementierung den Weg vom statischen HTML in den Templates zum dynamischen Inhalt gegangen. Die Platzhalter wurden nach und nach zunachst mit statischem, spater dann mit dynamischem Code direkt aus der Datenbank versorgt. Die Entscheidung, ein Obertemplate mit mehreren Untertemplates zu verwenden, stellte sich schnell als gut heraus. Aufgrund der modularen Templatestruktur ist es moglich gewesen, die einzelnen Bereiche nach und nach zu implementieren. Ich habe mir folgende Programmlogik erdacht. Beim Aufruf der Seite werden zunachst die Blogeintrage fur den aktuellen Monat angezeigt. Per Auswahl im Content-Menu (rechtes Menu) lasst sich ein anderer Monat zur Anzeige bringen. Ein Blogeintrag besteht aus Uberschrift, dem Text sowie einigen Buttons. Blogger erhalten Buttons zum Editieren und Loschen der Blogeintrage. Blogger und registrierte Benutzer haben die Moglichkeit, Kommentare anzulegen, jeder hat die Moglichkeit, Kommentare anzuzeigen. Der Zugriff auf die Buttons wird anhand des Userlevels dynamisch berechnet, ebenso werden die Kommentarbuttons deaktiviert, sobald ein Kommentar angelegt wurde. Die Uberprufung der Anzahl erfolgt stets dynamisch beim Anzeigen der Blogeintrage.

Das Profil, welches fur Blogeintrage eingebunden wird, nennt sich blog.tpl. Die Kommentareanzeige sowie das formular zum Anlegen/Editieren neuer Kommentare wird nicht durch ein Template gesteuert, es wird durch Javascript generiert. Durch Parameterubergabe ist sichergestellt, dass die UserID und die PostID mit ubertragen werden. Dadurch konnen der Kommentier-Button direkt deaktiviert und die Kommentaranzahl direkt ubernommen werden.

Zudem ist eine kleine Benutzerverwaltung implementiert. Registrierte Benutzer (inklusive Blogger) haben die Moglichkeit, im Profil ihren Anzeigenamen sowie das eigene Passwort zu andern. Die Administratoren hingegen haben die Moglichkeit, den Aktivierungsstatus der User sowie samtliche Anzeigenamen und Passworter zu andern. Dazu wurden die beiden Templates profil.tpl und adminprofile.tpl angelegt.

Fur jede Aktion, die durch den Benutzer ausgeubt werden kann ist ein Template hinterlegt. Viele davon werden aber wenig verwendet. Activate.tpl wird beispielsweise bei der Aktivierung der User verwendet, setup.tpl nur beim Einrichten der Datenbank.

Der RSS-Feed wird ebenfalls durch ein Template realisiert, es werden hier die gleichen Daten wie beim Anzeigen des Blogs aus der Datenbank geholt und nur durch das Template in eine andere Darstellung gebracht.

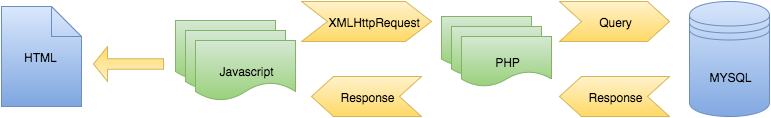
3.1 Abweichungen vom Entwurf

**Zu 1.4.6:**

Statt der Backend-Logik mit JavaScript haben wir uns dann doch für PHP entschieden, da Thomas hier schon Vorkenntnisse mitbrachte und der Server auf dem die Anwendung läuft PHP unterstützt, jedoch kein Node.js.

Wir setzten PHP ein, indem wir Anfragen vom JavaScript an PHP Dateien senden, welche dann Daten aus der DB holen und sie zurückliefern.

3.2 Programmlogik



3.3 Einschränkungen und Features

- folgt -

3.4 Quellcode

Der Quellcode befindet sich mit in dem Archiv, deshalb wird er hier nicht noch mal zusätzlich aufgeführt.

3.5 Installation

* Voraussetzung: Node.js und Grunt müssen installiert sein
* Datenbank muss vorhanden sein (siehe Exportdatei) -> in der config.php kann man die Einstellungen zur DB machen
* Es muss ein PHP Server laufen
* Im Verzeichnis wo das Gruntfile liegt, einfach "npm install" ausführen
* dann "grunt" ausführen (hbs und less werden kompiliert) (Es wird ein lokaler Server gestartet, das ist nur für Entwicklung)
* Hat man den grunt einmal gestartet und die Dateien kompiliert, dann kann man "grunt copy" ausführen
* Dadurch liegen nun ALLE benötigten Dateien im .tmp Verzeichnis, so wie sie auf den PHP Server geladen und dort ausgeführt werden können