Installations- und Bedienungsanleitung für die Meißner Webanwendung

von

Christian Meter

Inhaltsverzeichnis

Kapitel 1

Installation der Meißner Anwendung

1.1 Voraussetzungen

Dieses Installationsskript wurde für Debian entwickelt. Unter anderen Distributionen sind ähnliche Schritte notwendig, jedoch werden diese nicht von dem Shell Script übernommen.

Testsystem und Dauer der Installation Auf dem Testsystem war ein Kern einer Intel i5 2450M, 1 GB DDR3 Arbeitsspeicher und eine DSL 16000 Internetleitung verfügbar. Als Hostsystem wurde Linux Mint 15 verwendet, welches mit *debootstrap* ein unverändertes Debian System installiert hatte. Über *chroot* wurde dann ein Benutzer mit *sudo*-Rechten eingerichtet, welcher das Shell Script schließlich aufgerufen hat.

Für die Installation werden ca. 320 MB heruntergeladen. Darin enthalten sind allerdings alle notwendigen Pakete für einen Webserver und WebSocket Server sowie *gcc, g++, make und configure*, um node*node.js* aus den Quellen zu kompilieren.

Mit dieser Konfiguration wurden für die Installation 20 Minuten benötigt.

1.2 Beschreibung

Die Installation beginnt mit einem Update der Distribution auf die aktuellste Version. Danach wird ein LAMP-Server (Linux-Apache-MySQL-PHP) installiert und konfiguriert, gefolgt von der Installation der aktuellsten node.js Version. Letzteres muss vom Server kompiliert werden und nimmt daher einige Zeit in Anspruch.

1.3 Installation von LAMP

Zu Beginn der Installation ist es notwendig sich im **gleichen Verzeichnis** zu befinden, wie das Shell Script und der Ordner *meissner*. Von dort aus führt man folgenden Befehl mit root-Rechten aus:

```
sudo sh install.sh
```

sudo ist hier notwendig, da einige Pakete installiert und kompiliert werden müssen.

1.3.1 MySQL und phpmyadmin

MySQL Während der Installation muss nur das Passwort für den *root* Benutzer der MySQL Datenbank gesetzt werden.

phpmyadmin Auch hier müssen nur ein Passwort für phpmyadmin und anschließend das root-Passwort für die MySQL Datenbank eingegeben werden.

1.4 Installation von node.js

Da es keine vorgefertigten Pakete von node.js gibt, muss das Skript sich die aktuellste Version von dem Server des Entwicklers herunterladen und anschließend selbst compilieren. Dafür werden Pakete wie *make, configure, gcc, g++* benötigt, die schon im ersten Schritt dieser Installation installiert wurden.

Das Installationsskript startet automatisch den Download von node in das Verzeichnis */tmp* und führt dort alle nötigen Schritte aus, um node zu installieren. Hierbei ist keine Anpassung durch den Benutzer notwendig.

1.5 WebSocket Server

1.5.1 Ausführung

Die Startdatei für den WebSocket Server *socket-server.js* befindet sich in *meissner/app/webroot/websocket/*. Mit folgendem Befehl wird der WebSocket Server gestartet:

```
node socket-server.js
```

1.5.2 Bearbeitung

Die oben genannte Startdatei ist auch die Datei, in der der komplette WebSocket Server eingestellt werden kann. Dafür muss nur diese Datei bearbeitet und anschließend der WS Server neugestartet werden.

1.5.3 Port

Der Port des WebSocket Server ist: 9999

Wenn der Port geändert werden soll, so müssen die Änderungen sowohl im AppController als auch in OtherComponent angepasst werden.

1.6 Einrichtung der MySQL Datenbank

Zur Einrichtung der MySQL Datenbank muss folgende URL aufgerufen werden:
http://localhost/meissner/setup

Das Webinterface leitet durch die 3-Schritte-Einrichtung, in der die Adresse des Servers, der Benutzername und das Passwort der Datenbank eingegeben werden müssen.

In der Regel sind es folgende Einstellungen:

Host: localhost

User: root

Pass: <your password>

Danach kann die Anwendung mit *http://localhost/meissner* aufgerufen werden.

Kapitel 2

Verwendung der Anwendung

2.1 Veranstaltungen

Unter *Events* können können Veranstaltungen angelegt oder verwaltet werden. Sind Veranstaltungen verfügbar, so werden sie in einer Liste untereinander angezeigt.

2.1.1 Hinzufügen

Mit *Add Event* kann eine Veranstaltung erstellt werden. Dafür sind nur der Titel und eine Beschreibung nötig.

2.1.2 Bearbeiten

Titel und Beschreibung können hier bearbeitet werden. Danach folgen die eventspezifischen Felder, welche mit *Add Column* angelegt werden können.
Zuletzt erscheint eine Liste von Benutzern, die dem Event zugeordnet sind. Dabei gibt es zwei Optionen:

Edit Hinter diesem Link erscheinen alle eventspezifischen Felder, die unter *Add Column* definiert wurden. Hier können die Daten des Benutzers in die speziellen Felder der Veranstaltung eingetragen werden.

Delete Entfernt den Benutzer aus dieser Veranstaltung.

2.2 Benutzer

Die Erstellung und Zuweisung eines Benutzers zu einer Veranstaltung erfolgt an dieser Stelle.

2.2.1 Hinzufügen

Unter *Add User* kann ein neuer Benutzer hinzugefügt werden. Benutzername und Passwort müssen hierbei definiert werden. Danach folgt die Berechtigung des Benutzers.

Allowed to login Der Boolean *Allowed to login* gibt an, ob der Benutzer nur zur Verwaltung in die Anwendung integriert werden soll oder auch selbst Daten einsehen kann.

Select events Wenn Veranstaltungen verfügbar sind, können hier mehrere Events ausgewählt werden, zu denen der Benutzer zugeordnet werden soll. Das schaltet dann die Bearbeitungsmöglichkeit innerhalb der Veranstaltung frei, in der die eventspezifischen Felder definiert werden.

2.2.2 Bearbeiten

Mit Klick auf *Edit* können alle Eigenschaften, die unter *Add User* schon erschienen, erneut bearbeitet werden.

2.2.3 Entfernen

Delete entfernt den Benutzer und alle seine Daten aus der Datenbank.

2.3 Geopositionierung

Eine Google Map markiert die Positionen aller eingeloggten Benutzer. Diese automatisiert sich selbstständig und passt den Kartenausschnitt entsprechend der Positionen der Benutzer an.

2.3.1 Bedienelemente

Refresh Page Aktualisiert die Seite. Dieser Button ist notwendig, weil die Meißner Anwendung als Web-App installierte Webanwendung unter iOS keine Möglichkeit bietet die Seite zu aktualisieren, falls die Verbindung zum WebSocket Server unterbrochen wurde oder die Seite einfach neu geladen werden muss.

Clear Overlays Diese Option entfernt alle von der Anwendung hinzugefügten Marker und Infoboxen und zeigt nur noch die normale Google Map an.

Show Overlays Hierdurch werden wieder alle Marker und Infoboxen angezeigt.

Autozoom Hier hat der Benutzer die Möglichkeit die automatische Zoomfunktion der Map zu deaktivieren. Die Karte wird bei jedem Update neu ausgerichtet und das kann zu einem unangenehmen „Springen “ kommen, weshalb diese Funktion sinnvoll ist.

2.3.2 Markierungen

Eine Markierung auf der Karte kann angeklickt werden. So wird in einer Infobox der Benutzername angezeigt.

2.4 Statistiken

Die Statistiken werden vollständig automatisch generiert.

2.4.1 Allgemeine Informationen

Unter *Overview* erscheint die Gesamtzahl der Benutzer und Veranstaltungen. Mit einem Klick auf *Show / Hide Chart* wird mit einem Google Chart angezeigt, wie viele Benutzer den einzelnen Veranstaltungen zugeordnet sind.

2.4.2 Eventspezifische Statistiken

Zu jeder Veranstaltung werden alle Daten in einer Tabelle übersichtlich dargestellt. Wenn es mehr als einen spezifischen Wert zu einem Feld gibt, wird unterhalb der Tabelle eine Grafik generiert, welche diese Werte anschaulich darstellt.

2.5 Chat

Hier können alle Benutzer in einem globalen Channel miteinander chatten. Die Nachrichten werden in Echtzeit an alle verbundenen Clients verschickt.

Für die Verwendung von Chats muss eine WebSocket Verbindung bestehen.

Kapitel 3

Eigene Anpassungen

3.1 Erzeugung eines neuen Models

Um einen eigenen Bereich zu erzeugen, der später in der App verfügbar sein soll, können eigene Model, View und Controller entwickelt werden. Diese können im Ordner

meissner/app/{Model, View, Controller}

angelegt werden.

Damit der View nun in das Layout eingebunden wird, muss der entsprechende Eintrag noch in der Navigation verlinkt werden:

meissner/app/View/Layouts/{nav.ctp, mobilenav.ctp}

3.2 Bearbeitung des Layouts

Das Layout der Anwendung kann unter

meissner/app/View/Layouts/{default.ctp, mobile.ctp}

verändert werden. Dort befindet sich das Grundgerüst der Webanwendung. Stylesheets, JavaScripte, Navigation und so weiter können hier für alle Views eingebunden werden.

3.2.1 Bedeutung von webroot/

In dem Ordner webroot werden Grafiken, Skripte, Stylesheets und andere Dateien gespeichert, die der Anwendung zur Verfügung gestellt werden sollen.

meissner/app/webroot

Von diesem Ort geht die Anwendung aus, wenn etwas eingebunden werden soll, das bedeutet wenn man im Layout die Datei *css/stylesheet.css* aufrufen möchte, darf diese nicht in *meissner/app/View/Layouts/css/stylesheet.css* liegen, sondern eben in *meissner/app/webroot/css/stylesheet.css*. Das gilt auch für alle Bilder, Skripte und so weiter. Das hat den Vorteil, dass alle Dateien zentral in einem Ordner gespeichert sind und nicht verteilt im cakePHP Framework zu suchen sind.

3.2.2 Designs mit CSS

Stylesheets können entsprechend im webroot-Verzeichnis gefunden werden:

meissner/app/webroot/css

Standardmäßig laden die Desktop- und Mobilversion der Anwendung zuerst die *main.css*, worin die gemeinsamen Styles definiert sind. Danach werden layoutspezifische Stylesheets nachgeladen, welche ebenfalls in diesem Ordner zu finden sind.

3.2.3 Einbindung von JavaScript

Wie schon beschrieben, werden Skripte auch in *webroot/js* gespeichert. Dann können Sie im Layout oder in den Views der App einfach mit folgendem Befehl geladen werden:

```
<script type="text/javascript" src="js/javascript.js">
</script>
```

Unabhängig des Pfades, in dem sich der View oder das Layout befindet, wird mit diesem Befehl immer im webroot-Verzeichnis nach der angegebenen Datei gesucht. Das erleichtert das Entwickeln neuer Views, da der Entwickler nicht mehr nachdenken muss, in welchem Ordner er sich gerade befindet.