

Plattform zur Vermittlung von Nikolausdarsteller in Word-  
Press

Platform for the mediation of nicholas performers in WordPress

Philipp Lippold

Bachelor-Abschlussarbeit

Betreuer: Prof. Dr.-Ing. Georg Schneider

Trier, 01.08.2018

---

## Kurzfassung

Die Abschlussarbeit Entwicklung einer Webanwendung zur *Vermittlung von Nikolausdarsteller in WordPress* wurde für den Bund der Katholischen Jugend (BDKJ) in Köln entwickelt. Nutzer können auf der Webseite die Darstellersuche nutzen und damit Nikolausdarsteller in ihrer Umgebung suchen. Jeder Nutzer besitzt seinen eigenen Nutzerbereich und die Möglichkeit sich als Nikolausdarsteller zu präsentieren. Darüber hinaus werden Grundlegende Konzepte, sowie die Realisierung der Arbeit erläutert. In einem weiteren Kapitel werden die Sicherheitskonzepte und die Gründe der Verwendung erklärt. Im Abschluss wird eine Zusammenfassung, sowie Ausblicke beschrieben, wie das Plugin in der Zukunft erweitert werden könnte.

---

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b> .....	1
1.1	Motivation .....	1
1.2	Ziele der Arbeit .....	1
<b>2</b>	<b>Verwendete Systeme</b> .....	3
2.1	Werkzeuge .....	3
2.2	Externe Plugins .....	4
2.3	API .....	4
<b>3</b>	<b>Konzept</b> .....	5
3.1	Grundlegende Anforderungen .....	5
3.2	Rollen .....	5
3.3	Aufbau des Plugins .....	6
3.4	Navigation .....	10
<b>4</b>	<b>Realisierung</b> .....	12
4.1	Installation .....	12
4.2	Speicherung von Daten .....	13
4.3	Navigation .....	13
4.4	Registrierung und Login .....	13
4.5	Startseite .....	14
4.6	Darstellersuche .....	15
4.7	Filterung .....	16
4.8	Nutzerbereich .....	16
4.9	Nutzerprofile .....	20
<b>5</b>	<b>Implementierung</b> .....	22
5.1	Einbindung .....	22
5.2	Hauptseite .....	23
5.3	Nutzerbereich .....	31
5.4	Nutzerprofile .....	36
5.5	Automatische Löschung von inaktiven Nutzern .....	37
<b>6</b>	<b>Sicherheitskonzept</b> .....	40

---

<b>7 Zusammenfassung und Ausblick .....</b>	<b>42</b>
<b>Literaturverzeichnis .....</b>	<b>43</b>
<b>Erklärung der Kandidatin / des Kandidaten .....</b>	<b>44</b>

---

## Abbildungsverzeichnis

3.1	Architektur des Plugins .....	7
3.2	Navigation im Plugin .....	10
3.3	Navigation im Administratorbereich .....	11
4.1	ER-Diagramm der Nutzertabellen .....	13
4.2	Startseite Gesamtbild .....	14
4.3	Darstellersuche in Koblenz .....	15
4.4	filtern von Darstellern .....	16
4.5	Teil 1 Nutzerbereich .....	17
4.6	Teil 2 Nutzerbereich .....	18
4.7	Teil 3 Nutzerbereich .....	19
4.8	Profil nicht freigeschaltet .....	20
4.9	Freischaltanfrage .....	21
5.1	Arbeitsradius .....	23

# Einleitung

## 1.1 Motivation

Der Bund der Deutschen Katholischen Jugend (BDKJ) in Köln bietet seit zirka zehn Jahren die sogenannte *Nikolausaktion*<sup>1</sup> an. Über deren Webseite vertreibt der BDKJ den Schokoladennikolaus und organisiert bundesweite Nikolaustreffen. Das Ziel dieser Aktion ist, den Menschen zu zeigen, wie diese ihre Adventsfeier gestalten können. Heutzutage geraten Identifikationsfiguren für soziales Handeln, christliche Nächstenliebe und abendländische Kultur mehr und mehr in Vergessenheit laut [Kö]. Ebenfalls sollen den Menschen die Werte des Schenkens und des Heiligen Nikolaus gelehrt werden. In den Nikolausschulen werden Personen geschult um zertifiziert als Nikolaus auftreten zu können. Der BDKJ wünschte sich hierfür eine Plattform, damit die einzelnen Nikolausdarsteller leicht über das Internet vermittelt werden können.

## 1.2 Ziele der Arbeit

Ziel ist es dem BDKJ Köln eine Webanwendung als Plugin in WordPress zu entwickeln, damit Mitgliedern des *Nikolausbündnis* eine *Plattform zur Vermittlung von Nikolausdarsteller* zur Verfügung steht. Dabei soll diese Anwendung wartungsfrei laufen. Nutzer müssen die Möglichkeit haben auf einer Karte ihre Position darstellen zu können. Eine Darstellersuche soll ebenfalls implementiert werden, damit Nutzer auch Darsteller in ihrem Umkreis oder für spezielle Events finden können. Außerdem sollen bestimmte Sicherheitsaspekte berücksichtigt werden, damit potenziellen Angreifern keine Angriffsfläche geboten wird. Dafür müssen Sicherheitsrichtlinien der OWASP (Open Web Application Security Project) eingehalten und umgesetzt werden. WordPress ist so konzipiert, dass auch Laien mit wenig Aufwand diese Webanwendung betreiben können. Das Plugin soll ebenfalls nach diesem Prinzip funktionieren. Darüber hinaus muss es möglich sein, dass das Plugin in ein bestehendes System integriert werden kann. Außerdem soll es auch ohne Updates auskommen können. Auf Design wird einen gewissen Wert gelegt, da dieses

---

<sup>1</sup> <http://www.nikolausaktion.org/>

---

Plugin ein fertiges Produkt darstellt und auch beispielsweise mobilen Anwendern zur Verfügung stehen soll.

## Verwendete Systeme

### 2.1 Werkzeuge

1. **WordPress**<sup>1</sup> ist ein Content Management System, dass dazu dient Inhalte (Content) zu verwalten, meist in Form von Webseiten. Es gibt unterschiedliche Nutzergruppen mit unterschiedlichen Rechten, beispielsweise Autoren oder Administratoren. Das Programm ist leicht ausgestattet, bietet aber die Möglichkeit individuell angepasst zu werden mittels Plugins.[Fal]
2. **PHP**<sup>2</sup>(Hypertext Preprocessor) ist eine *Open-Source*-Skriptsprache, welche in HTML eingebettet werden kann. Im Gegensatz zu Javascript und anderen clientseitigen Skriptsprachen wird PHP serverseitig ausgeführt. Einvorteil davon ist, dass die Nutzer den Quellcode nicht sehen können. [Gro]
3. **jQuery**<sup>3</sup> ist eine umfangreiche Javascript-Bibliothek. Durch das Framework ist es möglich, komplexe Javascript-Funktionen wie DOM<sup>4</sup>-Syntax Manipulation oder Event-Handling leicht umzusetzen. Darüber hinaus werden lange Javascript-Funktionen in jQuery deutlich verkleinert und AJAX-Funktionalitäten beschleunigt. [aH]
4. **Ajax** (Asynchronous JavaScript and XML) wird verwendet, um im Hintergrund Daten mit dem Webserver auszutauschen. Der Vorteil ist hier von Ajax, dass nicht immer wieder ein neues HTML Dokument angefordert werden muss. Nur die zur Zeit benötigten Daten werden nachgeladen, was zu einer besseren Benutzerfreundlichkeit führt. [Mol]

---

<sup>1</sup> [de.wordpress.com/](http://de.wordpress.com/)

<sup>2</sup> <https://jquery.com/>

<sup>3</sup> <http://php.net/>

<sup>4</sup> Das *Document Object Model* repräsentiert den strukturellen Aufbau von HTML Dokumenten. Dadurch werden Webseiten mit Scripts und Programmiersprachen verbunden.[Moz]



## 2.2 Externe Plugins

1. **Login No Captcha reCAPTCHA**<sup>5</sup> ist ein externes Plugin von Robert Peake, welches im Login und der Registrierung ein Google Captcha einfügt. Ebenfalls können keine automatischen Registrierungen mehr erfolgen und es werden dadurch Brute-Force-Angriffe<sup>6</sup> verhindert.

## 2.3 API

1. **Google Maps**<sup>7</sup> ist eine Kartenanwendung von Google LLC, die es ermöglicht auf einer interaktiven Weltkarte Standorte zu markieren. Darüber hinaus werden mehrere Programmschnittstellen zu Verfügung gestellt, womit es z.B. möglich ist die geografische Daten von einzelnen Standorten zu bekommen.[LLC]
2. **GeoNames**<sup>8</sup> ist eine Datenbank mit über 11 Millionen Einträgen, welche alle Länder abdeckt. Dazu bietet es bestimmte Webservices, wie beispielsweise "Find nearby populated place". Dies ist eine API, welche alle Städte in einem gewissen Umkreis, mit einer bestimmten minimalen Einwohnerzahl zurückliefert.[geo]

---

<sup>5</sup> <https://wordpress.org/plugins/login-recaptcha/>

<sup>6</sup> Bei einem Brute-Force Angriff werden wahllos Passwörter getestet, um auf einen Benutzer Account Zugriff zu erlangen, womit auch sehr häufig Anfragen an den Server gestellt werden[Tut]

<sup>7</sup> <https://www.google.de/maps>

<sup>8</sup> <http://www.geonames.org>

## Konzept

### 3.1 Grundlegende Anforderungen

- Plattform für die Vermittlung von Nikolausdarstellern
- Einbindung als Plugin in *WordPress*
- Plugin soll den **Wordpress Sicherheits Implementierungs Richtlinien**<sup>1</sup> des OWASP<sup>2</sup> entsprechen
- Nikolausdarsteller sollen auf einen selbst ausgewählten Punkt in einer Karte abgebildet werden
- Der Arbeitsradius eines Nikolaus soll auf der Karte ersichtlich sein
- Das Plugin soll auch von Laien bedient werden können und simpel gehalten, aber dennoch ansprechend sein
- Ein Nikolaus kann bestimmte Arbeitsbereiche festlegen
- Personen, welche die Nikolausschule besucht haben sollen ihr Zertifikat hochladen können
- Gewisse Profilinformationen, zum Beispiel Name und Adresse sollen bis zur Freischaltung durch des Anfragenden versteckt bleiben
- Ein Benutzer erhält nach vier Monaten Inaktivität eine Erinnerung, damit sich dieser einloggt oder dieser automatisch gelöscht wird

### 3.2 Rollen

WordPress besitzt ein eigenes System zu Rollenverwaltung mit einem breiten Spektrum an Untergliederungen von Administrator, über Autoren, bis hin zu Abonnenten. In diesem Plugin wurden jedoch lediglich die Rollen des Administrators und des Abonnenten genutzt. Bei den Abonnenten unterscheidet das Plugin zwischen registrierten Nutzern und Nikolausdarstellern. Die nachfolgenden Rollen sind im System vertreten:

- Unregistrierte Nutzer: Können die Darstellersuche auf der Webseite nutzen und im gewissen Rahmen Profile begutachten, jedoch keinen Kontakt zu anderen Nutzern aufnehmen.

---

<sup>1</sup> [https://www.owasp.org/index.php/OWASP\\_Wordpress.Security.Implementation.Guideline](https://www.owasp.org/index.php/OWASP_Wordpress.Security.Implementation.Guideline)

<sup>2</sup> Open Web Application Security Project kurz OWASP ist eine non-Profit Organisation, welche es als Ziel hat die Sicherheit von Software zu erhöhen.

- **Registrierte Nutzer:** Können in ihrem Profil verschiedene Einstellungen treffen und dort den Status als Nikolausdarsteller wählen. Im Profil ist es möglich ein Profilbild hochzuladen oder wieder zu entfernen. Des weiteren können Nutzer sich untereinander kontaktieren und Profile melden. Ebenfalls können Nutzer ihr Profil für andere Nutzer freischalten. Im gleichen Zug ist es für sie möglich, den Zugang zu privaten Daten zu verwehren und alle Freischaltungen wieder rückgängig zu machen.
- **Nikolauser:** können sich auf der Karte anzeigen lassen für andere Nutzer, dabei können sie diesen Standort auf der Karte selbst bestimmen. Ein Zertifikat hochzuladen, zu entfernen sowie bestimmte Arbeitsbereiche festzulegen. Die Darsteller haben jederzeit die Möglichkeit ihren Nikolausstatus aus dem Profil zu entfernen.
- **Administratoren:** Können im Wordpress Administratorbereich Nutzer löschen und andere Einstellungen am WordPress System verändern. Alle registrierten Administratoren bekommen ebenfalls eine E-Mail, falls ein Profil gemeldet wird.

### 3.3 Aufbau des Plugins

Das Plugin besteht aus mehreren Komponenten, die alle mit der WordPress Anwendung kommunizieren 3.1. Der Hauptseite, dem Nutzerbereich und den Nutzerprofilen. Alle drei Komponenten müssen dementsprechend separat eingebunden werden. Von WordPress gegebene Methoden verändern die unterschiedlichen Komponenten der Wordpress Anwendung oder greifen auf Daten in der Datenbank zu

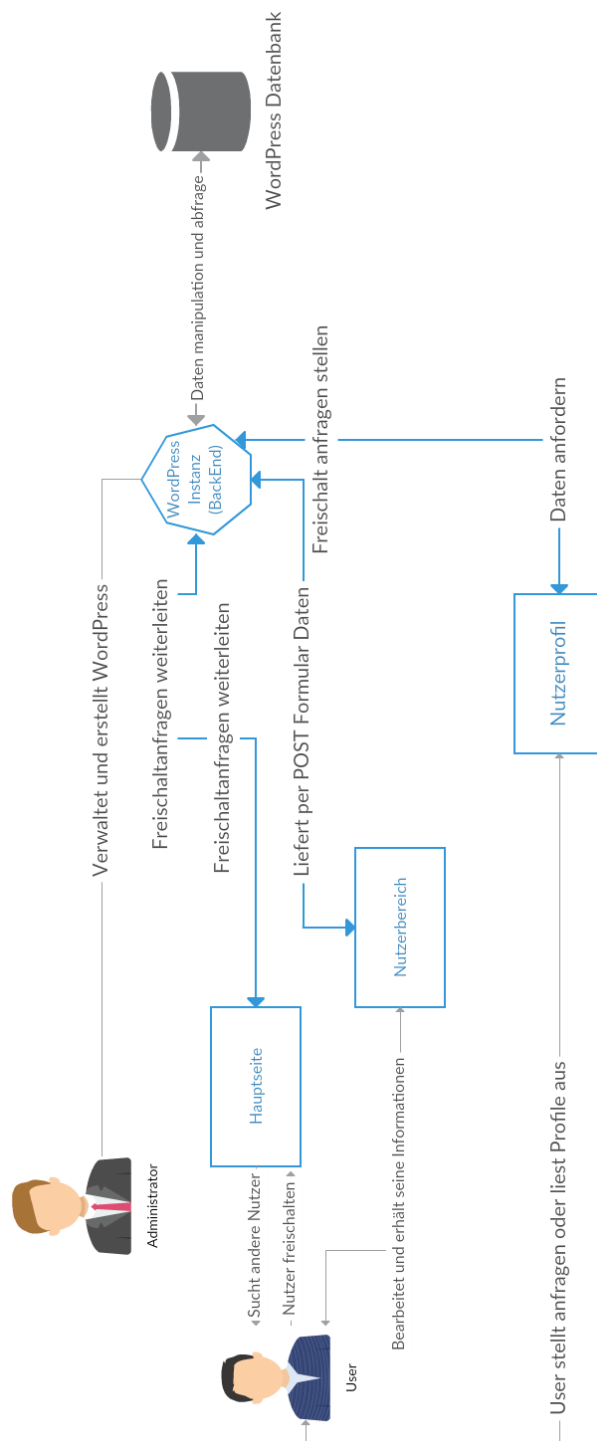


Abb. 3.1. Architektur des Plugins

### 1. Startseite

Auf der Startseite sehen die Besucher die von Google Maps eingebundene Karte sowie alle eingetragenen Nikolaus samt Profilbild. Falls ein Nikolaus kein Profilbild hat, wird ihm hier ein standardmäßiges gegeben. Des Weiteren kann

hier eine neue Suche gestartet werden, indem erst eine Stadt und als nächstes der Radius angegeben wird. Anschließend wird in die Karte je nachdem wie hoch der Suchradius eingestellt wurde, hereingezoomt und auf die gesuchte Stadt zentriert. Außerdem scrollt das Plugin das Browser Fenster nach oben auf die Karte. Bei einer Suche werden nur alle Nikolause angezeigt, welche sich in dem gewählten Umkreis zur gesuchten Stadt aufhalten. Nikolaue auf der Karte befinden sich gegebenenfalls in einem transparenten Kreis, welcher ihren Arbeitsbereich darstellt. Danach werden die Tags zur Filterung freigegeben, welche vorher nicht anklickbar waren und es kann weiter gefiltert werden. Falls eine neue Suche gestartet wird, werden die Tags zur Filterung wieder gesperrt. Unter der Darstellersuche ist ebenfalls eine Liste mit Nutzern, welche den aktuellen Suchkriterien entsprechen. Durch Klicken auf einen Darstellern auf der Karte oder in der Liste wird in einem neuen Fenster das Profil dieses Darstellers aufgerufen. Ebenfalls auf der Startseite werden hier für die eingeloggten Nutzer die Freischaltungsanfragen angezeigt. Außerdem können auch alle Freischaltungen rückgängig gemacht werden.

## 2. Nutzerbereich

Im Nutzerbereich können die Nutzer ihre Daten wie Name, Adresse und Profilbild anpassen. Darüber hinaus können die Nutzer festlegen, ob sie aktiv als Nikolaus auftreten möchten. Nutzer können eine Beschreibung über sich angeben, welche öffentlich im Profil angezeigt wird. Über eine weitere Karte ist es hier möglich als Nutzer sich den genauen Punkt auszusuchen. An diesem ist der Nutzer auf der Karte zu sehen. Dies funktioniert durch Klicken auf die Karte. Weiterhin ist es möglich einen Arbeitsradius zu definieren. Dadurch sehen andere Nutzer auf der Karte den Arbeits-, beziehungsweise Reisebereich in Form eines blauen Kreises. Auch die Zertifikate der Nikolausschule können hochgeladen und entfernt werden. Außerdem kann das Arbeitsumfeld in dem Darsteller arbeiten möchten angegeben werden, beispielsweise in einer Kirche oder einer Schule.

## 3. Nutzerprofil

Im Nutzerprofil werden alle Daten eines Nutzers angezeigt. Jedoch können andere Benutzer gewisse Informationen nur dann sehen, wenn sie vorher vom Eigentümer des Profils freigeschaltet wurden. Eine weitere Funktion auf dem Nutzerprofil für andere Benutzer ermöglicht es, über ein Formular eine Nachricht an diese zu senden. Für den Fall, dass sich auf einem Profil falsche oder unangemessene Inhalte befinden, gibt es einen Melde Button. Durch Klicken auf den Melde Button werden E-Mails an alle im Wordpress hinterlegten Administratoren gesendet und diese auf das Profil aufmerksam gemacht. Dadurch können Administratoren dann den Nutzer des Profils kontaktieren oder das Profil löschen.

## 4. Registrierung

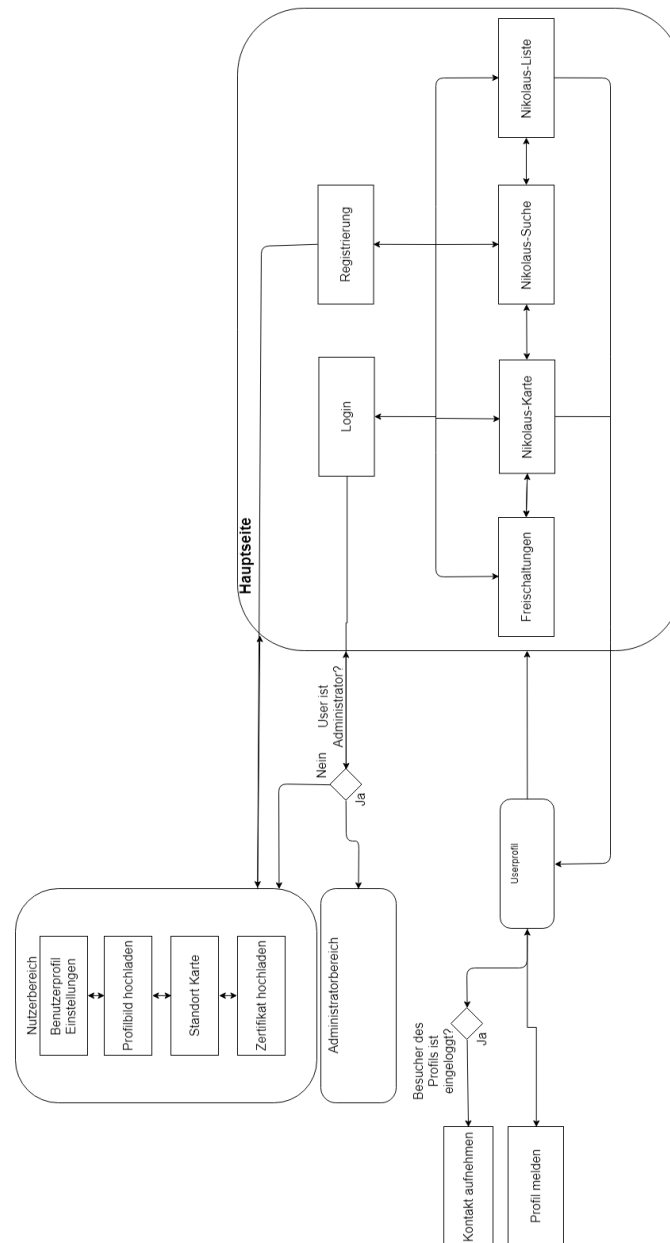
Bei der Registrierung müssen die folgenden Daten angegeben werden:

- E-Mail-Adresse
- Benutzername

Beides kann später als Login verwendet werden. Dem Nutzer wird eine E-Mail mit Password zugeschickt, welches er im Nutzerbereich ändern kann. Durch ein externes Plugin muss der Nutzer vorher ein Captcha lösen.

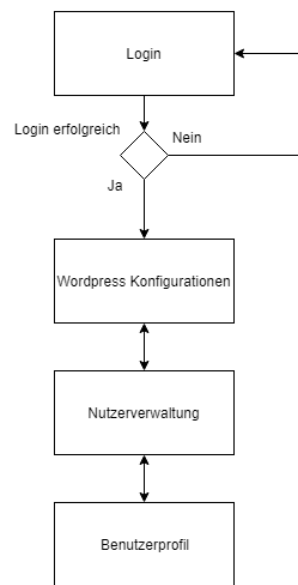
### 3.4 Navigation

Das nachfolgende Information Architecture (IA) Diagramm 3.2 und 3.3 nach Jesse James Garrett<sup>3</sup> spezifiziert wie die Navigation durch das Plugin abläuft:



**Abb. 3.2.** Navigation im Plugin

<sup>3</sup> <http://www.jjg.net/ia/visvocab/german.html>



**Abb. 3.3.** Navigation im Administratorbereich



## Realisierung

Das folgende Kapitel handelt über die Realisierung des Plugins. Es werden Funktionalitäten und Aussehen beschrieben. Außerdem wird die Benutzung von einzelnen Elementen erläutert.

### 4.1 Installation

Das Plugin besteht aus mehreren Komponenten. Die erste Komponente ist das sogenannte Main-Plugin, welches die Startseite sowie die Karte und die Darstellungen anzeigt. Das zweite Plugin ist für die Veränderung des standardmäßigen Nutzerbereiches zuständig und speichert die Daten in die Datenbank. Die dritte Komponente ist ein Theme, welches eine modifizierte Datei enthält, die dafür sorgt, dass Benutzerprofile angezeigt werden können.

## 4.2 Speicherung von Daten

Folgendes ER-Diagramm beschreibt den Aufbau der MySQL-Datenbank der WordPress Anwendung. Für das Plugin sind nur 2 Tabellen relevant 4.1. Die Daten werden hier in der wp\_usermeta Tabelle gespeichert. Jeweils mit einem bestimmten **Metakey** und einem dafür passenden **Metavalue**. Als Beispiel wird im Metakey der Vorname also in dem Fall "Philipp" gespeichert und beim Zugriff, welcher über die **Nutzer-Id** erfolgt ausgegeben oder verändert.

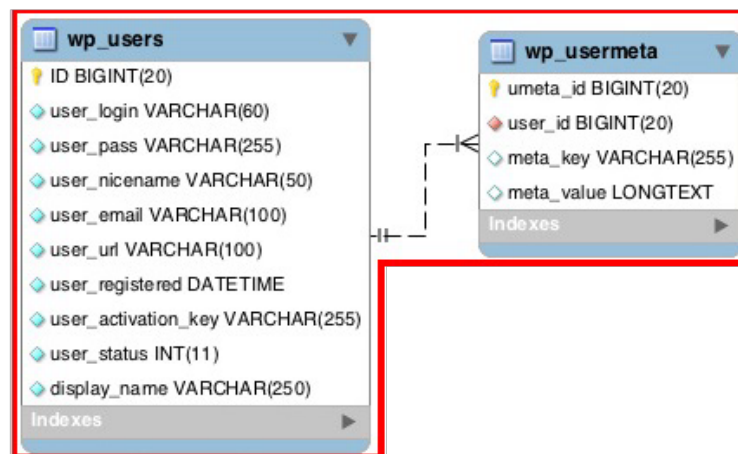


Abb. 4.1. ER-Diagramm der Nutzertabellen

## 4.3 Navigation

Die Navigation erfolgt zum Einen mithilfe eines Menüs, welches in WordPress eingebunden werden muss und zum Anderen durch die Aktion des Nutzers. Das Menü muss einmalig erstellt werden, wodurch das geladene Plugin neue Funktionalitäten hinzufügt.

Die beiden Funktionalitäten des Logins und der Registrierung sowie die Datumsanzeige werden durch das Plugin hinzugefügt.

## 4.4 Registrierung und Login

Die Registrierung der Benutzer auf der Website erfolgt sehr simpel über das interne WordPress Registrierungsformular. Dabei muss ein Nutzer zwangsläufig eine E-Mail-Adresse angeben, damit eine Email mit einem temporären Zugangspasswort versendet wird.

Der Benutzerlogin erfolgt über das standardmäßige Formular von WordPress, jedoch wird hier zum Schutz gegen Brute-Force-Attacken ein externes Plugin namens "Login No Captcha reCAPTCHA" von Robert Peake benutzt. Ein Vorteil dieses Plugins ist, dass dieses in Zukunft noch mit Updates versorgt wird.

## 4.5 Startseite

Die Startseite des Plugins 4.2 umfasst ein horizontales Menü, welches sich am oberen Rand befindet und in mobiler Ansicht zu einem Burger-Menü wird. Darunter kommt der Freischaltungsbereich, hier werden Nutzeranfragen von Nutzern, welche Zugriff auf das Profil des eingeloggten Nutzers erlangen möchten angezeigt. Der Freischaltungsbereich ist nur für registrierte und eingeloggte Nutzer sichtbar. Nun folgt die Google Map, welche alle Nikolausdarsteller mit ihren Wunschpositionen darstellt. Darunter folgen Tags zur Filterung und diese werden erst zur Verwendung freigegeben, wenn der Nutzer eine Darstellersuche gestartet hat. Unter der Filterung liegt die Darstellersuche, welche eine Stadt und eine Entfernung benötigt. Im Abschluss folgt eine Liste, hier werden die Nikolausdarsteller aufgelistet, welche der Filterung und der Suche entsprechen.

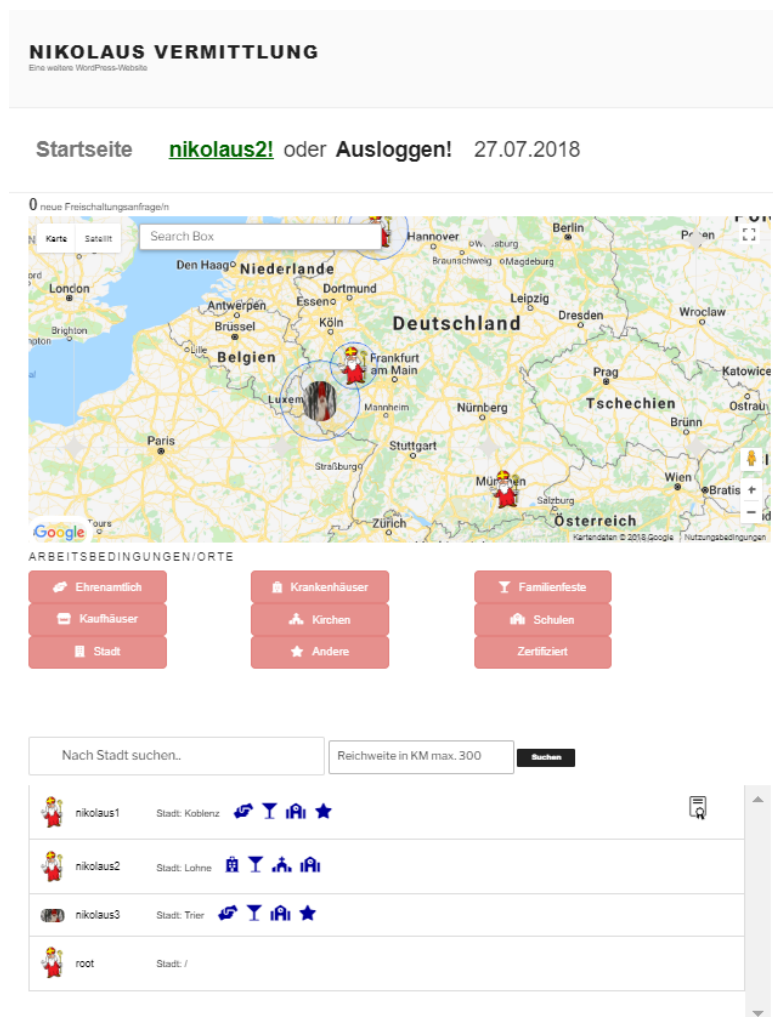
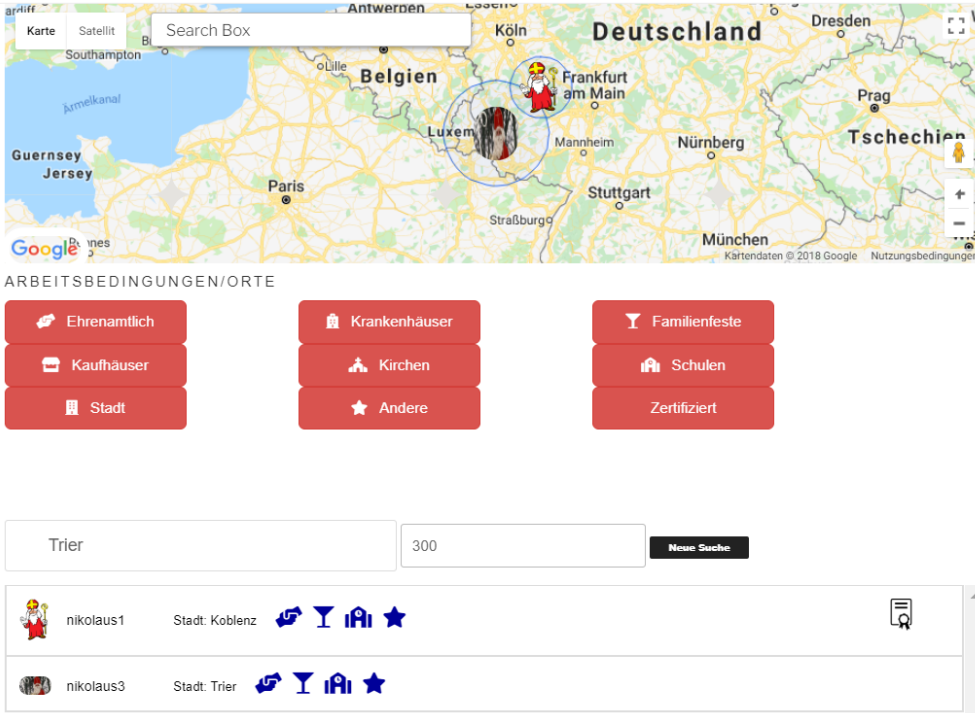


Abb. 4.2. Startseite Gesamtbild

## 4.6 Darstellersuche

Bei der Darstellersuche wird eine Stadt und ein gewisser Radius in Kilometer angegeben. Klickt der Nutzer den Suchbutton, wird die Karte auf diese Stadt zentriert und dazu je nachdem wie hoch der Kilometerbereich ist in die Karte hineingezoomt. Ebenfalls werden Darsteller, welche sich nicht im passenden Radius der gesuchten Stadt aufhalten ausgeblendet 4.3. Dementsprechend wird auch die Liste angepasst. Nachdem eine Suche gestartet wurde, werden auch die Tags, welcher der genaueren Filterung dienen freigeschaltet.

Startseite [nikolaus2!](#) oder **Ausloggen!** 27.07.2018



ARBEITSBEDINGUNGEN/ORTE

Ehrenamtlich Krankenhäuser Familienfeste  
Kaufhäuser Kirchen Schulen  
Stadt Andere Zertifiziert

Trier 300 **Neue Suche**

nikolaus1 Stadt: Koblenz  
nikolaus3 Stadt: Trier

Abb. 4.3. Darstellersuche in Koblenz

## 4.7 Filterung

Wird eine Suche gestartet kann weiter gefiltert werden und beispielsweise werden nur Zertifizierte Darsteller angezeigt<sup>4.4</sup>.

Startseite [nikolaus2!](#) oder **Ausloggen!** 27.07.2018

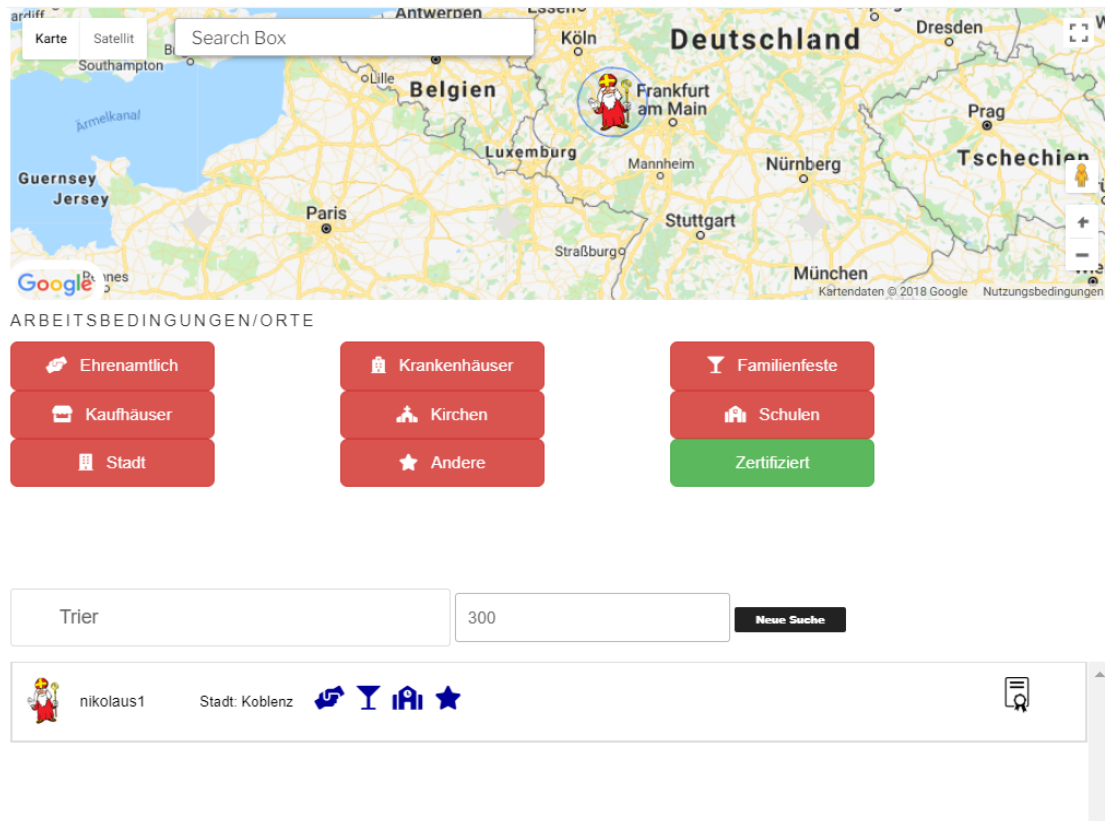


Abb. 4.4. filtern von Darstellern

In der Liste werden Darsteller aufgelistet, welche den Such- und Filterkriterien entsprechen. Durch ein Klicken auf einen Listeneintrag wird das Profil dieses Darstellers in einem neuen Tab geöffnet. Schwebt der Mauscursor über ein Tag, wird eine Beschreibung angezeigt.

## 4.8 Nutzerbereich

Der Nutzerbereich ähnelt dem von WordPress, jedoch wurde er für die Anforderungen erweitert <sup>4.5</sup>.

Im oberen Bereich des Nutzerbereiches kann der Nutzer das Erscheinungsbild anpassen und grundlegende Elemente wie E-Mail und Passwort ändern. Danach kann

er auswählen, ob er sich als Nikolausdarsteller kennzeichnen möchte. Jeder Darsteller akzeptiert, dass er damit ein Teil seiner Daten öffentlich preisgibt.

The screenshot shows the user profile interface for the 'Nikolaus\_Plugin\_Test' application. The left sidebar contains navigation links: 'Medien', 'Profil' (highlighted), and 'Menü einklappen'. The main content area is titled 'Profil' and includes a 'Persönliche Optionen' section. Under 'Farbschema verwalten', there are three color scheme options: 'Standard' (selected), 'Ektoplasma', and 'Mitternacht'. The 'Sprache' is set to 'Website-Einstellung'. The 'Name' section shows the 'Benutzername' as 'rudolph', with a note that usernames cannot be changed. The 'Kontaktinfo' section shows the 'E-Mail (erforderlich)' as 'a@a.de'. The 'Benutzerkonten-Verwaltung' section includes a 'Neues Passwort' button and a 'Sessions' button. The 'Nikolaus werden?' section has a checked checkbox and a warning message. The 'Vorname' and 'Nachname' fields are empty, with placeholder text indicating where to enter the first and last names.

Profil

Persönliche Optionen

Farbschema verwalten

☐ Standard

☐ Ektoplasma

☐ Hell

☒ Mitternacht

Sprache

Website-Einstellung

Name

Benutzername

rudolph

Benutzernamen können nicht geändert werden.

Kontaktinfo

E-Mail (erforderlich)

a@a.de

Benutzerkonten-Verwaltung

Neues Passwort

Passwort generieren

Sessions

Überall sonst abmelden

Du bist nur an diesem Ort angemeldet.

Nikolaus werden?

☒ Nikolaus werden?

Achtung, falls Sie ein Nikolaus werden wollen, stimmen Sie zu, dass ein Teil Ihrer Daten (Stadt) veröffentlicht werden.

Vorname

/

Bitte ihren Vornamen eingeben.

Nachname

/

Bitte ihren Nachnamen eingeben.

Abb. 4.5. Teil 1 Nutzerbereich

Im mittleren Teil des Nutzerbereiches existieren wichtige Elemente für Darsteller 4.6. Auf einer Karte kann der Darsteller bestimmen, an welcher Position er für andere Nutzer angezeigt werden soll. Ebenfalls kann er Tags auswählen, an welchen Orten er Arbeiten möchte und ob er dies Ehrenamtlich tun will. Der Kilometer Radius den er angibt zeigt später auf der Karte seinen Arbeitsradius. Dieser wird in Form eines blauen Kreises realisiert, welcher zentral vom Darsteller ausgeht. Dadurch erhalten andere Nutzer einen groben Eindruck vom Aktionsradius dieses Darstellers.

**Position**

Karte Satellit Search Box

Den Haag Niederlande Antwerpen Brüssel Gullie Belgien Paris Luxemburg Straßburg Zürich Mannheim Stuttgart Nürnberg München Salzburg Österreich

Frankfurt am Main

Bitte ihre Position auf der Karte klicken.

Postleitzahl /  
Bitte ihre Postleitzahl eingeben.

Land /  
Bitte ihr Land eingeben.

Alter /  
Bitte ihr Alter eingeben.

Bio  
Erzählen Sie etwas über sich..

In welcher Umgebung möchten Sie arbeiten?

☐ Ehrenamtlich ☐ Krankenhäuser  
☐ Familien Feste ☐ Kaufhäuser  
☐ Kirche ☐ Schulen  
☐ Stadt ☐ Andere

In wieviel KM Radius wollen sie arbeiten? 0  
Bitte Radius eingeben in Kilometern.

Abb. 4.6. Teil 2 Nutzerbereich

Im unteren Bereich kann ein Zertifikat und ein Bewerbungsfoto hochgeladen werden<sup>4.7</sup>. Falls der Darsteller hier keine Daten hochladen möchte, werden hier vordefinierte Standardbilder bereitgestellt.

Beiträge

Medien

Seiten

Kommentare

Design

Plugins 1

Benutzer

Alle Benutzer

Neu hinzufügen

Dein Profil

Werkzeuge

Einstellungen

Menü einklappen

In welcher Umgebung möchten Sie arbeiten?

☐ Ehrenamtlich

☐ Familien Feste

☐ Kirche

☐ Stadt

☐ Krankenhäuser

☐ Kaufhäuser


☐ Schulen

☐ Andere

In wieviel KM Radius wollen sie arbeiten?

Bitte Radius eingeben in Kilometern.

Nikolaus-Zertifikat



Zertifikat hochladen

Zertifikat löschen

Bitte laden Sie Ihr [Zertifikat des Erzbistum Köln](#) hoch.

Profilbild




Bild hochladen

Profilbild löschen

Bitte laden Sie Ihr Profilbild hoch.

Profil aktualisieren

Abb. 4.7. Teil 3 Nutzerbereich



## 4.9 Nutzerprofile

Auf den Nutzerprofilen gibt es für nicht registrierte Nutzer nicht sehr viel Informationen und Funktionen 4.8. Eingeloggte Nutzer besitzen die Möglichkeit Anfragen an den Nutzer zur Freischaltung dessen Profils zu stellen. Ebenfalls können sie über ein Formular eine Nachricht an diesen Nutzer senden. Die letzte Funktion, welche eingeloggten Nutzern zur Verfügung steht, besteht darin, ein unpassendes oder gar verbotenes Profil zu melden.

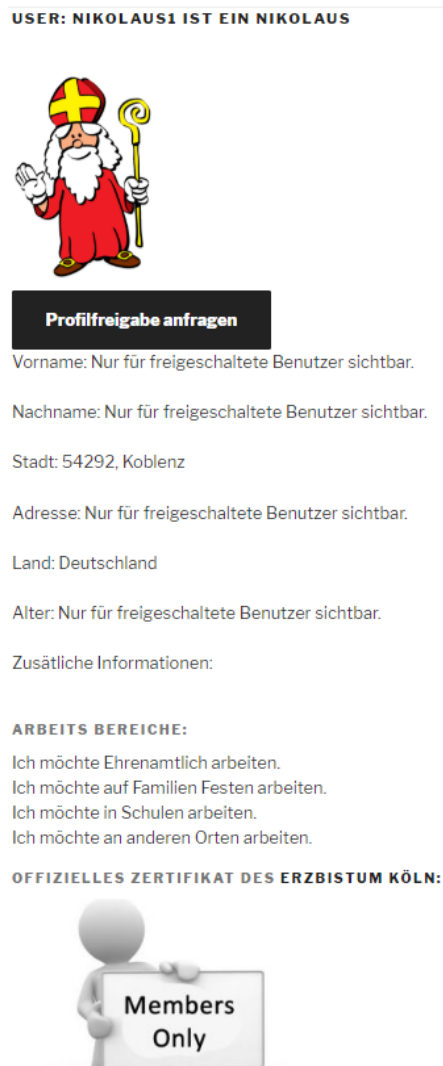


Abb. 4.8. Profil nicht freigeschaltet

### 1. Profilfreigabe Anfragen

Fragt ein Nutzer die Profilfreigabe eines anderen Nutzer an. Nimmt dieser Nutzer die Anfrage auf der Startseite wahr und kann sich das Profil des Anfragenden anschauen 4.9. Anschließend kann er entscheiden ob er es freigibt.

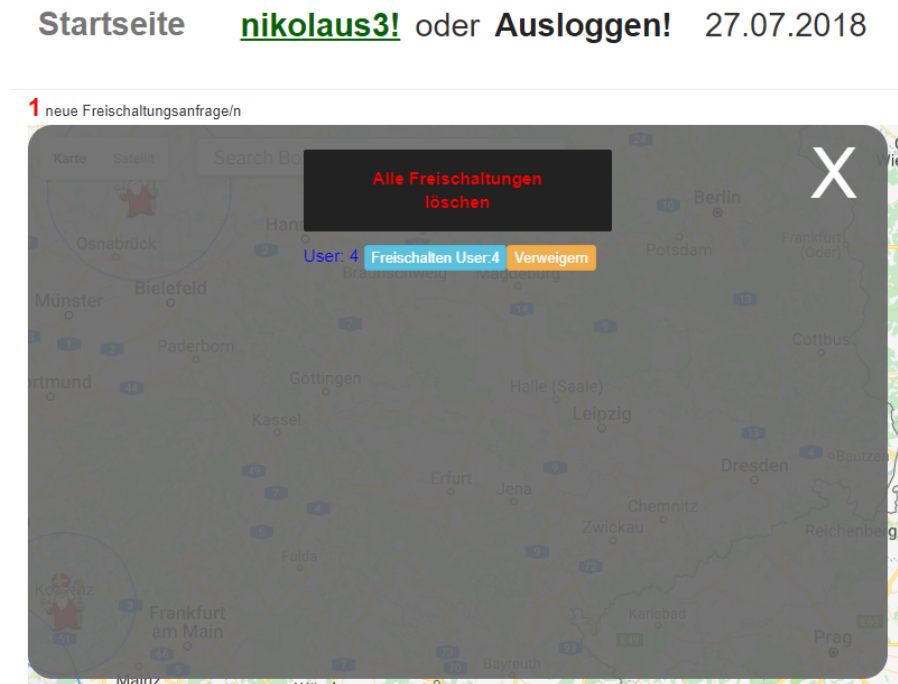


Abb. 4.9. Freischaltanfrage

### 2. Nutzerprofile löschen

Das Löschen von Nutzerprofilen erfolgt automatisch. Nach vier Monaten sendet WordPress eine E-Mail an inaktive Nutzer mit der Bitte sich einzuloggen, andernfalls wird eine Woche darauf das Profil aus der WordPress Datenbank entfernt.

### 3. Kontaktanfrage

Bei der Kontaktaufnahme wird der in der Nachricht eingegebene Text durch die PHP-Mail-Funktion an die E-Mail-Adresse des Nutzers gesendet. Ebenfalls wird die E-Mail-Adresse des Absenders mitgesendet. Jeder Nutzer muss beachten, dass von WordPress versandte E-Mails oft als Spam deklariert werden.

### 4. 'Profil melden' Funktion

Um Spam zu minimieren, können Profile nur von eingeloggten Nutzern gemeldet werden. Falls ein Nutzer ein Profil meldet, wird an alle Administratoren in der WordPress Applikation eine E-Mail-Adresse mit einem Hyperlink zum gemeldeten Profil gesendet.

## Implementierung

Bei der Implementierung muss im Vorhinein gesagt werden, dass hier viele Aspekte von Wordpress übernommen wurden. So zum Beispiel die Registrierung und der Login. Jedoch braucht das Plugin selbst einen Ankerpunkt, womit es auf eine von Wordpress erstellte Seite eingebunden werden kann.

### 5.1 Einbindung

Um die Hauptseite einzubinden, muss zuerst das Plugin bestimmte Kommentare enthalten, damit es in Wordpress unter Plugins angezeigt und aktiviert werden kann siehe Abb. 5.1.

**Listing 5.1.** Anfangskommentar

```
1  /*
2  Plugin Name: Nikolaus Vermittler Map
3  Description: Plugin, welches die Hauptkarte anzeigt mit
4  Shortcode: [nikolaus_plugin_map] auf einer Seite.
5  Version: 1.0
6  Author: Philipp Lippold
7  Author URI: https://lippold-it.de/
8  Min WP Version: 1.5
9  Max WP Version: 5.0.4
10 */
```

Zeile 2 gibt den Namen des Plugins an.

In Zeile 3 befindet sich die Beschreibung, in diesem Fall die Installationsanweisung.

Des Weiteren wird in Zeile 5 die Versionsnummer des Plugins hinterlegt.

In Zeile 6 befindet sich der Name des Erstellers.

Zeile 7 ist der Link zum Autor.

Auf der Webseite in der WordPress Anwendung, an welcher Stelle die Hauptseite erscheinen soll muss der passende Shortcode in diesem Fall **'nikolaus\_plugin\_main'** 5.2 eingefügt werden. An der Stelle dieses **Shortcodes** wird die Methode **'mapload'** aus dem Plugin ausgeführt.

**Listing 5.2.** mapload Funktion einbinden

```
1  add_shortcode( 'nikolaus_plugin_main', 'mapload' );
```

## 5.2 Hauptseite

### 1. Darstellerkarte

Die Darstellerkarte wird per Javascript eingebunden. Die Darsteller auf der Karte werden mithilfe der Informationen aus der Liste, welche am Anfang generiert wird, angezeigt. Es existiert eine Funktion, welche alle Darsteller und dazugehörigen Kreise für die Arbeitsradien löscht und eine Funktion, welche alle Darsteller der Karte hinzufügt. Beim Hinzufügen von Darstellern werden sogenannte Marker auf der Karte erstellt. Diese besitzen als Icon das Profilbild des jeweiligen Darstellers. Die Marker sind gleichzeitig Hyperlinks, welche in einem neuen Tab das Profil des angeklickten Darstellers öffnen. Ein Arbeitsradius wird für jeden Marker ein Arbeitsradius gezeichnet 5.1. Der Radius des Kreises, welcher gezeichnet werden soll, berechnet sich wie folgt:

$$\text{radius\_kreis} = (\text{eingabe\_in\_km} / \text{erddurchmesser\_in\_km}) * \text{erdradius\_in\_meter}$$



Abb. 5.1. Arbeitsradius

### 2. Darstellersuche

Die Darstellersuche ist in zwei Teile unterteilt. Zum einen die Umgebungssuche, welche Darsteller in einem bestimmten Umkreis ausgibt und zum anderen die Filterung durch sogenannte Tags. Tags sind sogenannte Suchwörter. Die Filterung mit Tags kann nur verwendet werden, falls eine Suche bereits gestartet wurde. Dies ist so implementiert da nach der Umgebungssuche die Liste nur noch Darsteller anzeigt, welche sich in einem bestimmten Gebiet aufhalten. Die

Liste kann danach mithilfe der Tags weiter gefiltert werden. Als Beispiel: Wenn zwei Darsteller aus Trier kommen und einer aus Koblenz und im Umkreis Trier gesucht wird, enthält die Liste nur die zwei Darsteller aus Trier. Daraufhin kann auf diese Liste die Filterung mithilfe von Tags angewendet werden. Die Suche funktioniert folgendermaßen:

1. Beim klicken auf den Suchen Button wird ein Ajax Befehl ausgeführt Abb.5.3. Dieser startet ein PHP-Skript, welches die Umgebungssuche startet.

#### Listing 5.3. Ajax Suchanfrage

```
1 function search() {  
2     return $.ajax({  
3         url: "wp-content/plugins/nk_plugin/cityarea.php",  
4         data: {  
5             cityname: $('#myInput').val(),  
6             radius: $('#number').val()  
7         },  
8         type: 'POST',  
9         success: function(response) {  
10             result = response;  
11         }  
12     });  
13 }
```

*Zeile 1* ist Name der Javascript Funktion.

*In Zeile 2* wird die Ajax Anfrage zurückgegeben und ausgeführt.

*Zeile 3* ist die Referenz auf das PHP-Skript für die Ajax Anfrage.

*Zeile 4 bis 7* die Daten, welche an das Skript übergeben werden, in diesem Fall Stadt und Radius.

*Zeile 8* ist die Art der Anfrage. Bei diesem Code wird 'POST' verwendet.

*Zeile 9 bis 11* beschreibt, falls die Funktion erfolgreich ist wird die variable 'result' zurückgegeben. Die Variable 'result' ist ein Array welches Städte als Strings enthält.

2. In dem PHP-Skript der Umgebungssuche wird nun mit dem übergebenen Stadtnamen eine Anfrage an die Google Maps API gestellt. Die API liefert die passenden Längen- und Breitengrade Abb. 5.4.

#### Listing 5.4. Positionsdaten abfrage

```
1 $geocodeObject = json_decode(file_get_contents('https://maps.  
    googleapis.com/maps/api/geocode/json?key=AIzaSyC9wMbx-  
    lXmvZMEmrV0aaFzC-pK4JMhazg&address='.$cityname.',germany'),  
    true);
```

3. Danach werden mit diesen Längen- und Breitengrade sowie dem Radius eine Anfrage an die Geonames API 'findNearbyPlaceName' gestellt 5.5. Anschließend liefert die API alle Städte im Radius der übergebenen Längen- und Breitengrade als Array zurück.

**Listing 5.5.** Anfrage Geonames Api

```
1 $nearbyCities = json_decode(file_get_contents('http://api.  
   geonames.org/findNearbyPlaceNameJSON?lat='.$latitude.'&lng  
   ='>'.$longitude.'&style='.$response.'&cities='.$size.'&radius  
   ='>'.$radius.'&maxRows='.$maxRows.'&username='.$username,  
   true));
```

4. Zuletzt wird geprüft, welche Stadt jeder Darsteller hinterlegt hat. Ist seine Stadt in dem Array enthalten wird der Darsteller auf der Liste angezeigt Abb. 5.6.

**Listing 5.6.** Umgebungssuche

```
1  for (i = 0; i < li.length; i++) {  
2      if(result.includes($(li[i]).attr('city'))){  
3          li[i].style.display = "list-item";  
4          li[i].classList.add("in_city_radius");  
5      }  
6      else  
7      {  
8          li[i].style.display = "none";  
9      }  
10  
11 }
```

*Zeile 1* läuft alle Elemente der Liste durch.

Die Variable 'result' ist ein Array. Dieses wird durch die Ergebnisse der Umgebungssuche gefüllt.

*Zeile 2 bis 9* prüft, ob die Stadt eines Darstellers im Array 'result' enthalten ist. Ist dies der Fall wird der Darsteller in der Liste angezeigt und enthält als extra Klassenattribut 'in\_city\_radius'. Ist seine Stadt nicht enthalten wird er ausgeblendet, da er dem Suchschema nicht entspricht.

Außerdem wird beim Suchen auf die Darstellerkarte gescrollt und je nachdem, wie hoch der eingegebene Suchradius ist, hineingezoomt. Bei der Filterung werden alle Filtertags, welche ausgewählt wurden in einem Array gespeichert. Zum Schluss wird verglichen, welcher Darsteller in der Liste dieses Tag besitzt. Wenn ein Darsteller dieses Tag besitzt, wird er in der Liste angezeigt Abb. 5.7.

**Listing 5.7.** Filterung durch Tags

```
1  function Filtertags(){
2      var li= document.getElementsByClassName('in_city_radius
      ');
3      for (j = 0; j < li.length; j++) {
4          var tag_include=false;
5          var classListe = li[j].classList;
6          for(var k=0;k<tags.length;k++){
7              if(classListe.length<tags.length){
8                  tag_include=false;
9                  break;
10             }
11             if(!(classListe[k]=='in_city_radius')){
12                 if(classListe.contains(tags[k])){
13                     tag_include=true;
14                 }
15             }
16             else{
17                 tag_includ=false;
18                 break;
19             }
20         }
21         if(tag_includ){
22             li[j].style.display="block";
23         }
24         else{
25             li[j].style.display="none";
26         }
27         if(tags.length==0){
28             li[j].style.display="block";
29         }
30     }
31     showUserOnMap();
32 }
```

In *Zeile 2* werden alle Darsteller, welche zuvor durch die Umgebungssuche gefiltert worden in eine Liste gespeichert.

In *Zeile 3* iteriert eine Schleife über die Darsteller in der Liste.

*Zeile 5* ist eine Liste mit den Klassenattributen des Darstellers über welchen derzeit iteriert wird.

*Zeile 6* iteriert mit einer Schleife über alle ausgewählten Tags.

*Zeile 7 bis 9* prüft nun, ob die Klassenattribute, des Darstellers kleiner sind als die Anzahl der ausgewählten Tags.

Durch *Zeile 11* wird das Klassenattribut 'in\_city\_radius' ignoriert, da die Funktion dadurch immer alle Darsteller raus filtern würde.



*Zeile 12 bis 18* prüft, ob in den Klassenattributen des Darstellers alle Tags enthalten sind. Ist dies der Fall wird der Darsteller angezeigt, ansonsten aussortiert.

*Zeile 27 bis 29* beschreibt den wichtigen Fall, falls keine Tags mehr ausgewählt werden. Es müssen daraufhin alle Darsteller wieder sichtbar sein.

*Zeile 31* ruft die Funktion auf, welche Darsteller der Liste auf die Karte darstellt.

### 3. Liste der gefilterten Darsteller

Die Liste der Darsteller wird mit Hilfe der Daten aus der Datenbank erstellt  
Abb. 5.8.

**Listing 5.8.** Datenbank funktionen

```
1 get_the_author_meta( 'tag1', $user_id->ID )
2 get_user_meta($user_id->ID, 'latitude', true )
3 get_author_posts_url($user_id->ID)
```

Die Liste wird nur einmalig erstellt beim Laden der Seite. Im Folgenden wird ein Teil der Funktion erläutert Abb.5.9.

**Listing 5.9.** Funktionsausschnitt

```
1  var ul = document.getElementById("myUL");
2      js_user_data = <?php echo json_encode(getUser_Data_search
      ());
3      ?>;
4      for(var i=0;i<js_user_data.length;i++){
5          var elem1 = document.createElement("li");
6          var elem2 = document.createElement("a");
7          var elem3 = document.createElement("img");
8          var elem4 = document.createElement("span");
9          var elem5 = document.createElement("img");
10         var elem6 = document.createElement("img");
11         var elem7 = document.createElement("span");
12         var elem8 = document.createElement("span");
13         var city= js_user_data[i][16];
14         var size="fa-2x";
15         for(var o=6;o<18;o++){
16             switch(js_user_data[i][o]) {
17                 case 'YES':
18                     if(o==6){
19                         elem1.classList.add("ehrenamtlich");
20                         var tagssymbol=document.createElement("i");
21                         tagssymbol.classList.add("fas");
22                         tagssymbol.classList.add(size);
23                         $(tagssymbol).attr("data-toggle", "tooltip");
24                         tagssymbol.title="Dieser Nutzer arbeitet
                                Ehrenamtlich";
25                         tagssymbol.classList.add("fa-hands-helping");
26                         elem8.append(tagssymbol);
27
28
29                     }
```

*Zeile 1* ist die ungeordnete Liste, welche mit Darstellern als Listenelemente befüllt werden.

In *Zeile 2* sind die Darstellerdaten in einer Variable gespeichert, welche vom Server per Funktion im JSon-Format gesendet werden.

In *Zeile 4 bis 12* werden Elemente dynamisch erstellt, welche in die Liste eingefügt werden. Hierbei ist 'i' die Laufvariable für die einzelnen Darsteller, welche vorhanden sind.

*Zeile 13* speichert als Variable die Stadt die der Darsteller hinterlegt hat.

*Zeile 15* an den Positionen 6 bis 17 stehen im 'js\_user\_data' Array immer nur 'YES' oder 'NO' dies sind die Werte der einzelnen Tags jedes Darsteller. Ab *Zeile 17* prüft wenn bei dein Darstellerdaten ein 'YES' hinterlegt ist, welches Tag dies ist. Aus dem Aufbau der 'js\_user\_data' wird dies ersichtlich. Dementsprechend bekommt jeder Listeneintrag das passende Klassenattribut hinzugefügt. Daneben werden noch visuelle Veränderungen wie in *Zeile 25* beispielsweise ein Handsymbol hinzugefügt.

Diese erstellten Elemente erhalten jetzt durch die 'js\_user\_data' die richtigen Informationen. Als Beispiel wird folgenden Code dem Profilbild die URL zugewiesen:

```
1 elem3.src=js_user_data[i][2];
```

#### 4. Freischaltungen

Die Freischaltungen werden realisiert, indem jeder Nutzer zwei Listen in der 'Usermeta-Tabelle' speichert. Zum Einen die Freischaltanfragen und zum Anderen die Freischaltungen. Erhält ein Nutzer eine Anfrage wird in dessen Liste "Freischaltanfragen" die Nutzer-ID des Anfragenden gespeichert. Nimmt der Nutzer die Anfrage an, wird in die Liste der "Freischaltungen" des Anfragenden die Nutzer-ID des Darstellers geschrieben. Gleichzeitig wird die Anfrage aus der Liste gelöscht, da die Anfrage abgearbeitet wurde. Lehnt der Nutzer diese jedoch ab, wird die Anfrage aus der Liste ohne weitere Aktionen gelöscht. Das Annehmen oder Ablehnen wird durch eine Ajax Anfrage gelöst 5.10, damit die Seite nicht neu geladen werden muss.

##### Listing 5.10. Freischaltung von Nutzern

```
1 function unlock_user(user_id,target_id){
2   if(confirm("Moechten Sie Ihr Profil freigaben fuer User:"+
3     target_id+"?")){
4     jQuery.ajax({
5       type: "POST",
6       url: 'wp-content/plugins/nk_plugin/unlock_user.php',
7       data: {id_target: target_id,user_id: user_id},
8       success: function(data){
9     },
10    error: function(XMLHttpRequest, textStatus, errorThrown) {
11      alert("User wurde nicht Freigegeben Status: " + textStatus);
12      alert("Error: " + errorThrown);
13    }});}}
```

In *Zeile 2* wird die Bestätigung des Users verlangt, da sonst keine Anfrage erfolgt.

Ab *Zeile 3* startet die Anfrage.

*Zeile 4* ist der Typ der Anfrage.

*Zeile 5* gibt das PHP Skript an, welches ausgeführt werden soll.

*Zeile 6* die Daten, welche übergeben werden.

*Zeile 7* ist die Funktion die ausgeführt wird, falls die Anfrage erfolgreich ist.  
*Zeile 9 bis 12* Ist die Funktion, welche bei einem Fehlschlag ausgeführt wird.  
In diesem Fall wird der Nutzer informiert, dass die Anfrage nicht geklappt hat.

## 5.3 Nutzerbereich

### 1. Anker

Es wird der von Wordpress standardmäßige Nutzerbereich verwendet. Dieser wird jedoch durch Anker ('Hooks') erweitert. Einer der wichtigsten Anker ist derjenige, welcher das Formular um weiteren HTML Code erweitert 5.11. Ebenfalls wird Javascript Code ausgeführt, damit Beispielsweise die URLs für die Bilder festgelegt werden.

**Listing 5.11.** Anker für Nutzerbereich

```

1  add_action( 'show_user_profile', 'new_user_fields' );
2  add_action( 'edit_user_profile', 'new_user_fields' );

```

Die Werte der Eingabe Felder werden immer aus der Datenbank ausgelesen, damit kann der Nutzer einfach seine alten Daten editieren. In der Abbildung 5.12 erhält das Input-Feld 'firstname' als Wert die Methode **get\_the\_author\_meta(Metakey, Nutzer-Id)**, welche den Metakey aus der Datenbank abrufen.

**Listing 5.12.** Inputfeld mit Funktion als Wert

```

1  <input type="text" name="firstname" id="firstname"
2  value="<?php echo esc_attr(
3  get_the_author_meta( 'firstname', $user->ID ) ); ?>" class="regular-text" />

```

Gibt der Nutzer keinen Wert an, wird trotzdem ein Standardwert eingefügt. Beispielsweise '/' beim Vornamen. In der Abbildung 5.13 wird in Zeile 1 geprüft ob der hinterlegte Wert beim Absenden des Formulars leer ist. Ist dies der Fall, wird in Zeile 2 für diesen Wert in der Datenbank '/' gespeichert.

**Listing 5.13.** Prüfen auf leeren Wert

```

1  if(empty($_POST['firstname'])){
2      update_user_meta( $user_id, 'firstname', '/' );
3  }
4  else{
5      update_user_meta( $user_id, 'firstname', $_POST['firstname'] );
6  }

```

**2. Als Darsteller agieren**

Die Auswahlboxen für die Abfrage, ob jemand als Darsteller agieren möchte wurde ein fester Wert, in diesem Fall 'YES' zugewiesen. Die versteckten Eingabefelder liefern, falls die Auswahlboxen nicht ausgewählt wurde den Wert 'NO' zurück Abb. 5.14. Die aktuellen Zustände der Auswahlboxen werden ebenfalls gespeichert. In Abbildung 5.15 wird in Zeile 1 der Wert einer Auswahlbox in der Datenbank geprüft. Falls dieser Wert 'YES' war, wird diese Auswahlbox wieder ausgewählt. Das versteckte Eingabefeld ist deshalb notwendig, da Auswahlboxen keine zwei Werte besitzen können.

**Listing 5.14.** Auswahlboxen und verstecktes Eingabefeld

```

1  <input name="tag1" value="NO" type="hidden">
2  <input type="checkbox" id="tag1" name="tag1" value="YES" />
3  <label class="tagcheckbox" for="nikolauswerden">Ehrenamtlich</label></>

```

**Listing 5.15.** Speichern des Zustandes

```
1  if("<?php echo esc_attr(get_the_author_meta( 'tag8 ', $user->ID )); ?>"=="YES")
2      {
3
4          $('#tag8 ').prop('checked', true);
5      }
6      else{
7
8          $('#tag8 ').prop('checked', false);
9      }
```

**3. Position bestimmen als Darsteller**

Damit der Darsteller seine Position festlegen kann, wurde hier eine weitere Google Map eingebunden. Beim Klicken auf die Karte wird ein Marker erstellt Abb. 5.16. Von diesem Marker werden beim Absenden des Formulars die Längen- und Breitengrade gespeichert. Später erscheinen dadurch die Darsteller auf der Karte für alle anderen Nutzer an ihren gewählten Positionen.

**Listing 5.16.** Position des Darstellers festlegen

```
1  function placeMarker(location) {
2      if(marker!=null){
3          marker.setMap(null);
4          marker=null;
5      }
6      marker = new google.maps.Marker({
7          position: location,
8          map: map,
9          animation: google.maps.Animation.DROP
10     });
11     $('#latmarker').val(marker.getPosition().lat());
12     $('#lngmarker').val(marker.getPosition().lng());
13 }
```

**4. Bilder hochladen**

Die beiden Bilder, das Zertifikat und das Profilbild werden mit derselben Mechanik eingefügt 5.17. Es wird ein Wordpress Medienfenster geöffnet zum Hochladen. Danach wird überprüft, ob es ein Bild ist und ob die Größe kleiner als vier Megabyte ist. Vom Bild wird die URL, wo es sich auf dem Server befindet in einem für den Nutzer nicht sichtbaren Eingabefeld gespeichert. Beim abspeichern des Formulars, wird diese URL in der Datenbank hinterlegt.

**Listing 5.17.** Bilder hochladen

```
1  $( '.additional-certificate-image' ).on( 'click ', function( event ){
2      event.preventDefault();
3      if ( file_frame2 ) {
4          file_frame2.open();
5          return;
6      }
7      file_frame2 = wp.media.frames.file_frame = wp.media({
8          title: $( this ).data( 'uploader_title' ),
9          button: {
10             text: $( this ).data( 'uploader_button_text' ),
11         },
12         multiple: false
13     });
14     file_frame2.on( 'select ', function() {
15         attachment = file_frame2.state().get( 'selection' ).first().toJSON();
16         if ( !( attachment.mime == "image/jpeg" ||
17             attachment.mime == "image/gif" ||
18             attachment.mime == "image/png" ) &&
19             attachment.filesizeInBytes < 4000000 )
20         {
21             return;
22         }
23         $( zertifikat2 ).val( attachment.url );
24     });
25     file_frame2.open();
26 });
```

*Zeile 1* startet die Funktion beim Buttonklick.

In *Zeile 2* soll das standardmäßige Event unterbunden werden, da sonst das Formular abgeschickt wird.

*Zeile 3 bis 6* prüft ob es schon ein 'file\_frame' Objekt gibt und öffnet dieses. In *Zeile 7 bis 13* wird ein 'file.frame' Objekt erstellt. Dies ist ein von Wordpress modales Medienfenster. Dabei kann beispielsweise festgelegt werden, ob man mehrere Dateien hochladen kann *Zeile 12* und den Titel in *Zeile 8* der Datei.

*Zeile 14* beschreibt was passieren soll, wenn eine Datei ausgewählt wurde.

In *Zeile 15* wird die ausgewählte Datei in das 'JSON' Format umgewandelt und in eine Variable gespeichert.

*Zeile 16 bis 22* prüft ob die hochgeladene Datei kein Bild und größer als 4 Megabytes ist, ansonsten wird die Funktion abgebrochen.

Ab *Zeile 23* wird die URL zum Bild in einem Eingabefeld gespeichert. Ebenfalls wird in *Zeile 25* das 'File\_Frame' geöffnet, sofern keines existierte.

### 5. 'Fallback' Lösung

Lädt der Nutzer kein Bild hoch, greift eine 'Fallback' Lösung ein. Dafür muss geprüft werden ob die entsprechenden Variablen leer sind, mithilfe der Empty-Funktion. In der Abbildung 5.18 prüft die Funktion 'Empty' in Zeile 1, ob die Variable 'Zertifikat' leer ist. Ist dies der Fall wird in Zeile 2 in der Datenbank ein Standardbild festgelegt.

**Listing 5.18.** Bilder Fallback Lösung

```
1  if(empty($_POST['zertifikat'])){
2      update_user_meta( $user_id , 'zertifikat' , $url2 );
3  }
4  else{
5      update_user_meta( $user_id , 'zertifikat' , $_POST['zertifikat'] );
6  }
```



## 5.4 Nutzerprofile

### 1. Geschützte Daten

Die persönlichen Daten jedes Nutzers sind geschützt. Dies wird mit einer einfachen Abfrage gelöst, welche die richtigen Daten ausgibt, sofern der Nutzer freigeschaltet ist oder wenn es sich um sein eigenes Profil handelt siehe Abb. 5.19.

**Listing 5.19.** private Daten verstecken

```

1 <p>Vorname: <?php
2     if (in_array($user_id,$unlock_array)
3         || $current_user->ID==$user_id){
4         echo get_the_author_meta( 'firstname', $user_id ) ;
5     } else{
6         echo 'Nur fuer freigeschaltete Benutzer sichtbar.';
7     }
8     ?></p>
```

*Zeile 1* die Bezeichnung des Feldes in HTML-Code.

*Zeile 2* fragt ab ob sich die 'Nutzer-Id' des besuchten Profils in der Liste der Freischaltungen des Besuchers befindet. Ist dies der Fall oder der Besucher ist der eigentliche Nutzer, werden Serverseitig die Daten dieses Nutzers ausgegeben.

In *Zeile 5* wird andernfalls ein Standardstring ausgegeben.

### 2. Profil melden

Die Funktion Profile zu melden wird mit einer Ajax Anfrage gelöst siehe Abb. 5.20.

**Listing 5.20.** Profil melden Funktion

```

1 function report(){
2 <?php
3     if (isset($_GET['author'])) {
4         $user_id = $_GET['author'];
5     }
6     $url = "://{$_SERVER['HTTP_HOST']}{$_SERVER['REQUEST_URI']}";
7     ?>
8     var user_id=<?php echo $user_id; ?>;
9     var url='<?php echo $url; ?>';
10    if(confirm("Moechten Sie den Nutzer wirklich melden?")){
11        jQuery.ajax({
12            type: "POST",
13            url: 'wp-content/plugins/nk-plugin/report.php',
14            data: {url: url,user_id:user_id},
15            success: function(data){
16                alert("Danke sehr der Nutzer wurde gemeldet.");
17            },
18            error: function(XMLHttpRequest, textStatus, errorThrown) {
19                alert("User wurde nicht gemeldet Status: " + textStatus); alert("Error: " +
20            }
21        });
22    }
23 }
```

*Zeile 4* speichert die 'Nutzer-Id' des geöffneten Profils und *Zeile 6* die URL zum Profil.

Nun wird wieder eine Ajax Anfrage gestartet und als Daten werden die URL und die Nutzer-Id des Profils übergeben.

Daraufhin wird das PHP-Skript gestartet, welches am Ende an alle Administratoren Emails verschickt siehe Abb. 5.21.

**Listing 5.21.** PHP-Skript Profil melden

```

1  function get_administrator_email(){
2      if(isset($_POST['user_id'])){
3          {
4              $user_id = $_POST['user_id'];
5          }
6          if(isset($_POST['url'])){
7              {
8                  $url = $_POST['url'];
9              }
10         $blogusers = get_users('role=Administrator');
11         foreach ($blogusers as $user) {
12             $to = $user->user_email;
13             $subject = 'Profil Meldung Bitte ueberpruefen';
14             $escaped_url = htmlspecialchars( $url, ENT_QUOTES, 'UTF-8' );
15             $link = "<a href=" . $escaped_url . ">Jetzt ueberpruefen</a>";
16             $message = 'Bitte das Profil von Usernr: ' . $user_id . ' ' . $link . ' ';
17             $headers = array('Content-Type: text/html; charset=UTF-8');
18             wp_mail( $to, $subject, $message, $headers );
19         }
20     }

```

In *Zeile 2 bis 9* werden die übergebenen Daten als Variablen abgespeichert. *Zeile 10* speichert als Variable alle Administratoren ab und iteriert in *Zeile 11* über diese.

In *Zeile 12* wird die E-Mail-Adresse, des jeweiligen Administrators in einer Variablen hinterlegt.

*Zeile 13* ist der Betreff der Email.

*Zeile 14* die URL zum Profil des gemeldeten Nutzers. die Funktion 'htmlspecialchars' wandelt hierbei alle vordefinierten Zeichen in HTML Zeichen um. Beispielsweise ist '&' das Undzeichen (&) in HTML-Code.

*Zeile 15* erstellt einen Link und gibt als Ziel die URL des Profils an.

*Zeile 16* ist die Nachricht, welche auch den Link zum Profil aus *Zeile 15* enthält.

In *Zeile 17* wird ein Array mit Header-Daten definiert.

In der letzten Zeile, *Zeile 18* wird die E-Mail mit Empfänger, Betreff und Nachricht versendet. Ebenfalls liegen dieser Email Header-Daten für HTML bei.

## 5.5 Automatische Löschung von inaktiven Nutzern

Wenn Nutzer vier Monate und eine Woche nicht mehr auf der Plattform gewesen sind, werden diese aus dem System gelöscht. Bevor der Löschvorgang jedoch startet, werden diese Nutzer auf dieses Ereignis durch eine E-Mail hingewiesen, damit dies Ereignis verhindert werden kann. Dieses Feature wurde wie folgt implementiert:

1. Gibt es eine Funktion die einen Zeitplan zurückliefert siehe Abb. 5.22.

**Listing 5.22.** Funktion für Zeitplan

```

1 function schedule( $schedules ) {
2     $schedules['every_hour'] = array(
3         'interval' => 1 * HOUR_IN_SECONDS,
4         'display'  => __( 'Jede Stunde' ),
5     );
6     return $schedules;
7 }

```

*Zeile 2* ist ein Array, welches an der Stelle mit dem Schlüsselwert 'every\_hour' folgende Elemente in einem Array enthält:

1. Einer Variablen 'Intervall', mit einem Wert, wann dieser immer aus geführt werden soll.
2. Die Variable 'Display', welche die Beschreibung angibt.
3. Folgen die Ankerfunktionen, damit WordPress weiß, welche Funktionen regelmäßig ausgeführt werden sollen siehe Abb. 5.23.

**Listing 5.23.** Anker Funktionen

```

1 if ( ! wp_next_scheduled( 'cron_hook' ) ) {
2     wp_schedule_event( time(), 'every_hour', 'cron_hook' );
3 }
4 add_action( 'cron_hook', 'cron_function' );
5 add_filter( 'cron_schedules', 'schedule' );

```

*Zeile 1* fragt ab ob die Funktion 'cron\_hook' bereits geplant ist zum ausführen.

- Falls nicht wird in *Zeile 2* ein Schedule-Event angelegt. Dabei erhält dieses mindestens 3 Parameter:
  1. Parameter sagt aus, wann das Event ausgeführt wird im 'UNIX-Timestamp' Format.
  2. Wie oft soll das Event wieder ausgeführt werden. In diesem Fall stündlich.
  3. Die Funktion, welche ausgeführt werden soll. In diesem Fall die 'cron\_hook' Funktion. In *Zeile 4* wird festgelegt, welche Aktion ausgeführt werden soll, wenn die Funktion 'cron\_hook' ausgeführt wird. In diesem Fall wird die Funktion 'cron\_function' ausgeführt. Dies passiert durch den Zeitplan alle sechs Stunden. In *Zeile 6* werden an den WordPress Zeitplänen 'cron\_schedules' unser Zeitplan 'schedule' angehängt. Die WordPress Zeitpläne erwarten ein Array voller Arrays. Wobei das äußerste Array immer ein Schlüsselwort enthält, als Beispiel 'weekly' für wöchentlich [Wor].

Die folgende Funktion wird alle sechs Stunden aufgerufen durch eine Zeitplan, welcher zuvor angelegt wurde. Diese Funktion ermöglicht es inaktive Nutzer nun per Email zu benachrichtigen und zu löschen siehe Abb. 5.24.

**Listing 5.24.** Cron Funktion

```

1 function cron_function() {
2     $users = get_users( array( 'fields' => array( 'ID' ) ) );
3     foreach($users as $user_id){
4         $delta = time() - get_the_author_meta( "last_login", $user_id->ID );
5         $delta2 = 11104800 - $delta ;
6         if (!(isSiteAdmin($user_id->ID))){
7             $mail=get_the_author_meta( "mail", $user_id->ID );
8             if ($delta2<10500000&&$mail=="no"){
9                 update_user_meta( $user_id->ID, 'mail', 'yes' );
10                $user = get_user_by('id',$user_id->ID);
11                $to=$user->user_email;
12                $subject='Sind Sie noch Nikolaus? :)';
13                $message = "Hallo sind Sie noch bei der
14                Nikolausvermittlung dabei?
15                Falls Ja loggen Sie sich bitte auf Ihrem Nutzerkonto ein ,
16                da dieses sonst Automatisch geloescht wird.";
17                wp_mail( $to, $subject, $message );
18            }
19            else if ($delta2<=0){
20                wp_delete_user( $user_id->ID);
21                delete_user_meta( $user_id->ID);
22            }
23        }
24    }
25 }

```

In *Zeile 2* werden alle Nutzer in eine Variable gespeichert und in *Zeile 3* wird über diese iteriert.

In *Zeile 4* wird eine Variable 'Delta' gespeichert diese ergibt sich aus der aktuellen Serverzeit, wenn diese Funktion ausgeführt wird minus der Serverzeit, als der Nutzer zuletzt online war. Also steigt Delta immer weiter an.

In *Zeile 5* gibt es eine Variable 'Delta 2' welche sich aus der Zahl 11104800 ergibt, welche als Zeitabdruck vier Monate und eine Woche ergibt minus des zuvor kalkulierten Wertes 'Delta'.

*Zeile 6* prüft nun ob es sich um einen Administrator handelt. Ist dies der Fall, wird die Funktion nicht weitergeführt.

*Zeile 7* speichert die Variable Mail ab, damit wird festgelegt, ob der Nutzer bereits eine Email Benachrichtigung erhalten hat.

*Zeile 8* prüft nun, ob 'Delta 2' kleiner ist als 10500000, was in diesem Fall vier Monate entspricht und ob er eine E-Mail-Benachrichtigung erhalten hat.

*Zeile 9* vermerkt nun, dass der Nutzer eine Email erhalten hat, beziehungsweise wird.

In *Zeile 10 bis 16* werden E-Mail-Adresse, sowie Betreff und Nachricht gespeichert.

In *Zeile 17* wird eine E-Mail abgeschickt.

*Zeile 19* prüft, ob 'Delta 2' kleiner als 0 und der Nutzer, somit länger als vier Monate und einer Woche inaktiv war.

Ist dies der Fall, wird er in *Zeile 17 und 18* aus dem System gelöscht.

## Sicherheitskonzept

Das Sicherheitskonzept stammt von der OWASP dem sogenannten Open Web Application Security Project. Die Wordpress Security Implementation Guideline beschreibt Möglichkeiten, welche erheblich zur Sicherheit der WordPress Anwendung beitragen und werden kostenlos zur Verfügung gestellt [Pro]. Zu erwähnen wäre hier, dass sie die meisten Punkte auf die WordPress Anwendung im Allgemeinen beziehen und weniger auf die Plugin Entwicklung. Die wichtigsten Richtlinien sind die Folgenden:

1. Plugins und Themes müssen unbedingt automatisch aktualisiert werden. Dafür gibt es zwei Filterfunktionen, welche in die 'wp-config.php' Datei eingefügt werden siehe Abb. 6.1.[Pro]

**Listing 6.1.** Funktionen für automatische Aktualisierungen

```
1 add_filter( 'auto_update_plugin', '__return_true' );
2
3 add_filter( 'auto_update_theme', '__return_true' );
```

2. Ebenfalls sollten ungenutzte Plugins, sowie Themes entfernt werden. Sollten Sicherheitslücken in ungenutzten Plugins oder Themes entstehen, können diese trotzdem ausgenutzt werden. [Pro]
3. Den Kerncode von WordPress aktualisiert zu halten, verhindert viele Sicherheitslücken. Dafür muss in der 'wp-config.php' Datei die Standardeinstellung angepasst werden siehe Abb. 6.2.[Pro]

**Listing 6.2.** WordPress Kern automatisch Updaten

```
1 define( 'WP_AUTO_UPDATE_CORE', true );
```

4. Es gibt Dateien welche entfernt werden sollten. Als Beispiel die 'license.txt' Datei. Diese gibt Angreifern die Möglichkeit herauszufinden, wann WordPress zuletzt aktualisiert wurde. Dadurch ist es möglich 'Exploits', also sogenannte Schwachstellen des Systems auszunutzen.[Pro]
5. Datenbankprefixe sollten während der Installation umgeändert werden. Beispielsweise sollte das standardmäßige Präfix 'wp\_' in 'vesluaq3\_' oder einem anderen zufälligen Präfix umgeändert werden. Dadurch werden SQL-Injection-Angriffe eher verhindert, welche das Ziel haben Daten zu stehlen oder die Webseite zu hacken.[Pro]

6. Der direkte Zugriff auf Dateien sollte verhindert werden. Dafür muss am Anfang des eingebundenen Plugins eine Funktion eingebunden werden siehe Abb. 6.3.[Pro]

**Listing 6.3.** Nur Absolutpfade

```
1 defined( 'ABSPATH' ) or die( 'No script kiddies please!' );
```

7. Ein Captcha sollte eingebunden werden, damit Brute-Force-Attacken verhindert werden. Empfehlenswert ist hierfür zum Beispiel das reCAPTCHA-Plugin. Dadurch wird bei der Registrierung und beim Login ein Captcha abgefragt. Ebenfalls werden automatische Registrierungen durch Bots unterbunden.[Pro]
8. Ein Sicherheitsplugin wie beispielsweise WordFence Security Plugin sollte installiert werden. Dadurch wird der Administrator ebenfalls informiert, wenn besondere Ereignisse auftreten, wie Hackingangriffe.[Pro]

## Zusammenfassung und Ausblick

Das Ziel war es gewesen für Personen, welche als Nikolausdarsteller auftreten wollen eine Plattform zur Verwaltung anzubieten. Die Darsteller, ob hobbymäßig oder zertifiziert, vom Bund der Deutschen Katholischen Jugend (BDKJ) haben die Möglichkeit ihren Standort auf einer Karte darzustellen und ein Profil mit persönlichen Informationen anzulegen. Nutzer welche registriert sind besitzen hingegen die Optionen in Kontakt mit anderen Nutzern zu treten, Profilfreigaben anzufragen oder Profile zu melden. Das Plugin selbst ist wartungsfrei und benötigt wenig Administration. Ebenfalls wurden Sicherheitsaspekte nach OWASP berücksichtigt. Die Anwendung ist als fertiges Produkt zu sehen, kann jedoch um weitere Features erweitert werden. Als Beispiel könnte in der Zukunft ein Newsletter-System oder ein Chat-System integriert werden. Die Filterfunktion könnte um weitere Optionen erweitert werden. Das Plugin bietet eine solide Kernstruktur und besitzt Potential umstrukturiert zu werden, um andere Anwendungsfälle abzudecken.

---

## Literaturverzeichnis

- aH. HAMBURG, DOCKMEDIA CONTAO AGENTUR AUS: *Was ist jQuery*. jQuery.
- Fal. FALTINGS, THORSTEN: *Was ist Wordpress*. Was ist Wordpress.
- geo. GEONAMES: *GeoNames Webservices*. GeoGames.
- Gro. GROUP, THE PHP: *Was ist PHP*. Was ist PHP.
- Kö. KÖLN, NIKOLAUSAKTION – BDKJ STADTVERBAND: *Den Nikolaus kennenlernen*. Nikolaus Aktion.
- LLC. LLC, GOOGLE: *Google Maps Dokumentation*. Google Maps.
- Mol. MOLILY: *JavaScript: Serverkommunikation und dynamische Webanwendungen (Ajax)*. Ajax.
- Moz. MOZILLA: *Was ist PHP*. DOM.
- Pro. PROJECT, THE OPEN WEB APPLICATION SECURITY: *Wordpress Security Implementation Guideline*. [https://www.owasp.org/index.php/OWASP\\_Wordpress\\_Security\\_Implementation\\_Guideline](https://www.owasp.org/index.php/OWASP_Wordpress_Security_Implementation_Guideline).
- Tut. TUTANCH: *Was ist ein Brute-Force Angriff*. Brute-Force.
- Wor. WORDPRESS: *WordPress Documentation*. WordPress Docu.



**A**

---

## **Erklärung der Kandidatin / des Kandidaten**

☐ Die Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen- und Hilfsmittel verwendet.

☐ Die Arbeit wurde als Gruppenarbeit angefertigt. Meine eigene Leistung ist ...

Diesen Teil habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Namen der Mitverfasser: ...

---

Datum

---

Unterschrift der Kandidatin / des Kandidaten