Bachelorarbeit

Konzeption und Implementierung einer webbasierten Applikation zur Darstellung und Verwaltung von Veranstaltungen

zur Erlangung des akademischen Grades eines

Bachelor of Science

erstellt von

Marius Kai Preikschat, geb. in Oberhausen

im Studiengang

Informatik.Softwaresysteme

des Fachbereiches

Wirtschaft und Informationstechnik

bei Erstprüfer

Prof. Dr. Martin Schulten

und Zweitprüferin

Carolina Wehrmann

Abgabedatum:

20. August 2019

Westfälische Hochschule

University of Applied Sciences
Gelsenkirchen Bocholt Recklinghausen

Inhaltsverzeichnis

Abkürzungsverzeichnis

PHP Hypertext Preprocessor, urspünglich Personal Home Page[1]

HTML Hypertext Markup Language[2]

CSS Cascading Style Sheets[3]

SQL Structured Query Language[4]

GUI Graphical User Interface

SFTP Secure File Transfer Protocol[5]

[1] [Webtechnologien, S.54]
[2] [HTML5 und CSS3, S.2]
[3] [HTML5 und CSS3, S.44]
[4] [Datenbanken und SQL, S.99]
[5] [SFTP]

Abbildungsverzeichnis

1 Einführung

Im Rahmen dieser Bachelorarbeit wird eine webbasierte Applikation zur Darstellung und Verwaltung von Veranstaltungen konzipiert und entwickelt. Den Ausgangspunkt dieser Applikation bildete die Faehre, welche eine Einrichtung für Menschen ist, die sich in einer psychischen Krise befinden. Alle weiteren Informationen zur Faehre und dessen Anforderungen an diese Applikation sind in den nachfolgenden Unterkapiteln erörtert.
Die Grundanforderung an diese Applikation ist, dass die Bewohner des Wohnheims der Faehre sich Informationen zu einmaligen, sowie mehrmaligen Veranstaltungen anzeigen lassen können. Solche Veranstaltungen sind beispielsweise ein gemeinsamer Spieleabend oder Spaziergänge.
In diesem Kapitel ist die Faehre, mit der Idee einer eigenen Social Media Plattform, und die Motivation zur Gestaltung dieser Applikation beschrieben.

1.1 Die Faehre - Das Faehrbook

Die Faehre[6] bietet ein ambulantes Betreuungsangebot für Menschen, die sich in einer psychischen Krise befinden. Die Betreuungsangebote können sowohl in der eigenen Wohnung, als auch in bereitgestellten Wohnunterkünften wahrgenommen werden. Die Wohnunterkünfte sind alle betreut und geben die Möglichkeit entweder in einem Wohnhaus unterzukommen, in dem Einzelwohneinheiten zur Verfügung stehen, oder in einer Wohngemeinschaft mit anderen Klienten zu leben. Des Weiteren verfügt die Faehre über eine Begegnungsstätte, die es ermöglicht sich zum Austausch und zu Gesprächen zusammenzufinden. Darüber hinaus werden zahlreiche Freizeitgestaltungsmöglichkeiten angeboten, wie beispielsweise gemeinsame sportliche Aktivitäten oder auch ein gemeinschaftlicher Gesellschaftsspieleabend. Einige dieser Gruppenaktivitäten werden regelmäßig in der bereits erwähnten Begegnungsstätte der Faehre veranstaltet.
Im Zusammenhang mit den stetig wachsenden Freizeitangeboten, ist die Idee des sogenannten *Faehrbooks* entstanden. Das Faehrbook soll als Social Media Plattform für die Klienten dienen. Den Klienten soll es möglich sein, untereinander in einem Messenger zu kommunizieren, sowie einen Überblick über die angebotenen Veranstaltungen zu erhalten. Mit Hilfe der Nextcloud[7], welche im Kapitel 2.1 genauer vorgestellt wird, soll das Faehrbook umgesetzt werden. Diese bietet von vornherein die Funktionalität eines simpel aufgebauten Messengers. Für die Darstellung der Veranstaltungen innerhalb des Faehrbooks dient eine Applikation, welche im Rahmen dieser Bachelorarbeit entsteht.
Das Faehrbook bietet den Vorteil, dass diese Social Media Plattform selbst gehostet wird und somit die volle Kontrolle über jeglichen Datenverkehr, im Gegensatz zu alternativ Plattformen gewährleistet ist.

[6] [Die Faehre E.V.]
[7] [Nextcloud]

1.2 Motivation zur Gestaltung einer Veranstaltungsapplikation

Die Faehre ist ein eingetragener Verein mit Sitz in Hamburg und bietet psychisch erkrankten Menschen die Möglichkeit in einer Gemeinschaft, die speziell auf die Krankheiten angepasst ist und individuell auf die Bedürfnisse zugeschnitten ist, zu leben. Aufgrund der stetig wachsenden Anzahl an Hilfsbedürftigen und somit auch den immer vielfältigeren Aktivitätsangeboten, wünscht sich die Faehre eine übersichtliche und allgemein zugängliche Plattform für ihre Patienten. Die Anforderungen an diese Plattform sind bereits in den vorherigen Kapiteln ansatzweise erwähnt worden, jedoch folgt im nächsten Kapitel die genaue Umsetzungsidee für die Applikation, die im Rahmen dieser Bachelorarbeit entsteht. Die Ansprüche, beziehungsweise die Umsetzungsidee der Faehre an die Plattform übersteigen das zur Verfügung stehende Budget. Da die Kontor Consulting GmbH & Co. KG[8] offen für ehrenamtliche Tätigkeiten ist, wird das Projekt in Form dieser Bachelorarbeit umgesetzt. Als Nebenziel wird das Sammeln des Know-Hows zur Applikationsentwicklung für die Nextcloud verfolgt. Innerhalb der Kontor Consulting GmbH & Co. KG, mit der die Bachelorarbeit in Kooperation entsteht, wird die Nextcloud zum internen Datenaustausch eingesetzt. Aufgrund dessen besteht auch von Seiten der Kontor Consulting Interesse in Zukunft individuelle, selbst entwickelte Applikationen für die Nextcloud bereitzustellen. Der Aspekt die Nextcloud als Universalplattform einzusetzen überzeugt ebenfalls, da jederzeit die gesamte Kontrolle über das System besteht. Bei der Planung und Umsetzung der Applikation wird darauf geachtet, dass auch bei Aufrufen der Veranstaltungsapplikation des Faehrbooks auf einem mobilen Endgerät, wie zum Beispiel das Smartphone, eine angepasste, mobile Darstellung erfolgt. Dies bietet den interessanten Aspekt, dass die Applikation sowohl Desktopansprüchen, als auch den vielfältigen Anforderungen der Mobilgeräte gerecht werden muss.
Den spannendsten Aspekt der Entwicklung einer solchen Webapplikation, die Veranstaltungen mit Hilfe einer selbst gehosteten Nextcloud verwaltet und darstellt, bieten die eingesetzten Technologien. Deren Komplexität besteht in erster Linie darin, die Vielzahl der Technologien zusammenspielen zu lassen, sodass die Applikation mit gewünschter Funktionalität agieren kann. Ebenso muss sich genauer mit den einzelnen Technologien auseinandergesetzt werden, um mehr als die Grundfunktionalitäten jener Technologien in der Applikation umzusetzen. Nachfolgende Technologien sind in der Applikation angewandt:

- Nextcloud - MySQL Datenbank - PHP - HTML
- CSS - JavaScript - SQL

Im nachfolgendem Kapitel ist genauer erläutert, welche Eigenschaften die jeweiligen Technologien aufweisen und in welcher Form diese ihren Einsatz innerhalb der Applikation finden.

[8] [Kontor Consulting]

2 Theoretische Grundlagen

In diesem Kapitel sind die theoretischen und technischen Grundlagen erörtert, die notwendig sind, um die im Rahmen dieser Bachelorarbeit entstandenen Applikation entwickeln zu können. Wie bereits in den vorherigen Abschnitten erwähnt, bildet die Nextcloud die Basis und ist im anschließenden Unterkapitel genauer vorgestellt. Im darauffolgenden Kapitel ist beschrieben, welche Voraussetzungen mit der Konzeption der Applikation einhergehen. Unter anderem sind die ebenfalls im vorherigen Abschnitt aufgelisteten Technologien teil der Grundlagen für die Konzeption.

2.1 Nextcloud als Universalplattform

Das Unternehmen Nextcloud ist aus der Idee entstanden, den Anwendern die Kontrolle über deren Daten und Kommunikation zurückzugeben. Initiatoren waren Frank Karlitschek, Gründer von ownCloud[9], sowie weitere Open-Source-Unternhemen und -Ingenieure[10]. Die Nextcloud bietet eine selbst gehostete Fileserver Lösung an und ermöglicht somit eine komplette Kontrolle über jeglichen Datenverkehr. Der Grundgedanke hinter der Nextcloud ist, dass im Unternehmen alles an einem Ort stattfindet. In diesem Sinne bedeutet alles, dass die Nextcloud die Basis bildet und mittels integrierter Applikationen auf individuelle Ansprüche angepasst wird. Beispielsweise lässt sich mit Hilfe dieser Applikationen eine interne Kommunikation gestalten, die nicht über Drittanbieter abgewickelt wird. Weitere Anwendungsfälle sind beispielsweise das Nutzen eines geteilten Kalenders, Notizen oder Passwortverwaltung. Somit kann die Nextcloud auf nahezu jeden Anwendungsfall zugeschnitten werden. Falls es Szenarien gibt, die nicht mit Bordmitteln umzusetzen sind, gibt es die Möglichkeit, im eigenen App Store Applikationen herunterzuladen. Diese Applikationen können von Jedermann erstellt und im Store zum Download bereitgestellt werden. In dem Fall, dass man im Store nicht fündig wird, kann selbst eine Applikation entwickelt werden. Es besteht ebenfalls die Möglichkeit, selbst entwickelte Applikationen in den hauseigenen App Store zu stellen.
Aufgrund der Grundfunktionalitäten, die die Nextcloud bietet, sowie deren freie Gestaltungsmöglichkeit von weiteren Funktionen mit Hilfe von Applikationen, eignet sich diese als Grundgerüst, beziehungsweise Universalplattform für viele verschiedene Anwendungsfälle. Ein möglicher Anwendungsfall ist es, eine Nextcloud als Social Media Plattform zu etablieren. Dieser Einsatzzweck bildet den Grundgedanken für die webbasierte Applikation, die im Rahmen dieser Bachelorarbeit entsteht. Wie in vorherigen Kapiteln erläutert, soll das sogenannte Faehrbook entwickelt werden, basierend auf eine selbst gehostete Nextcloud. Im Kapitel 1.1 angeschnitten, bietet die Nextcloud die Möglichkeit Benutzer anzulegen, sowie eine gegenseitige Verständigung der Klienten. Die Kommunikation erfolgt unabhängig davon, in welcher Applikation man sich befindet.

[9] [ownCloud]
[10] [Nextcloud - About]

Realisiert wird dies mit Hilfe einer ausblendbaren Seitenleiste, welche in den folgenden Bildern dargestellt ist:

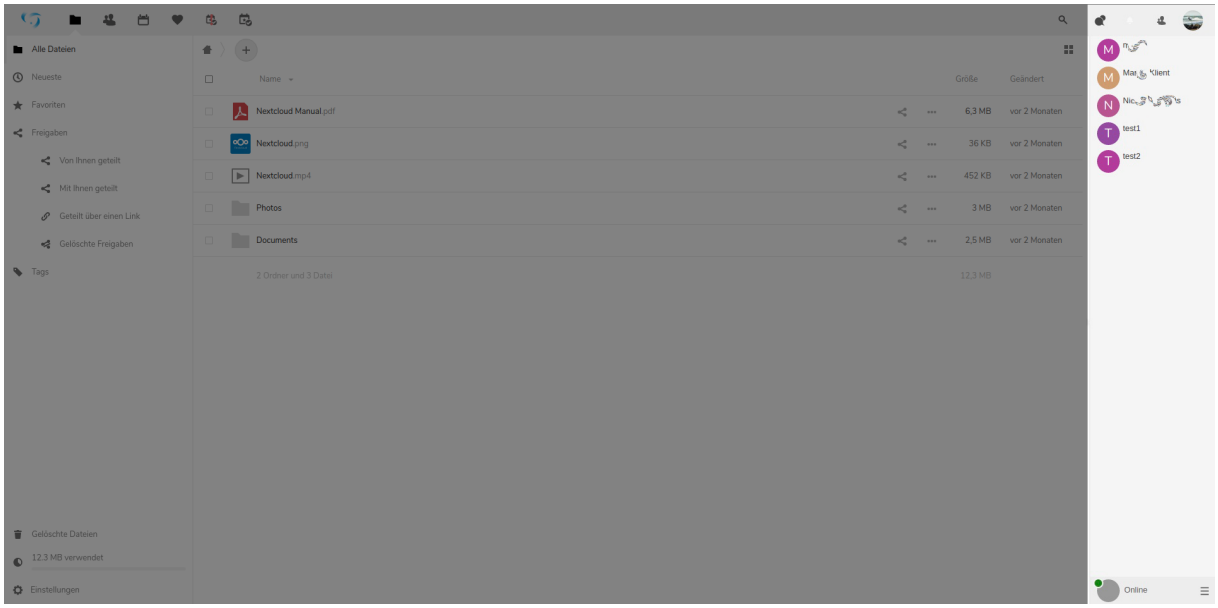


Abbildung 1: Chat in der Dateien Applikation, Namen sind unkenntlich gemacht

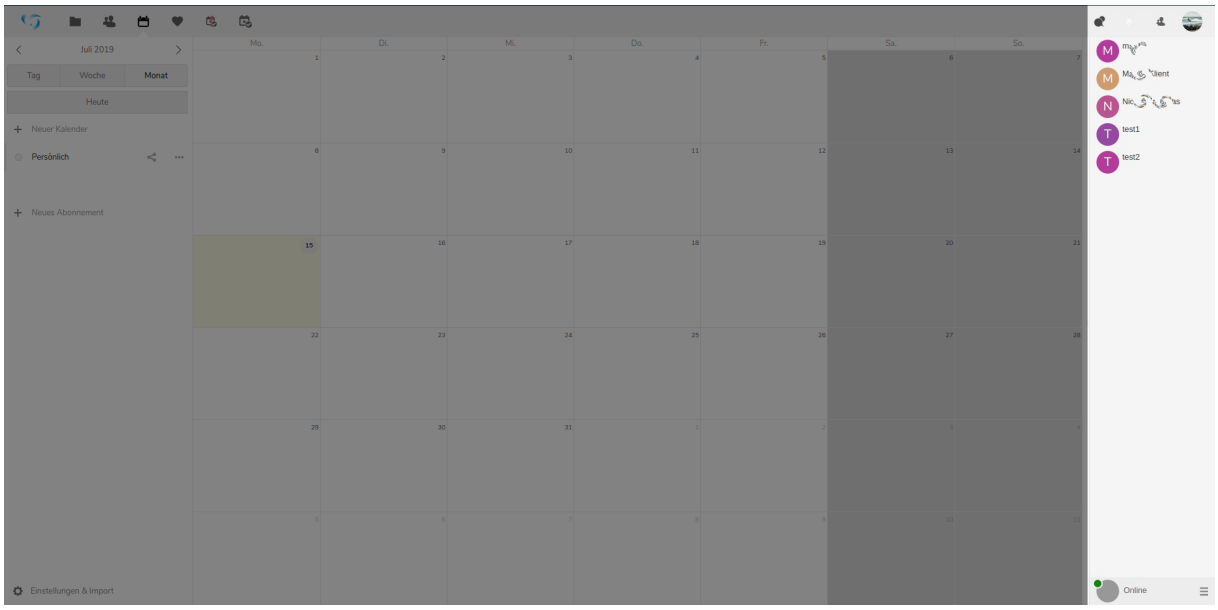


Abbildung 2: Chat in der Kalender Applikation, Namen sind unkenntlich gemacht

Die Chatfunktionalität steht anhand des Grundaufbaus der Nextcloud zur Verfügung und benötigt keinerlei Nacharbeit, um den gewünschten Funktionen gerecht zu werden. Neben dem Chat sollen Veranstaltungen für Klienten dargestellt und von Dozenten, beziehungsweise Verantwortlichen, verwaltet werden können. Im nachfolgenden Kapitel sind die technischen Voraussetzungen für die Erstellung einer Applikation für die Nextcloud beschrieben.

2.2 Voraussetzungen der Konzeption

In diesem Kapitel sind die technischen Voraussetzungen, die notwendig sind, um die webbasierte Applikation umzusetzen, erläutert. Die Auswahl der anzuwendenden Technologien ist zunächst dahingehend eingeschränkt, dass diese im Browser und somit im Web laufen müssen. Ein weiterer einschränkender Faktor ist, dass das Faehrbook mit der Nextcloud umgesetzt werden soll. Die Nextcloud bildet die Basis der Applikation und schränkt somit die Auswahl weiter ein, jedoch können alle gängigen Webtechnologien genutzt werden. Wie in Kapitel 1.2 bereits erwähnt, werden die folgenden Technologien eingesetzt:

- Nextcloud - MySQL Datenbank - SQL - PHP
- HTML - CSS - JavaScript

Nachgehend sind diese Technologien, sowie deren Einsatzbereiche innerhalb der Applikation genauer erläutert
Die Nextcloud ist bereits im vorherigen Kapitel 2.1 vorgestellt worden.
Das Grundgerüst der Nextcloud bildet eine Datenbank. MySQL ist ein Vertreter von möglichen Datenbanken und wird von der Nextcloud offiziell unterstützt. Aufgrund dessen fällt die Wahl der einzusetzenden Datenbank auf diese, die nach der Norm für Datenbanken aufgebaut ist. Datenbanken werden bei MySQL als Schemata bezeichnet, jedoch sind diese nicht speziellen Benutzern zugeordnet. Der Benutzer wählt selbst eine Datenbank, beziehungsweise ein Schema aus, um mit dieser zu arbeiten[11]. Jeder Server betreibt genau eine MySQL Datenbank, die mehrere Schemata beinhalten kann. Mit MySQL kann mit Hilfe einer textbasierten oder graphischen Benutzeroberfläche kommuniziert werden[12].
Die Technologie die eingesetzt wird, um mit einer Datenbank zu interagieren, heißt SQL. SQL steht für *Structured Query Language* und ist eine Zugriffsprache für Datenbanken. In der Anfangszeit von SQL war diese für den Endbenutzer gedacht, um mit Datenbanken zu arbeiten, jedoch hat sich dies im Lauf der Zeit gewandelt. Heutzutage werden GUIs eingesetzt. Mit der ersten Normierung im Jahr 1987 hat sich SQL zum Standard für die Kommunikation mit Datenbanken entwickelt. Die SQL2 Norm von 1992 ist bis heute völlig ausreichend für das alltägliche Arbeiten mit Datenbanken[13]. Lediglich vier Grundbefehle sind ausreichend für den Umgang mit Datenbanken mittels SQL. Diese Grundbefehle werden auch als *CRUD* bezeichnet. CRUD ist das Akronym aus den folgenden Begriffen[14]:

Create	→	Insert	→	Erstellung von Einträgen in Schemata
Read	→	Select	→	Lesen von Einträgen in Schemata
Update	→	Update	→	Aktualisieren von einträgen in Schemata
Delete	→	Delete	→	Löschen von Einträgen aus Schemata

[11] [Datenbanken und SQL, S.176]
[12] [Datenbanken und SQL, S.179f]
[13] [Datenbanken und SQL, S.99]
[14] [Datenbanken und SQL, S.139]

Eine weitere Technologie, die zum Einsatz kommt, ist PHP. PHP steht für *Hypertext Preprocessor* und ist eine Programmiersprache mit Anlehnung an Perl, sowie C. Entwickelt wurde PHP 1994 von Rasmus Lerdorf und ist für das Internet konzipiert worden. Um ein PHP Skript auszuführen, wird ein Webserver benötigt. Da PHP für den Einsatzzweck im Web entwickelt wurde, kann diese vollständig in HTML eingebettet werden. An welcher Stelle im HTML Skript PHP vorkommt, ist nicht von Bedeutung für das Ausführen des Programms. Anders als reine HTML Dateien, endet HTML Skript, welches PHP Code beinhaltet, mit der Endung *.php* statt *.html*[15].
PHP bildet das Grundgerüst für die Umsetzung der Applikation für das Faehrbook und muss somit auch mit der MySQL Datenbank interagieren können. Diese Funktionalität ist mit der PHP-Erweiterung *MySQLi*[16] realisiert. MySQLi bildet die Schnittstelle zwischen dem PHP Skript und der Datenbank. Somit ist eine simple Möglichkeit zur Kommunikation zwischen Webapplikation und Datenbank gegeben. Dabei übernimmt MySQLi den Verbindungsaufbau, sowie die Vermittlung von Anfragen an die Datenbank und die damit verbundene Rückmeldung[17].
Wie zuvor beschrieben, lassen sich PHP Skripte beliebig in HTML Skripte einpflegen. HTML steht für *Hypertext Markup Language* und bildet den weltweiten Standard für Webseiten. Entwicklet wurde HTML von Tim Burners-Lee im Jahr 1989. Innerhalb von HTML werden verschiedene Texte mittels Hyperlinks beziehungsweise Links verbunden und heben somit die lineare Struktur, die beispielsweise in einem Text aus einem Buch vorherrscht, auf. Damit wird *Hin- und Herspringen* innerhalb von Texten möglich. Mit der Idee des Internets, also dem World Wide Web, wird durch die Verteilung von Texten auf verschiedenen Endgeräten ein großes Informationssystem geschaffen. HTML besteht aus Elementen, die auch als Tags bezeichnet werden. Diese Elemente unterscheiden sich beispielsweise in Überschriften, Tabellen oder Links. Gespeichert wird ein HTML Skript als eine Textdatei, mit der Endung *.html*. Dies ermöglicht einen geringen Speicherbedarf und jeder Texteditor ist in der Lage, HTML Dateien zu editieren. Die geringe Datenmenge einer HTML Datei war zu den Anfangszeiten des Internets von großer Bedeutung, da die Internetverbindung, im Vergleich zu heute, langsam war. Bei der Entwicklung von HTML wurde nicht an Bilder, Videos und vergleichbares gedacht, da in dieser Zeit ein Computer über reine Textein- beziehungsweise -ausgabe mittels Terminal bedient wurde. Im Verlauf der Zeit wandelte sich dies hin zu graphischen Benutzeroberflächen. Der Wunsch nach graphischen Elementen wurde bei Internetseiten fortgeführt, jedoch bot HTML keine Möglichkeiten dafür. An dieser Stelle wurde CSS entwickelt[18].
CSS steht für *Cascading Style Sheet* und hat HTML ab 1996 erweitert. CSS wurde aufgrund des Aufkommens von graphischen Benutzeroberflächen entwickelt. CSS bietet unter anderem die Möglichkeiten, das Layout dynamisch an den Bildschirm des Endgerätes

[15] [Datenbanken und SQL, S.191f]
[16] [MySQLi]
[17] [Webtechnologien, S.80ff]
[18] [HTML5 und CSS3, S.2f]

anzupassen, verschiedene Schriftarten einzupflegen und Schriften in ihrem Aussehen zu verändern oder Animation zu verwenden. Designs lassen sich abhängig vom Endgerät ebenfalls mit Hilfe von CSS dynamisch anpassen. Die Konklusion daraus ist, dass HTML für den Inhalt und CSS für das Design einer Webseite zuständig sind. Ebenfalls sind CSS Dateien Textdateien und bieten somit die gleichen Vorteile, wie bei HTML. Zudem ermöglicht die Einteilung in Inhalt und Design, dass beispielsweise zwei Entwickler unabhängig voneinander an einer Webseite arbeiten können. Der eine Entwickler kümmert sich um den Inhalt, der andere um das Design. Des Weiteren können bereits entwickelte Designs mittels CSS Projektübergreifend eingesetzt werden[19].

Die letzte Technologie, die beim Entwickeln der Applikation für die Nextcloud zum Einsatz kommt, ist JavaScript. JavaScript wurd am 18. September 1995 von der Firma Netscape vorgestellt. Bei der Vorstellung hieß JavaScript jedoch noch LiveScript und wurde entwickelt von Brendan Eich. Im Zuge der Kooperation von Netscape und Sun Microsystems wurde LiveScript umbenannt in JavaScript, da in der Basis der Programmiersprache eine Schnittstelle zu Java vereinbart wurde. Ziel der Entwicklung von JavaScript war es, dass man eine einfache interaktive Funktionalität bereitstellt. Inspiriert von den Programmiersprachen Java und C unterscheidet sich JavaScript jedoch deutlich von den beiden. JavaScript ist eine Skriptsprache und wird nicht vom Compiler kompiliert, sondern von einem Interpreter zur Laufzeit transformiert. Der Compiler wandelt den gesamten Programmcode in Maschinencode vor der Laufzeit um, der Interpreter hingegen transformiert den aktuell aufgerufenen Befehl im JavaScript Code zur Laufzeit in Maschinencode. Im Gegensatz zum Compiler erzeugt der Interpreter bei erneutem Aufruf des gleichen Befehls den Maschinencode neu und verwendet nicht den bereits erstellten Maschinencode. Jeder aktuelle Browser unterstützt von Grund auf aus JavaScript. JavaScript ermöglicht das Ansprechen und Gestalten von HTML Elementen. Beispielsweise lässt sich mit Hilfe von JavaScript die Funktionalität eines gedrückten Knopfes realisieren. Die beiden Webtechnologien HTML und CSS werden somit um ein dynamisches Verhalten erweitert. Ebenso wie PHP, kann JavaScript Code in einer bestehenden HTML Datei eingebunden werden[20]. Zudem sind externe Dateien möglich, beziehungsweise der aktuelle Standard[21].

[19] [HTML5 und CSS3, S.44 bis 46]
[20] [Objektorierte Programmierung mit JavaScript, S.43f]
[21] [Objektorierte Programmierung mit JavaScript, S.46]

3 Konzeption der Applikation

Dieses Kapitel befasst sich mit der Konzeption der webbasierten Applikation zur Darstellung und Verwaltung von Veranstaltungen. Zunächst werden die Anforderungen von Seiten der Faehre und anschließend die Konzeption der Applikation dargestellt. Diese dient als Vorlage für die Implementierung, welche im darauffolgenden Kapitel behandelt wird.

Wie in den vorherigen Kapiteln erwähnt, soll eine Nextcloud die Basis bilden und als Social Media Plattform, das Faehrbook, für die Patienten der Faehre dienen. Wie in Kapitel 2.1 mit den Bildern verdeutlicht, ist die Funktionalität eines Chats bereits in der Nextcloud integriert. Die zweite große Anforderung an das Faehrbook ist eine Veranstaltungsapplikation, die in diesem Kapitel konzipiert ist. Die grobe Idee der Leitung der Faehre sieht wie folgt aus:

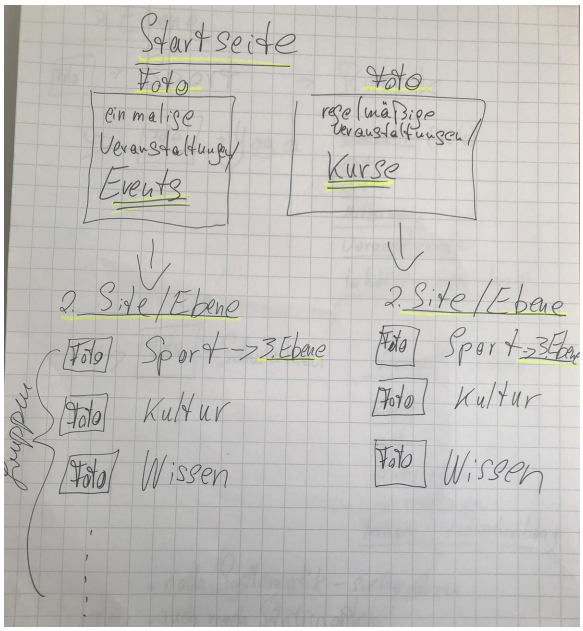


Abbildung 3: Konzeptionsidee Seite 1

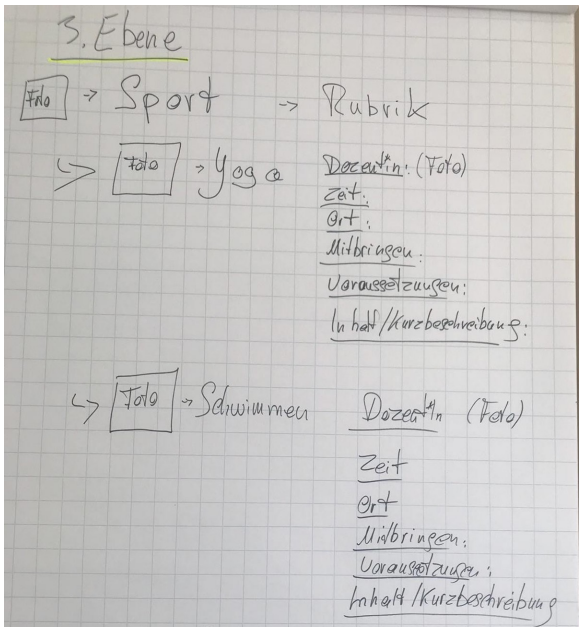


Abbildung 4: Konzeptionsidee Seite 2

Den Einstieg in die Applikation bildet die Startseite. Diese beinhaltet eine Auswahl zwischen Events und Kursen. Events sind Veranstaltungen, die einmalig abgehalten werden, wie beispielsweise eine jährliche Hauptversammlung. Kurse hingegen sind mehrmalige, beziehungsweise regelmäßige Veranstaltungen, wie zum Beispiel ein wöchentlicher Yoga Kurs. Bilder für die beiden Auswahlmöglichkeiten sollen die Übersichtlichkeit erhöhen, sodass sich die Benutzer schnell zurechtfinden, auch wenn Deutsch nicht die Muttersprache ist. Nach der Entscheidung für Events oder Kursen, folgt die zweite Seite beziehungsweise Ebene der Applikation. Diese bietet eine Übersicht über alle Rubriken, wie beispielhaft Sport oder Kultur. In dieser Ebene sind ebenfalls alle Kategorien mit Bildern dargestellt, sodass sich auch hier ohne Lesen der einzelnen Rubriken zurecht gefunden werden kann. Die jeweiligen Rubriken führen zur dritten Ebene, die innerhalb der ausgewählten Kategorie die einzelnen Veranstaltungen darstellt. Dargestellt werden die einzelnen Ver-

anstaltungen wieder mit passenden Bildern, die einem schnelleren Zurechtfinden dienen. Die Veranstaltungen beinhalten folgende Details:

Dozent*in (mit Bild)	**Zeit**
Ort	**Mitbringen (z.B. Badekleidung)**
Voraussetzungen	**Inhalt/ Kurzbeschreibung**

Im weiteren Verlauf der Konzeption ergaben sich folgende Änderungen von Seiten der Faehre. Das Bild für die jeweiligen Dozenten fällt weg, um möglichen Datenschutzproblemen vorzubeugen. Zudem wurde die Liste der Details einer Veranstaltungen um vier Punkte erweitert. Zusätzlich zu den bestehenden Punkten kommen die folgenden hinzu:

Bereich	**Kosten**
Maximale Teilnehmeranzahl	**Aktuelle Teilnehmerzahl**

Ein weiterer Faktor, welcher nicht bei der initialen Idee von Seiten der Faehre bedacht wurde, ist die Verwaltung der angebotenen Veranstaltungen. Das Angebot an Veranstaltungen soll verwaltet werden können, wofür eine weitere administrative Applikation benötigt wird.

Anhand dieser groben Vorgaben sind die nachfolgenden Applikationen konzipiert worden. Jedoch ist zunächst festzuhalten, dass Applikationen, die schon in der Nextcloud zur Verfügung stehen, wie beispielsweise der Kalender oder die Kontaktverwaltung, eine gemeinsame Designsprache sprechen. Solche Applikationen sind aufgeteilt in Spalten, durch diese man sich von links nach rechts navigiert. Dieses Grundkonzept ist bei der Konzeption beibehalten. Die erste Applikation übernimmt den Part der Darstellung und auch in gewisser Weise den der Verwaltung von Veranstaltungen. Die Verwaltungsmöglichkeit ist in sofern gegeben, dass die Klienten der Faehre ihre Teilnahmen an Veranstaltungen administrieren können. Aufgebaut ist die Applikation in drei Spalten. Die erste Spalte von links bildet den Einstieg und ist zu jeder Zeit erreichbar. In dieser Spalte gibt es drei Oberpunkte. Ganz oben steht *Meine Veranstaltungen* und bildet somit den ersten Punkt. Der nächsten Reiter sind *Kurse* und *Events*, welche jeweils die Rubriken als Unterpunkte aufweisen. In der Konzeption sieht dies folgendermaßen aus:

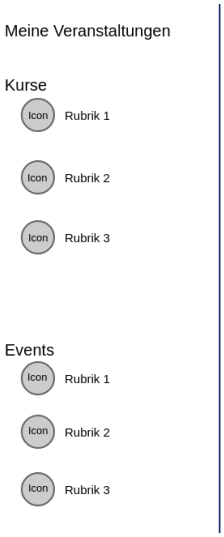


Abbildung 5: 1. Spalte Konzeption Veranstaltungsapplikation

Die zweite Spalte von links weist grundlegend den gleichen Aufbau auf, jedoch wird der Inhalt abhängig vom gewählten Punkt aus Spalte eins gewählt und angezeigt. Grundlegend wird die zweite Spalte mit den jeweilig zugehörigen Veranstaltungen gefüllt und zeigt die Titel dieser an. Dies sieht beispielsweise für Kurse und Rubrik 1 wie folgt aus:

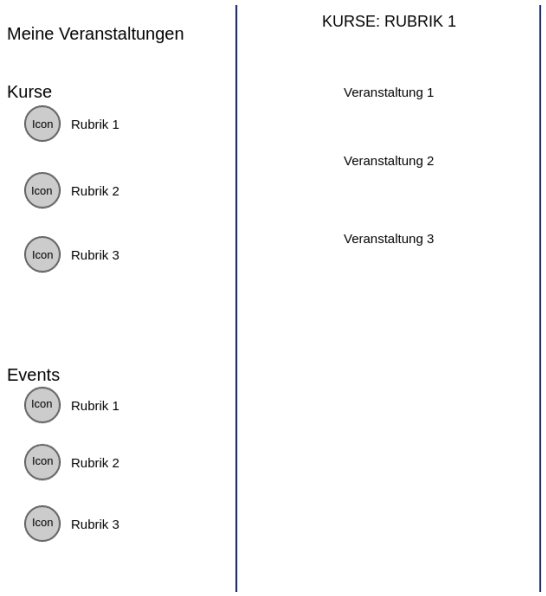


Abbildung 6: 1. und 2. Spalte Konzeption Veranstaltungsapplikation

Wie in Abbildung 6 zu sehen, wird oben angezeigt, in welcher Rubrik man sich befindet und ob es sich um einmalige oder regelmäßige Veranstaltung handelt. Verdeutlicht wird dies, indem entweder *Kurse* oder *Events* in der obersten Zeile steht. Unterhalb dieser Zeile werden alle Veranstaltungen der Rubrik aufgelistet, die sich in der Datenbank befinden. Bei einem Klick auf eine der dargestellten Veranstaltungen gelangt man zur Detailansicht, welche die dritte Spalte bildet und folgendermaßen aussieht:

Abbildung 7: 1., 2. und 3. Spalte Konzeption Veranstaltungsapplikation

Der Knopf unten in der Detailansicht verändert seinen Text und Funktionalität dynamisch

anhand verschiedener Umstände. Im Grunde hat der Knopf vier Zustände. Einzuteilen sind die vier Zustände in zwei Gruppierungen, entweder ist der Knopf sichtbar oder nicht sichtbar. Der Knopf ist nicht sichtbar und es wird eine Meldung angezeigt. Die Meldung kommt dann, wenn der Kurs oder das Event die Teilnehmerbegrenzung erreicht hat oder wenn eine Veranstaltung aktuell nicht aktiv ist. Wie in Abbildung 7 zu sehen, weist der Knopf den Text *Teilnehmen* auf. Dieser Text wird angezeigt, wenn man sich zur Veranstaltung durchklickt, wie in den Abbildungen zu sehen. Mit dem Knopf kann ein Klient an einer Veranstaltung teilnehmen. Wenn hingegen der Klient bereits teilnimmt, löst der Knopf keine Aktion aus. Nachdem ein Klient an einer Veranstaltung teilnehmen will und somit auf den Knopf drückt, wird dieser weitergeleitet zu seinen Veranstaltungen. Wählt der Klient dort eine seiner Veranstaltungen aus, ist die Detailansicht identisch, bis auf der Text und Funktionalität des Knopfes. In diesem Fall weist der Knopf den Text *Nicht mehr Teilnehmen* auf. Beim Klick wird der Teilnehmer aus der Veranstaltung und die Veranstaltung aus der persönlichen Übersicht entfernt.
In diesem Abschnitt ist der Part der Konzeption beschrieben, welcher sich mit der Administration von Veranstaltungen befasst. Für die Verwaltung der Veranstaltungen wird eine simple, eigenständige Applikation zusätzlich konzipiert und implementiert. Die Abgrenzung zur Applikation, die für die Darstellung zuständig ist, folgt aus dem Hintergrund, dass bei der Nextcloud Applikationen verschiedenen Gruppen zugeordnet werden kann. Diese Gruppen sind explizit beim Faehrbook unterteilt in Dozenten, welche zu den Administratoren gehören, und Klienten. Die Klienten haben lediglich Zugriff auf die Applikationen, die für sie freigegeben sind. Im Gegensatz zur Veranstaltungsapplikation, beinhaltet die Administrationsapplikation zwei statt drei Spalten zum Navigieren. In der linken Spalte ist ebenfalls die Navigation mittels Rubriken untergebracht. Die folgenden Funktionalitäten sind vorhanden:

Veranstaltung erstellen	**Veranstaltung aktualisieren**
Veranstaltung löschen	**Teilnehmer entfernen**
Rubriken Übersicht	**Neue Rubrik erstellen**
Rubrik löschen	

Wenn ein Dozent eine neue Veranstaltungen einstellen will, wählt dieser den Reiter *Veranstaltung erstellen* und trägt die entsprechenden Daten der Veranstaltungen in die vorgesehenen Felder ein. Pflichtfelder für eine neue Veranstaltung sind die Felder für *Art*, *Rubrik* und *Titel*. Für die *Art* ist gemeint, dass es sich entweder um ein Event, also ein einmaliges Ereignis, oder einen Kurs, also ein mehrmaliges Ereignis, handelt. In diesem Feld werden, je nach Art der Veranstaltung, ausschließlich *event* und *kurs* für die Eingabe akzeptiert. So ist sichergestellt, dass es nicht zu einem Eingabefehler kommen kann, welcher somit dazu führt, dass die Veranstaltung sich nicht richtig Zuordnen und Anzeigen lässt. Jede Veranstaltunge ist einer bestimmten *Rubrik* zugeordnet, sodass eine Sortierung alle Ereignisse entsteht. Wie bereits erwähnt, handelt es sich um ein Pflichtfeld. In diesem Feld werden nur Rubriken akzeptiert, die sich innerhalb der Datenbank befinden und somit aktuell sind. Die Eingabe für *Art* und *Rubrik* werden mit Hilfe eines regulären

Ausdrucks geprüft und nur akzeptiert, wenn die Eingabe dem Ausdruck entspricht. Ein regulärer Ausdruck kann beispielsweise sein, dass eine Eingabe immer mit einer Zahl beginnen muss. So kann überprüft werden, dass es nicht zu Tippfehlern oder ähnliches bei der Eingabe kommt. Die weiteren Felder sind:

Bereich	**Dozent**
Zeit	**Ort**
Mitbringen	**Voraussetzungen**
Maximale Teilnehmeranzahl	**Kosten**
Details	**Aktiv**

Alle Felder nehmen einen Text als Input außer das Feld für die maximale Anzahl an Teilnehmer. Die maximale Teilnehmeranzahl akzeptiert nur Zahlen, die nicht negativ sind. Die Kontrolle, dass die eingegeben Zahlen nicht negativ sind, wird vom Eingabefeld selbst kontrolliert. Wenn eine **0** eingetragen wird, steht die für eine unbegrenzte Teilnehmerzahl. In der Detailansicht einer Veranstaltung wird *keine Begrenzung* für die maximale Teilnehmerzahl angezeigt, wenn im Datenbankeintrag eine **0** für jene Anzahl steht. Für das Feld *Aktiv* wird ebenfalls ein regulärer Ausdruck verwendet, um sicherzustellen, dass nur eine **1** oder eine **0** eingetragen wird. Eine **0** steht für Veranstaltungen, die entweder pausiert oder inaktiv ist, hingegen eine **1** aktive Veranstaltungen symbolisiert. Sind diese Felder beim speichern nicht ausgefüllt, werden voreingestellte Werte automatisch von der Datenbank eingetragen. Für alle Textfelder, außer *Maximale Teilnehmeranzahl* und *Aktiv*, wird *'-'* eingetragen. Initial wird eine neue Veranstaltung aktiv und ohne Begrenzung der Teilnehmeranzahl erstellt. Unten auf der Seite befindet sich der Knopf, der für das Sichern der Veranstaltung zuständig ist. Zur Hilfe, welche Rubrik möglich ist für einen Eintrag, wird eine Übersicht rechts von den Eingabefeldern angezeigt.

Im Gegensatz zum neu Erstellen, kann eine Veranstaltung aktualisiert werden, wenn sich beispielsweise die Zeit oder der Ort ändert. In diesem Fall wählt der Dozent die zu aktualisierende Veranstaltung aus einem Dropdown-Menü aus und die bisherigen Einträge werden mit Eingabefeldern, wie beim neu Erstellen, dargestellt. Anders als beim hinzufügen von neuen Veranstaltungen, sind keine der Felder verpflichtend zu füllen. Jedoch gelten bei den Feldern *Art, Rubrik, maximale Teilnehmeranzahl und Aktiv* die gleichen Vorgaben. Ebenso ist ein Knopf vorhanden, der die Änderungen speichert und in die Datenbank schreibt.

Ist eine Veranstaltung obsolet, kann sie aus der Datenbank entfernt werden, sodass diese nicht mehr den Klienten angezeigt wird. Dies geschieht im nächsten Reiter der linken Spalte. Dort lässt sich eine Veranstaltung auswählen, die mittels Knopf aus der Datenbank entfernt wird. Es kann immer nur eine Veranstaltung ausgewählt und gelöscht werden, um ein versehentliches Löschen von mehreren Ereignissen zu verhindern.

Falls ein Teilnehmer sich zu einem Kurs angemeldet hat, jedoch nicht mehr zu den Terminen erscheint, blockiert dieser einen freien Platz. Um dies zu verhindern, gibt es die Möglichkeit, einen Teilnehmer aus einer Veranstaltung zu entfernen. Damit wird einem interessierten Klient ermöglicht, an der Veranstaltung teilzunehmen. Wie beim aktualisie-

ren der Veranstaltungen, wird der zu entfernende Teilnehmer aus einem Dropdown-Menü ausgewählt. Die Teilnehmer sind den jeweiligen Veranstaltungen zugeordnet und es tauchen nur Teilnehmer auf, die zugesagt haben. Nach der Auswahl wird der Teilnehmer mittels Knopf aus der Liste der Veranstaltung entfernt und das Ereignis erscheint nicht mehr im Reiter *Meine Veranstaltungen* des Klienten.
Mit dem Reiter *Rubriken Übersicht* kann man sich alle Rubriken anzeigen lassen, die in der Datenbank vorhanden sind. Die letzten beiden Reiter der linken Spalte sind *neue Rubrik erstellen* und *Rubrik löschen*. Wie sich aus den beiden Titeln schließen lässt, kann eine neue Rubrik erstellt und eine bestehende gelöscht werden. Um eine neue Rubrik zu erstellen, gibt man den Titel ein und speichert diesen in der Datenbank mittels Knopf ab. Der Titel ist verpflichtend für das Erstellen. Ohne Titel löst der Knopf keine Aktion aus. Das Löschen einer bestehenden Rubrik ähnelt dem Löschen einer Veranstaltung. Es werden alle aktuellen Rubriken aufgelistet, von denen ausschließlich eine ausgewählt werden kann, die anschließend durch das Drücken des Knopfes aus der Datenbank gelöscht wird. Im nachfolgendem Kapitel wird die Konzeption der beiden Applikationen implementiert.

4 Implementierung der Applikation

In diesem Kapitel ist erläutert, wie die Implementierung der vorherigen Konzeption erfolgt und ist aufgeteilt in drei Abschnitte. Der erste Abschnitt befasst sich mit den grundlegenden Voraussetzungen der Implementierung, wie beispielsweise die Entwicklungsumgebung. Im nächsten Abschnitt ist zunächst die Implementierung der Applikation, die sich um die Darstellung der Veranstaltungen kümmert, beschrieben. Der letzte Abschnitt in diesem Kapitel befasst sich mit der Umsetzung der Applikation zum administrieren der Veranstaltungen.
Da die Applikationen für die Nextcloud gedacht sind, benötigt man diese für das Testen der Anwendungen. Das Aufsetzen einer Nextcloud wird nicht durchgeführt, weil bereits eine bestehende innerhalb der Kontor Consulting GmbH & Co. KG für Testzwecke bereitsteht. Für den Einstieg bietet die Nextcloud auf ihrer Internetseite den Download einer Vorlage für Applikationen an[22]. Diese Vorlage bildet die Grundlage der beiden Applikationen, die im Rahmen dieser Bachelorarbeit entstehen. Die sogenannte *Skeleton App* hilft beim Verstehen des grundlegenden Aufbaus einer Nextcloudapplikation und ermöglicht die Anpassung von vorhandenen Elementen für die eigenen Zwecke.
Der Ordner einer Applikation wird in dem Verzeichnis *nextcloud/apps/*[23] untergebracht und ermöglicht, dass man diese Applikation innerhalb der Nextcloud Einstellungen findet und aktivieren kann. Ebenfalls im Entwicklerhandbuch zu lesen, kann jedes JavaScript Framework[24] eingesetzt werden, jedoch wird bei diesen Applikation darauf verzichtet, da diese möglichst simpel und schlank sein sollen. Ebenfalls bedeutet der Einsatz eines JavaScript Frameworks, dass Zeit investiert werden muss, um sich in jenes einzuarbeiten. Des Weiteren sollen nur bekannte und grundlegende Techniken, wie reines PHP, eingesetzt werden. Je kleiner die Applikation gehalten ist, desto schneller ist diese im Browser, da der Server weniger Daten abrufen und durch das Internet senden muss. Als Entwicklungsumgebung kommt Visual Studio Code[25] von Microsoft zum Einsatz, da diese bereits in anderen Projekten verwendet wurde.
Da die Nextcloud auf einem eigenständigen Server läuft, wird die Visual Studio Code Erweiterung *SFTP*[26] eingesetzt, um den lokal geschrieben Code an den Server zu senden. SFTP steht für SSH File Transfer Protocoll oder auch Secure File Transfer Protocol und ermöglicht eine sichere und verschlüsselte Verbindung zum Server, um Dateien auszutauschen. Damit der Überblick über das Projekt jeder Zeit gegeben ist und es zu keinem Datenverlust kommen kann, wird zur Versions- und Codeverwaltung GitLab eingesetzt.
Wie bereits im Kapitel 2.2 erwähnt, wird als Datenbank MySQL eingesetzt. In dieser existieren zwei grundlegende Tabellen. Zum einen ein Schema für die Veranstaltungen, zum anderen eine Tabelle für die Rubriken. Die Tabelle der Rubriken beinhaltet nur zwei

[22] [Nextcloud Skeleton App]
[23] [Nextcloud Developer Intro]
[24] [Nextcloud JavaScript Frameworks]
[25] [Visual Studio Code]
[26] [Visual Studio Code Erweiterung SFTP]

Einträge, nämlich die *RubrikID* und den *Titel*. Das Schema wird mit folgendem SQL Statement initialisiert:

Codeverzeichnis 1: SQL Statement Erzeugung Rubriken Tabelle

```
create table nextcloud.Faehrbook_Rubriken (
  RubrikID int primary key not null auto_increment,
  Titel varchar(255) not null
);
```

Hingegen beinhaltet die Tabelle für die Veranstaltungen mehr Einträge und wird wie folgt erstellt:

Codeverzeichnis 2: SQL Statement Erzeugung Veranstaltungen Tabelle

```
create table Faehrbook_veranstaltungen (
  VeranstaltungsID int primary key not null auto_increment,
  Art varchar(255) not null,
  Rubrik varchar(255) not null,
  Titel varchar(255) not null,
  Bereich varchar(255) not null DEFAULT '-',
  Dozent varchar(255) not null DEFAULT '-',
  Zeit varchar(255) not null DEFAULT '-',
  Ort varchar(255) not null DEFAULT '-',
  Mitbringen varchar(1000) DEFAULT '-',
  Voraussetzungen varchar(1000) DEFAULT '-',
  Teilnehmer_Max int unsigned not null DEFAULT 0,
  Teilnehmer_Current int unsigned not null DEFAULT 0,
  Teilnehmer varchar(255) DEFAULT '',
  Kosten varchar(255) DEFAULT '-',
  Details varchar(2550) not null DEFAULT '-',
  Aktiv boolean not null DEFAULT true
);
```

Es werden weitere Tabellen angelegt, jedoch dynamisch innerhalb der Applikation. Jeder Klient, beziehungsweise User, erhält seine eigene Tabelle, in der seine Veranstaltungen hinterlegt werden. Wie genau dieses Schema erstellt und befüllt wird, ist später im Kapitel genauer erläutert.
Im kommenden Abschnitt wird die Implementierung der Applikation für die Verwaltung der Veranstaltungen genauer behandelt. Den Einstieg in die Implementierung bildet die Navigation und der Grundaufbau der Applikation. Wie im Kapitel 3 konzipiert, geschieht die Navigation mittels drei Spalten. Beim Starten der Applikation wird zunächst nur die erste, also linke Spalte, angezeigt. Mittels dieser Spalte navigiert man sich durch die komplette Applikation und ist zu jeder Zeit sichtbar. Diese Spalte ist in PHP wie folgt implementiert:

Codeverzeichnis 3: Veranstaltungsapplikation linke Spalte 1

```
1  <?php
2    $rubriken = getRubriken();
3  ?>
```

Mittels dem obigen PHP Code, Codeverzeichnis 3, werden erstens die Rubriken aus der Datenbank bezogen, sowie in einer Variable gespeichert. Wie genau die Daten aus der Datenbank bezogen werden, ist später im Abschnitt erläutert.

Mit dem nachfolgendem Code wird anhand der Rubriken die Liste für die Spalte erstellt.

Codeverzeichnis 4: Veranstaltungsapplikation linke Spalte 2

```
1   <ul>
2     <li>
3       <?php
4         echo "<a href=\"&fav=yes\">Meine Veranstaltungen</a>";
5       ?>
6     </li>
7     <li>
8       <?php
9         $initial = str_replace(' ', '', $rubriken[0]['Titel']);
10        $initial = strtolower($initial);
11        echo "<a href=\"?art=kurs&rubrik=" . $initial . "\">Kurse</a>";
12      ?>
13      <ul>
14        <?php
15          foreach ($rubriken as $key => $value)
16          {
17            $titel = str_replace(' ', '', strtolower($value['Titel']));
18            echo "<li>";
19              echo "<a href=\"?art=kurs&rubrik=" . $titel . "\">";
20                echo "<div>";
21                  echo "<img src=\"/apps/faehrbookevents/img/rubriken/" . $value['
                        Titel'] . ".png\"/>";
22                  echo "<div class=\"kategorien\">" . $value['Titel'] . "</div>";
23                echo "</div>";
24              echo "</a>";
25            echo "</li>";
26          }
27        ?>
28      </ul>
29    </li>
30    <li>
31      <?php
32        echo "<a href=\"?art=event&rubrik=" . $initial . "\">";
33          echo "Events";
34        echo "</a>";
35      ?>
36      <ul>
37        <?php
38          foreach ($rubriken as $key => $value)
39          {
40            $titel = str_replace(' ', '', strtolower($value['Titel']));
41            echo "<li>";
42              echo "<a href=\"?art=event&rubrik=" . $titel . "\">";
43                echo "<div>";
44                  echo "<img src=\"/apps/faehrbookevents/img/rubriken/" . $value['
                        Titel'] . ".png\"/>";
45                  echo "<div class=\"kategorien\">" . $value['Titel'] . "</div>";
46                echo "</div>";
47              echo "</a>";
48            echo "</li>";
49          }
50        ?>
51      </ul>
52    </li>
53  </ul>
```

Der $< ul >$ *Tag*, wie beispielsweise im Codeverzeichnis 4 Zeile 1, steht für eine Liste mit Listenelementen, die mittels $< li >$, Codeverzeichnis 4 Zeile 2, erzeugt werden. Der erste Listeneintrag lautet *Meine Veranstaltungen* und leitet mittels $< a >$ weiter zur mittleren Spalte. Genauso verhält es sich mit den Rubriken, die zusätzlich ein Icon haben, um einen besseren Überblick zu verschaffen. Die mittlere, sowie rechte Spalte gehören zusammen und befinden sich in einer Datei. Der Inhalt für die mittlere Spalte ergibt sich anhand der linken. Wiederum orientiert sich die rechte Spalte an der mittleren. Die jeweiligen, spezifischen Inhalte werden dynamisch erzeugt und angezeigt. Die beiden Listenelemente *Kurse* und *Events* können ebenfalls gedrückt werden und leiten initial zur ersten Rubrik weiter. Die Weiterleitung mittels der jeweiligen Rubriken oder den bereits erwähnten Rei-

tern erfolgt durch den zuvor erwähnten $< a >$ *Tag* und einem *href*. In dem *href* werden Variablen gesetzt, die abgerufen werden können. Anhand dieser Variablen setzt sich der spezifische Inhalt der dargestellten Seite zusammen. Für den Reiter *Kurse* sieht der *href* mit der beispielhaften Daten für eine Rubrik *Sport* folgendermaßen aus:
?art=kurs&rubrik=sport
In diesem String werden den drei Variablen *userid, art und rubrik* die Beispieldaten zugewiesen. Diese werden für die Darstellung der nächsten Spalte notwendig. In der fertigen Applikation sieht die linke Spalte und somit auch die Startseite wie folgt aus:

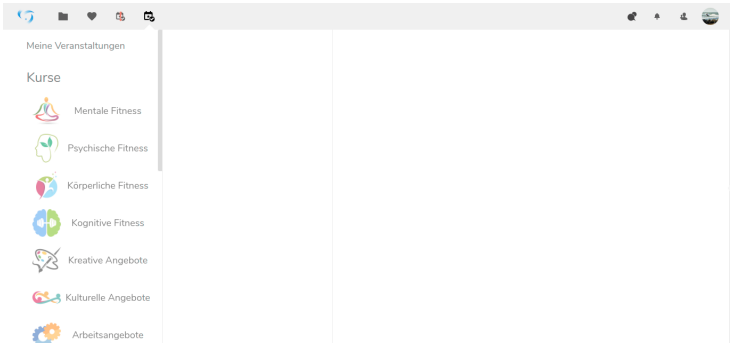


Abbildung 8: 1. Spalte Implementierung Veranstaltungsapp

Nach dem Klick auf einem der Reiter führt dies zur nächsten Seite in der Applikation. Wie aus dem Kapitel 3 zur Konzeption hervorgeht, wird in der mittleren Spalten eine Auswahl an Veranstaltungen dargestellt. Zusehen im nachfolgendem Bild:

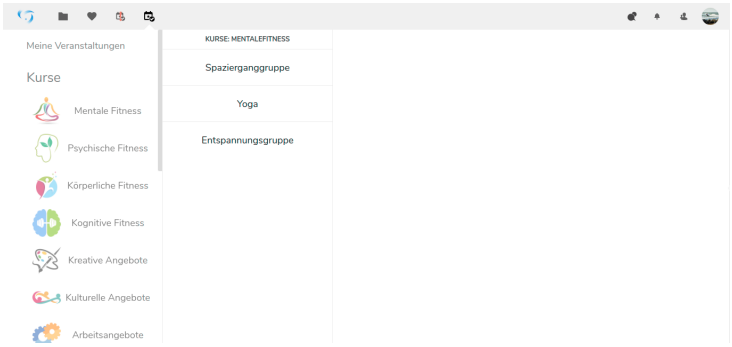


Abbildung 9: 2. Spalte Implementierung Veranstaltungsapp

Wie zuvor beschrieben, richtet sich dieser Inhalt nach den Variablen und dem User, der gerade angemeldet ist. Der aktuelle Benutzer wird, wie im nachfolgenden Code zu sehen, abgefragt. Des Weiteren werden die Inhalte von drei Variablen, die sich, nach einem Klick in der linken Spalte, in der URL befinden können, abgefragt. Anhand dieser wird festgelegt, welcher Inhalt in der mittleren Spalte dargestellt werden soll.

Codeverzeichnis 5: Veranstaltungsapplikation mittlere Spalte 1

```
1  <?php
2
3     $user = \OC::$server->getUserSession()->getuser();
4     $userid = str_replace(' ', '%', $user->getUID());
5
6     $art = $_GET['art'];
7     $rubrik = $_GET['rubrik'];
8     $favAnzeigen = $_GET['fav'];
```

Ebenfalls wird, wie nachfolgend zu sehen, mittels einer *Switch Anweisung* der erste Teil des Titels für die zweite Spalte festgelegt.

Codeverzeichnis 6: Veranstaltungsapplikation mittlere Spalte 2

```
1   switch ($art) {
2     case 'event':
3       $r_art = 'events: ';
4       break;
5     case 'kurs':
6       $r_art = 'kurse: ';
7       break;
8     default:
9       $r_art = '';
10      break;
11  }
12 ?>
```

Anhand zuvor festgelegten Variable für die Art, wird die mittlere Spalte folgendermaßen mit Inhalt gefüllt:

Codeverzeichnis 7: Veranstaltungsapplikation mittlere Spalte 3

```
1  <div id="app-content-wrapper">
2    <div class="app-content-list">
3      <ul>
4      <?php
5        if($favAnzeigen === 'yes')
6        {
7          $events = getMyEvents(str_replace(' ', '_', $user->getUID()));
8          echo "<h2 id=\"head2\">Meine Veranstaltungen</h2>";
9          foreach ($events as $key => $value)
10         {
11           echo "<a href=\"?fav=yes&id=" . $value['VeranstaltungsID'] . "\" class
                 =\"app-content-list-item\">";
12           echo "<div class=\"app-content-list-item-line-one\">" . $value['Titel']
                 . "</div>";
13           echo "</a>";
14         }
15       }
16       else
17       {
18         $events = getAll($art);
19         echo "<h2 id=\"head2\">" . strToUpper($r_art) . " " . strToUpper($rubrik)
               . "</h2>";
20         foreach ($events as $key => $value)
21         {
22           if($value['Art'] === $art && str_replace(' ', '', strtolower($value['
                 Rubrik'])) === $rubrik)
23           {
24             echo "<a href=\"?&art=" . $art . "&rubrik=" . $rubrik . "&id=" .
                   $value['VeranstaltungsID'] . "\" class=\"app-content-list-item\">"
                   ;
25             echo "<div class=\"app-content-list-item-line-one\">" . $value['Titel'
                   ] . "</div>";
26             echo "</a>";
27           }
28         }
29       }
30     ?>
31   </ul>
32 </div>
```

Die Ids und Klassen *app-content-wrapper, app-content-list, app-content-list-item und app-content-list-item-line-one* sind Elemente, die von der Nextcloud bereitgestellt werden, um eine Spalte darzustellen. Beispiele dazu lassen sich im Entwicklerhandbuch der Nextcloud finden[27]. In Zeile 5 ist zu sehen, dass abgefragt wird, welchen Inhalt die Variable für favorisierte Veranstaltungen aufweist. Wenn ein *yes* gesetzt ist, werden nur die Veranstaltungen angezeigt, zu denen der jeweilige Benutzer zugesagt hat. Ist diese Variable anders gesetzt, wird der Code im *else* Zweig ausgeführt. Zunächst werden alle Veranstaltungen anhand

[27] [Nextcloud Design Guideline Content]

der Art aus der Datenbank bezogen, zu sehen in Zeile 18. Anschließend wird der Titel für die Spalte ausgegeben. Danach werden die jeweiligen Veranstaltungen aufgelistet und bilden ebenso per $< a >$ *Tag*, sowie *href* einen Link zur dritten, rechten Spalte. Nach diesen Links richtet sich die Detailansicht einer Veranstaltung. Die Veranstaltungstitel bilden die Titel der einzelnen Reiter. Mit dem Klick auf eine Veranstaltung werden den folgenden Variablen Werte zugeteilt:

Art, Rubrik und Id

Die Id der Veranstaltung wird verwendet, um den Index im Array zu finden, in dem alle Veranstaltungen gespeichert sind. Das Finden es Indexes geschieht im Code folgendermaßen:

Codeverzeichnis 8: Veranstaltungsapplikation rechte Spalte 1

```
1  <?php
2    $id = $_GET['id'];
3    $index = 0;
4
5    if($id != null)
6    {
7      foreach ($events as $key => $value)
8      {
9        if($value['VeranstaltungsID'] === $id)
10         $index = $key;
11     }
```

Nachdem der Index gefunden ist, wird genau diese Veranstaltung in der dritten und somit letzten Spalte dargestellt. Die Darstellung erfolgt mit dem Ausführen des nachfolgenden Codes.

Codeverzeichnis 9: Veranstaltungsapplikation rechte Spalte 2

```
1    echo "<div id=\"app-content-details\">";
2      echo "<header id=\"header1\">";
3        echo "<h2 id=\"header2\">" . $events[$index]['Titel'] . "</h2>";
4      echo "</header>";
5      echo "<div id=\"details\">";
6        echo "<img class=\"veranstaltunge-img\" src=\"/apps/faehrbookevents/img/
             veranstaltungen/" . $events[$index]['Rubrik'] . '_' . $events[$index]['
             Titel'] . ".jpeg\" />";
7        echo "<div style=\"display: grid;\">";
8          echo "<div style =\"grid-column: 1;\">";
9            echo "<div><div>Dozent*in:</div>" . str_replace("\n", '<br>', $events[
                 $index]['Dozent']) . "</div>";
10           echo "<div><div>Bereich:</div>" . str_replace("\n", '<br>', $events[
                 $index]['Bereich']) . "</div>";
11           echo "<div><div>Zeit:</div>" . str_replace("\n", '<br>', $events[$index
                 ]['Zeit']) . "</div>";
12           echo "<div><div>Ort:</div>" . str_replace("\n", '<br>', $events[$index][
                 'Ort']) . "</div>";
13           echo "<div><div>Mitbringen:</div>" . str_replace("\n", '<br>', $events[
                 $index]['Mitbringen']) . "</div>";
14           echo "<div><div>Voraussetzungen:</div>" . str_replace("\n", '<br>',
                 $events[$index]['Voraussetzungen']) . "</div>";
15
16           $max = ($events[$index]['Teilnehmer_Max'] == 0) ? 'Keine Begrenzung' :
                 $events[$index]['Teilnehmer_Max'];
17
18           echo "<div><div>Teilnehmer Maximal:</div>" . $max . "</div>";
19           echo "<div><div>Teilnehmer angemeldet:</div>" . $events[$index]['
                 Teilnehmer_Current'] . "</div>";
20           echo "<div><div>Kosten:</div>" . str_replace("\n", '<br>', $events[
                 $index]['Kosten']) . "</div>";
21           echo "<div><div>Details:</div>" . str_replace("\n", '<br>', $events[
                 $index]['Details']) . "</div>";
```

Der Code aus dem Codeverzeichnis 9 erzeugt die Detailansicht einer Veranstaltung. Der zuvor gefundene Index wird dazu verwendet, die Einzelheiten aus dem Array mit allen

Veranstaltungen herauszulesen und anschließend darzustellen. In Zeile 16 wird anhand der maximalen Teilnehmerzahl unterschieden, was angezeigt werden soll. Wie im Kapitel 3 bereits beschrieben, beschreibt eine 0, dass keine Begrenzung existiert. Damit der Benutzer einen besseren Überblick erhält, wird anstatt der 0 *Keine Begrenzung* dargestellt. Ebenfalls im Kapitel 3 erläutert, soll es dem Benutzer möglich sein, an Veranstaltungen teilzunehmen. Die ist realisiert mit einem Knopf, beziehungsweise einem Text. Die Auswahl, welches der beiden Elemente angezeigt wird, ist abhängig von zwei Faktoren. Zum einen von der aktuellen Teilnehmerzahl, im Hinblick auf die maximale Anzahl, und zum anderen, in wie weit die Veranstaltung aktiv, beziehungsweise inaktiv ist. Ist entweder die maximale Anzahl an Teilnehmern erreicht oder die Veranstaltung inaktiv, so wird *Keine Teilnahme möglich!* statt einem Knopf angezeigt. Ist jedoch eine Teilnahme möglich, so wird ein Knopf angezeigt, welcher zwei Zustände einnehmen kann. Der Knopf befindet sich im ersten Zustand, wenn die Detailansicht von einer Rubrik ausgeht. In diesem Fall weist der Knopf den Text *Teilnehmen* auf. Beim Druck auf *Teilnehmen*, wird geprüft, ob der Benutzer bereits an der Veranstaltung teilnimmt. Nimmt dieser teil, passiert nichts weiter. Im Gegensatz dazu nimmt der Benutzer an der Veranstaltung teil und findet diese unter seinen Veranstaltungen. Befindet sich der Benutzer in Übersicht über seine Veranstaltungen, hat der Knopf den Text *Nicht mehr Teilnehmen*. Mit dem Klick auf diesen, meldet sich der Benutzer von der jeweiligen Veranstaltung ab. Die genaue Umsetzung einer Zusage, beziehungsweise Absage erfolgt im weiteren Verlauf diesen Kapitels. Im Code realisiert ist dies im nachfolgendem Codeverzeichnis.

Codeverzeichnis 10: Veranstaltungsapplikation rechte Spalte 3

```
1            if($favAnzeigen === 'yes')
2            {
3              echo "<div><button type=\"button\" id=\"absagen\">Nicht mehr
                   Teilnehmen</button></div>";
4            }
5            else
6            {
7              if(($events[$index]['Teilnehmer_Max'] > $events[$index]['
                   Teilnehmer_Current'] || $events[$index]['Teilnehmer_Max'] == 0) &&
                    $events[$index]['Aktiv'] == true)
8              {
9                echo "<div><button type=\"button\" id=\"zusagen\">Teilnehmen</button
                     ></div>";
10             }
11             else
12             {
13               echo "<div>Keine Teilnahme moeglich!</<div>";
14             }
15           }
16         echo "</div>";
17       echo "</div>";
18     echo "</div>";
19   echo "</div>";
20   }
```

Mit dem Klick auf den Knopf unterhalb der Detailansicht wird, wie zuvor erwähnt, eine Aktion ausgeführt. Bei dieser Aktion werden mittels einem *href* Variablen gesetzt, jedoch mit Hilfe einer externen JavaScript Datei. Dabei werden für das Zu- und Absagen zwei Funktionen ausgeführt, in denen die Variablen gesetzt werden. Der zugehörige JavaScript sieht folgendermaßen aus:

Codeverzeichnis 11: Veranstaltungsapplikation rechte Spalte 4

```
let zusagen = document.getElementById("zusagen")
let absagen = document.getElementById("absagen")

let urlString = window.location.href
let url = new URL(urlString)
let eventID = url.searchParams.get('id')
let userID = url.searchParams.get('user')
let search = window.location.search

if(zusagen !== null)
{
  zusagen.addEventListener("click", function()
    {
      search = "?fav=yes&user=" + userID + "&zusagen=" + eventID
      window.location.search = search
    }
  )
}

if(absagen !== null)
{
  absagen.addEventListener("click", function()
    {
      search = "?fav=yes&user=" + userID + "&absagen=" + eventID
      window.location.search = search
    }
  )
}
```

Nachfolgend der Darstellung der Detailansicht aus Codeverzeichnis 2 und 3 werden die beiden Variablen aus dem *href* abgefragt. Anhand der Belegung dieser Variablen wird der folgende Code ausgeführt, welcher die Zu- und Absage handhabt.

Codeverzeichnis 12: Veranstaltungsapplikation rechte Spalte 5

```
  if(isset($_GET["absagen"])) {
    abmelden(str_replace(' ', '_', $user->getUID()), $_GET['absagen']);
    header("Refresh:0; url=?fav=yes");
  }

  if(isset($_GET["zusagen"])) {
    anmelden(str_replace(' ', '_', $user->getUID()), $_GET['zusagen']);
    header("Refresh:0; url=?fav=yes");
  }
  ?>
</div>
```

In beiden Fällen wird weitergeleitet zum Reiter *Meine Veranstaltungen*, nachdem die jeweilige Aktion ausgeführt ist.
In Abbildung 9 sind die linke und mittlere Spalte mit den drei Veranstaltungen *Spaziergganggruppe, Yoga und Entspannungsgruppe* zu sehen. Beim Klick auf beispielsweise der Spaziergganggruppe ergibt sich die folgende Detailansicht:
Damit die angezeigten Elemente richtig platziert sind und dem vorgesehenen Design entsprechen, wird CSS verwendet. Die Grundlagen von CSS wurden bereits im Kapitel 2.2 erläutert. Die Elemente sind mittels CSS folgendermaßen platziert und optisch verändert:

Codeverzeichnis 13: Veranstaltungsapplikation CSS Design Desktop Ansicht

```
// HINWEIS! Klassen

.app-content-list .app-content-list-item .app-content-list-item-line-one, .app-
    content-list .app-content-list-item .app-content-list-item-line-two {
  font-size: large;
  text-align: center;
}

.app-content-list {
  min-width: 320px;
}

.kategorien {
```

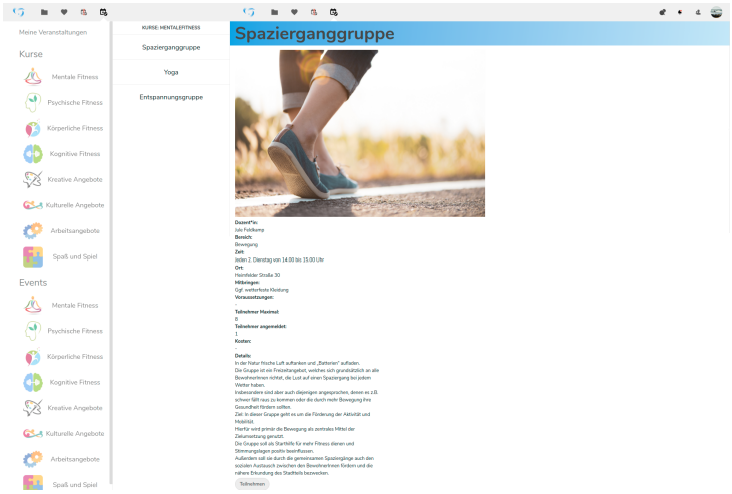


Abbildung 10: 3. Spalte Implementierung Veranstaltungsapp

```
13    margin: auto;
14    font-size: large;
15  }
16
17  .imgDozent {
18    width: 150%;
19    height: auto;
20  }
21
22  .veranstaltunge-img {
23  align-self: baseline;
24    max-width: 50vw;
25  }
26
27  // HINWEIS! Animations
28
29  @keyframes marquee { /* Hier ist "marquee" der Name der Animation */
30    0%   { text-indent: 0% }
31    50% { text-indent: -60% }
32    100% { text-indent: 0% }
33  }
34
35  // HINWEIS! IDs
36
37  #app-navigation > ul > li > a, #app-navigation > ul > li > ul > li > a {
38    font-size: 4vh;
39    min-height: 60px;
40    line-height: 60px;
41    background-size: 20px 20px;
42  }
43
44  #header2 {
45    font-size: 3rem;
46    line-height: initial;
47    margin-left: 1vw;
48  }
49
50  #head2 {
51    text-align: center;
52    margin: auto;
53    padding: 3px;
54    font-size: 2vh;
55  }
56
57  #header1 {
58    background: linear-gradient(98deg, #02a4e4, #c4e8f8);
59    width: 100vw;
60  }
61
62  #details {
63    margin-left: 1vw;
64    max-width: 30vw;
65  }
66
67  #app-navigation {
68    position: absolute;
69  }
```

```
70
71  #app-content #app-content-wrapper {
72    max-width: 10vw;
73  }
74
75  #app-navigation > ul > li > a:first-child img, #app-navigation > ul > li > ul > li
        > a:first-child img {
76    margin-right: 1px;
77    height: auto;
78    margin-left: -38px;
79  }
```

Eine mobile Ansicht sollte auch gegeben sein, sodass beispielsweise auf einem Smartphone die Applikation nutzbar ist. Die Nextcloud stellt von Haus aus mit den verfügbaren Elementen eine mobile Ansicht bereit, weshalb nur kleine Anpassungen vorgenommen werden müssen. Die Änderungen sehen im CSS Code wie folgt aus:

Codeverzeichnis 14: Veranstaltungsapplikation CSS Design mobile Ansicht

```
1   // HINWEIS! Mobile Ansicht
2
3   @media only screen and (max-width: 812px)
4   {
5   // HINWEIS! Klassen
6
7     .app-content-list {
8       min-width: 60vw;
9     }
10
11    .app-content-list .app-content-list-item .app-content-list-item-line-one, .app-
          content-list .app-content-list-item .app-content-list-item-line-two {
12      font-size: 140%;
13      max-width: 55vw;
14      min-width: 20vw;
15      padding: 0px;
16    }
17
18    .marquee {
19      animation: marquee 5s linear 1;
20    }
21
22    .kategorien {
23      font-size: 110%;
24    }
25
26    .veranstaltunge-img {
27      max-width: 95vw;
28    }
29
30  // HINWEIS! IDs
31
32    #details {
33      display: block;
34      max-width: 100vw;
35    }
36
37    #app-navigation-toggle {
38      position: absolute;
39      height: 35px;
40      width: 35px;
41    }
42
43    #app-content-details {
44      min-width: 100vw;
45    }
46
47    #head2 {
48      text-align: right;
49      margin: auto;
50      margin-right: 5px;
51      padding: 6px;
52      font-size: 130%;
53    }
54
55    #app-navigation > ul > li > a, #app-navigation > ul > li > ul > li > a {
56      font-size: 5vh;
57      min-height: 60px;
58      line-height: 60px;
59      background-size: 20px 20px;
```

```
60      }
61
62      #header2 {
63        font-size: x-large;
64      }
65    }
```

Die optischen Anpassungen mittels CSS und die Darstellung der jeweiligen Inhalte sind bereits erläutert worden, jedoch fehlt abschließend die Kommunikation zwischen Applikation und Datenbank. Diese Interaktion ist notwendig, um die darzustellenden Inhalte abzurufen. Wie in Kapitel 2.2 bereits erwähnt, wird eine MySQL Datenbank, sowie die PHP-Erweiterung MySQLi verwendet. Die Funktionen für den Datenbankaustausch befinden sich in einer eigenständigen Datei, die in der Hauptdatei eingebunden ist. Die erste Funktion erstellt eine Tabelle für jeden Benutzer, sofern dieser noch keine hat. In dieser Tabelle werden die zugesagten Veranstaltungen eines Benutzers abgespeichert und wird folgendermaßen realisiert:

Codeverzeichnis 15: Veranstaltungsapplikation Datenbankfunktionen 1

```
1   function createUserTable($userid)
2   {
3     $connection = new mysqli(servername, username, password, database);
4
5     if($connection->connect_error){
6       $connection->close();
7     }
8     else
9     {
10      $sql = "CREATE TABLE IF NOT EXISTS Faehrbook_" . $userid . " (ID int primary
            key not null auto_increment, Events varchar(255) not null)";
11      $result = $connection->query($sql);
12      $connection->close();
13    }
14  }
```

Im Gegensatz dazu, löscht die nächste Funktion, alle Tabellen, die nicht mehr verwendet werden. Eine Tabelle wird obsolet, wenn ein Nutzer nicht mehr im System vorhanden ist. Da das Schema nicht mehr benutzt werden kann, wird Speicher belegt, der nicht mehr verwendet werden kann. Diese Funktion wird bei jedem Start der Applikation aufgerufen und sorgt somit für eine aufgeräumte Datenbank, da der Speicher der unbenutzten Tabellen freigegeben wird. Das Löschen dieser Schemata sieht folgendermaßen aus:

Codeverzeichnis 16: Veranstaltungsapplikation Datenbankfunktionen 2.1

```
1   function deleteUnknownTables()
2   {
3     $connection = new mysqli(servername, username, password, database);
4     $uids = array();
5     $tables = array();
6
7       if($connection->connect_error){
8         $connection->close();
9       }
10      else
11      {
12        $sql = "Select uid from oc_users";
13        $result = $connection->query($sql);
14
15        if($result->num_rows > 0)
16        {
17          while($row = $result->fetch_assoc())
18          {
19            $uids[] = $row;
20          }
21        }
22
23        $sql = "SELECT TABLE_NAME
```

```
24          FROM INFORMATION_SCHEMA.TABLES
25          WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_SCHEMA='nextcloud' AND
                TABLE_NAME LIKE 'Faehrbook%'";
26        $result = $connection->query($sql);
27
28        if($result->num_rows > 0)
29        {
30          while($row = $result->fetch_assoc())
31          {
32            $tables[] = $row;
33          }
34        }
```

Codeverzeichnis 17: Veranstaltungsapplikation Datenbankfunktionen 2.2

```
1         foreach ($tables as $table)
2         {
3           $user_exists = false;
4           foreach ($uids as $uid)
5           {
6             $tmp = "Faehrbook_" . str_replace(' ','_', $uid['uid']);
7
8             if($table['TABLE_NAME'] === $tmp || $table['TABLE_NAME'] === '
                  Faehrbook_Rubriken' || $table['TABLE_NAME'] === '
                  Faehrbook_veranstaltungen')
9             {
10              $user_exists = true;
11            }
12          }
13          if(!$user_exists)
14          {
15            $sql = "DROP TABLE IF EXISTS " . $table['TABLE_NAME'];
16            $result = $connection->query($sql);
17          }
18        }
19        $connection->close();
20      }
21    }
```

Die nächste Funktion, beschafft alle Informationen zu einer Veranstaltung anhand der Art. Wie bereits erwähnt, kann ein Ereignis einmalig, also ein Event, oder regelmäßig, also ein Kurs, sein. Die Funktion sieht im Code folgendermaßen aus:

Codeverzeichnis 18: Veranstaltungsapplikation Datenbankfunktionen 3

```
1   function getAll($art)
2   {
3     $results = array();
4
5     $connection = new mysqli(servername, username, password, database);
6
7     if($connection->connect_error)
8     {
9       $connection->close();
10      return $results;
11    }
12    else
13    {
14      $sql = "SELECT * FROM Faehrbook_veranstaltungen WHERE Art = '$art'";
15      $result = $connection->query($sql);
16
17      if($result->num_rows > 0)
18      {
19        while($row = $result->fetch_assoc())
20        {
21          $results[] = $row;
22        }
23      }
24      $connection->close();
25    }
26    return $results;
27  }
```

Für die korrekte Darstellung der Veranstaltungen innerhalb zugewiesener Kategorien, müssen die verfügbaren Rubriken aus der Datenbank bezogen werden. Das Beziehen dieser Daten geschieht mit Hilfe der nachstehenden Funktion:

Codeverzeichnis 19: Veranstaltungsapplikation Datenbankfunktionen 4.1

```
function getRubriken()
{
  $results = array();

  $connection = new mysqli(servername, username, password, database);

  if($connection->connect_error)
  {
    $connection->close();
    return $results;
  }
```

Codeverzeichnis 20: Veranstaltungsapplikation Datenbankfunktionen 4.2

```
  else
  {
    $sql = "SELECT * FROM Faehrbook_Rubriken";
    $result = $connection->query($sql);

    if($result->num_rows > 0)
    {
      while($row = $result->fetch_assoc())
      {
        $results[] = $row;
      }
    }
    $connection->close();
  }
  return $results;
}
```

Abschließend existieren zwei Funktionen, die eine handhabt Anmeldungen zu Veranstaltungen, die andere befasst sich mit der Abmeldung. Die Funktion zum Anmelden sieht wie folgt aus:

Codeverzeichnis 21: Veranstaltungsapplikation Datenbankfunktionen 5

```
function anmelden($userid, $veranstaltungsid)
{
  $connection = new mysqli(servername, username, password, database);
  $in = 0;

  if($connection->connect_error)
  {
    $connection->close();
  }
  else
  {
    $sql = "SELECT Teilnehmer_Current from Faehrbook_veranstaltungen where
        VeranstaltungsID = '$veranstaltungsid'";
    $result = $connection->query($sql);
    $current = $result->fetch_assoc()['Teilnehmer_Current'];
    $current = $current + 1;

    $sql = "SELECT Teilnehmer from Faehrbook_veranstaltungen where
        VeranstaltungsID = '$veranstaltungsid'";
    $result = $connection->query($sql);
    $teilnehmer = $result->fetch_assoc()['Teilnehmer'];

    $tmp = explode(';', $teilnehmer);

    foreach($tmp as $value)
    {
      if($value === $userid)
      {
        $in++;
      }
    }

    if($in < 1)
    {
      $teilnehmer .= $userid . ';';

      $sql = "Update Faehrbook_veranstaltungen set Teilnehmer = '$teilnehmer'
          where VeranstaltungsID = '$veranstaltungsid'";
      $result = $connection->query($sql);

```

```
38          $sql = "Update Faehrbook_veranstaltungen set Teilnehmer_Current = $current
                where VeranstaltungsID = '$veranstaltungsid'";
39          $result = $connection->query($sql);
40
41          $sql = "INSERT INTO Faehrbook_$userid (Events) values ('$veranstaltungsid')"
                ;
42          $result = $connection->query($sql);
43        }
44        $connection->close();
45      }
46    }
```

Im Gegensatz zur Anmeldung steht die nachfolgende Funktion, die einen Benutzer von einer Veranstaltung abmeldet. Beide Funktionen haben die gleichen Parameter, die zur An-, beziehungsweise Abmelden benötigt werden. Diese Parameter sind zum einen die *userid* und zum anderen die *veranstaltungsid*. Das Abmelden ist in dieser Weise implementiert:

Codeverzeichnis 22: Veranstaltungsapplikation Datenbankfunktionen 6

```
1   function abmelden($userid, $veranstaltungsid)
2   {
3     $connection = new mysqli(servername, username, password, database);
4
5     if($connection->connect_error)
6     {
7       $connection->close();
8     }
9     else
10    {
11      $sql = "SELECT Teilnehmer_Current from Faehrbook_veranstaltungen where
            VeranstaltungsID = '$veranstaltungsid'";
12      $result = $connection->query($sql);
13      $current = $result->fetch_assoc()['Teilnehmer_Current'];
14      $current = $current - 1;
15
16      $sql = "SELECT Teilnehmer from Faehrbook_veranstaltungen where
            VeranstaltungsID = '$veranstaltungsid'";
17      $result = $connection->query($sql);
18      $teilnehmer = $result->fetch_assoc()['Teilnehmer'];
19
20      $tmp = explode(';', $teilnehmer);
21      $new_teilnehmer = '';
22
23      foreach($tmp as $value)
24      {
25        if($value !== $userid)
26        {
27          $new_teilnehmer .= $value . ';';
28        }
29      }
30
31      $sql = "Update Faehrbook_veranstaltungen set Teilnehmer = '$new_teilnehmer'
            where VeranstaltungsID = '$veranstaltungsid'";
32      $result = $connection->query($sql);
33
34      $sql = "Update Faehrbook_veranstaltungen set Teilnehmer_Current = $current
            where VeranstaltungsID = '$veranstaltungsid'";
35      $result = $connection->query($sql);
36
37      $sql = "Delete FROM Faehrbook_$userid where Events = '$veranstaltungsid'";
38      $result = $connection->query($sql);
39
40      $connection->close();
41    }
42  }
```

Mit dieser Funktionalität ist die Implementierung der Veranstaltungsapplikation abgeschlossen.

Im nachfolgendem Abschnitt ist die Umsetzung der administrativen Applikation erläutert, die zur Verwaltung der Veranstaltungen zuständig ist. Dieser Abschnitt beginnt mit der grundlegenden Funktionalität und behandelt im Anschluss den Aufbau der Applikation. Anschließend werden das Routing innerhalb der Anwendung und abschließend die Kom-

munikation zwischen Datenbank und Applikation genauer erläutert. Die administrierende Anwendung ist, im Gegensatz zur Veranstaltungsapplikation, in zwei Spalten eingeteilt. Die erste, beziehungsweise linke Spalte beinhaltet eine Übersicht über alle Funktionalitäten, die zur Verfügung stehen. Die zweite Spalte beinhaltet dementsprechend die Maske der jeweiligen Aktionen, je nach gewählten Reiter. Ebenfalls, wie in der Applikation für die Darstellung der Veranstaltungen, gelangt man über die verschiedenen Reiter mittels *href* zur jeweiligen Detailseite der Funktion. Erweiternd zur Konzeption ergaben sich weitere Anforderungen an diese Applikation.
Die beiden nachfolgenden Funktionen sind zu den eigentlichen hinzugekommen: **Veranstaltungsbil**
Realisiert ist der Aufbau dieser Spalte folgendermaßen:

Codeverzeichnis 23: Admin Applikation linke Spalte 1

```
1  <!-- HINWEIS! linke Spalte fuer die Auswahl der moeglichen Funktionen im
       Adminbereich -->
2  <div id="app-content-wrapper">
3    <div class="app-content-list" style="font-weight: bold; text-align: center; font
         -size: large; min-width: fit-content;">
4      <ul>
5        <a href="?doing=create" class="app-content-list-item">
6          <div class="app-content-list-item-line-one">Veranstaltung erstellen</div>
7        </a>
8
9        <a href="?doing=create-img-v" class="app-content-list-item">
10         <div class="app-content-list-item-line-one">Veranstaltungsbild hochladen</
               div>
11       </a>
12
13       <a href="?doing=update" class="app-content-list-item">
14         <div class="app-content-list-item-line-one">Veranstaltung aktualisieren</
               div>
15       </a>
16
17       <a href="?doing=delete" class="app-content-list-item">
18         <div class="app-content-list-item-line-one">Veranstaltung loeschen</div>
19       </a>
20
21       <a href="?doing=teilnehmer" class="app-content-list-item">
22         <div class="app-content-list-item-line-one">Teilnehmer entfernen</div>
23       </a>
24
25       <a href="?doing=rubriken" class="app-content-list-item">
26         <div class="app-content-list-item-line-one">Rubriken Uebersicht</div>
27       </a>
28
29       <a href="?doing=new-rubriken" class="app-content-list-item">
30         <div class="app-content-list-item-line-one">Neue Rubrik erstellen</div>
31       </a>
32
33       <a href="?doing=create-img-r" class="app-content-list-item">
34         <div class="app-content-list-item-line-one">Rubrikbild hochladen</div>
35       </a>
36
37       <a href="?doing=delete-rubriken" class="app-content-list-item">
38         <div class="app-content-list-item-line-one">Rubrik loeschen</div>
39       </a>
40     </ul>
41   </div>
```

Die Zeilen 2 und 3 spiegeln erneut von der Nextcloud angebotene Elemente wieder. Die jeweiligen Listenelemente sehen in der fertigen Applikation wie folgt aus:
Wie bereits erwähnt, folgt die Navigation der gleichen Idee, wie der Applikation für die Veranstaltungen und realisiert dies mittels $< a >$ und *href*, zu sehen beispielsweise in Zeile 33. Um die Detailseite darzustellen, wird im ersten Schritt die Variable *doing* ausgelesen und gespeichert. Anhand des Wertes werden die jeweiligen Inhalte geladen und angezeigt. Das Auslesen der Variable und das Sichern des Inhalts erfolgen folgendermaßen:

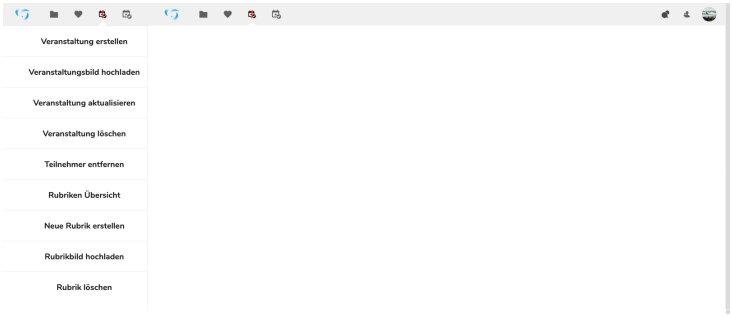


Abbildung 11: 1. Spalte Implementierung Admin Applikation

Codeverzeichnis 24: Admin Applikation rechte Spalte 1

```
<!-- HINWEIS! Handling der Auswahl aus der Navigation -->
<?php
  $doing = $_GET['doing'];
  $titel = '';
  switch ($doing) {
    case 'create':
      $titel = 'Veranstaltung erstellen';
      break;
    case 'update':
      $titel = 'Veranstaltung aktualisieren';
      break;
    case 'delete':
      $titel = 'Veranstaltung loeschen';
      break;
    case 'teilnehmer':
      $titel = 'Teilnehmer aus Veranstaltung entfernen';
      break;
    case 'rubriken':
      $titel = 'Rubriken Uebersicht';
      break;
    case 'new-rubriken':
      $titel = 'Neue Rubrik erstellen';
      break;
    case 'delete-rubriken':
      $titel = 'Rubrik loeschen';
      break;
    case 'create-img-v':
      $titel = 'Bild hochladen fuer eine Veranstaltung';
      break;
    case 'create-img-r':
      $titel = 'Bild hochladen fuer eine Rubrik';
      break;
  }
?>
```

Im Anschluss dessen ist der Titel der Detailseite gesetzt und wird wie folgt dargestellt:

Codeverzeichnis 25: Admin Applikation rechte Spalte 2

```
<!-- HINWEIS! Darstellung rechte Spalte anhand der ausgewaehlten Funktion -->
<div id="app-content-details">
  <header>
    <h2 style="margin-left: 10px; margin-top: 10px; width: 100vw; font-size: xx-
        large;">
      <?php echo $titel; ?>
    </h2>
  </header>
```

Im nächsten Schritt beginnt die Darstellung der Details, jedoch wird zuvor der reguläre Ausdruck erstellt, der für die Überprüfung von im Verlauf auftretenden Eingaben verwendet wird. Mit dem nachstehenden Code wird dieser realisiert:

Codeverzeichnis 26: Admin Applikation rechte Spalte 3

```
  <div id="details">
    <?php
      $pattern_rubriken = getRubriken();
      $pattern = '^(';
```

```
5
6         foreach ($pattern_rubriken as $value)
7         {
8           $pattern .= $value['Titel'] . '|';
9         }
10
11        $pattern = substr($pattern, 0, -1);
12        $pattern .= '$)';
```

Um die Wahl für die richtige Darstellung zu realisieren, wird eine *switch* Anweisung verwendet. Wie die zuvor verwendete switch Anweisung, agiert diese anhand des Inhalts der *doing* Variable. Damit lassen sich die verschiedenen Fälle von *doing* unterscheiden die jeweiligen case Bedingungen sind die gleichen, wie in Codeverzeichnis 24, allerdings wird deutlich mehr Code ausgeführt. Der erste Fall ist das *create*, welcher für das Erstellen einer Veranstaltung zuständig ist. Im folgenden Code ist dies implementiert:

Codeverzeichnis 27: Admin Applikation rechte Spalte 4.1 create

```
1  switch ($doing) {
2    case 'create': // HINWEIS! Darstellung fuer die Funktion, eine neue
         Veranstaltung zu erstellen
3      ?>
4      <div style="margin-left: auto; margin-right: auto; margin-top: 1rem; text-
           align: left; display: flex;">
5        <form action="" method="get">
6          <label for="art">Art (event -> einmalig, kurs -> mehrmalig):</label>
7          <input required type="text" id="art" name="art" pattern="^(event|kurs)$">
8
9          <label for="rubrik">Rubrik (z.B. Mentale Fitness):</label>
10         <?php
11           echo "<input required type=\"text\" id=\"rubrik\" name=\"rubrik\"
                 pattern=\"$pattern\">";
12         ?>
13
14         <label for="titel">Titel:</label>
15         <textarea class='autoExpand' rows='3' data-min-rows='3' required type="
               text" id="titel" name="titel"></textarea>
16
17         <label for="bereich">Bereich:</label>
18         <textarea class='autoExpand' rows='3' data-min-rows='3' type="text" id="
               bereich" name="bereich"></textarea>
19
20         <label for="dozent">Dozent*in:</label>
21         <textarea class='autoExpand' rows='3' data-min-rows='3' type="text" id="
               dozent" name="dozent"></textarea>
22
23         <label for="zeit">Zeit:</label>
24         <textarea class='autoExpand' rows='3' data-min-rows='3' type="text" id="
               zeit" name="zeit"></textarea>
25
26         <label for="ort">Ort:</label>
27         <textarea class='autoExpand' rows='3' data-min-rows='3' type="text" id="
               ort" name="ort"></textarea>
28
29         <label for="mitbringen">Mitbringen:</label>
30         <textarea class='autoExpand' rows='3' data-min-rows='3' type="text" id="
               mitbringen" name="mitbringen"></textarea>
31
32         <label for="voraussetzungen">Voraussetzungen:</label>
33         <textarea class='autoExpand' rows='3' data-min-rows='3' type="text" id="
               voraussetzungen" name="voraussetzungen"></textarea>
34
35         <label for="teilnehmer-max">Teilnehmer Maximal(0 = keine
               Teilnehmerbegrenzung):</label>
36         <input type="number" id="teilnehmer-max" name="teilnehmer-max" min="0">
37
38         <label for="kosten">Kosten:</label>
39         <textarea class='autoExpand' rows='3' data-min-rows='3' type="text" id="
               kosten" name="kosten"></textarea>
40
41         <label for="details">Details:</label>
42         <textarea class="autoExpand" rows='3' data-min-rows='3' type="text" id="
               details" name="details"></textarea>
43
44         <label for="active">Aktiv(0 = inaktiv/pausiert | 1 = aktiv):</label>
45         <input type="text" id="active" name="active" pattern="[0-1]">
46
```

```
47          <input type="hidden" name="doing" value="create">
48
49          <label for="submit"></label>
50          <input type="submit" name="submit" value="sichern">
51        </form>
52        <div style="margin-left: 50px;">
53        <div style="font-size: x-large; font-weight: bold;">Rubriken:</div><br>
54        <?php
55          foreach ($pattern_rubriken as $value)
56          {
57            echo "<div style=\"font-size: large;\">" . $value['Titel'] . "</div><br>
                  ";
58          }
59        ?>
60        </div>
61        </div>
```

Der oben stehende Code repräsentiert eine *form*, in der verschiedene Eingabefelder vorhanden sind, die zum Erstellen von Veranstaltungen genutzt werden. Diese Felder werden vom Benutzer gefüllt und mit Hilfe des Knopfes wird pro Feld eine Variable mit dem Inhalt des Feldes gefüllt. Diese Variablen werden, wie beim *href*, in die URL gesetzt. Die Inhalte werden mit dem nachfolgenden Code aus der URL bezogen und in die Datenbank geschrieben. Nach dem Einpflegen der Daten, wird die Detailseite neu geladen, jedoch ausschließlich mit der gesetzten Variable *doing*.

Codeverzeichnis 28: Admin Applikation rechte Spalte 4.2 create

```
1         <?php
2           if(isset($_GET['submit']))
3           {
4             $data = array(
5               'Art'         => $_GET['art'],
6               'Rubrik'      => $_GET['rubrik'],
7               'Titel'       => $_GET['titel'],
8               'Bereich'     => $_GET['bereich'],
9               'Dozent'      => $_GET['dozent'],
10              'Zeit'        => $_GET['zeit'],
11              'Ort'         => $_GET['ort'],
12              'Mitbringen'     => $_GET['mitbringen'],
13              'Voraussetzungen' => $_GET['voraussetzungen'],
14              'Kosten'      => $_GET['kosten'],
15              'Details'     => $_GET['details'],
16              'Teilnehmer_Max'  => intval($_GET['teilnehmer-max']),
17              'Aktiv'       => intval($_GET['active'])
18            );
19
20            newVeranstaltung($data);
21            header("Refresh:0; url=?doing=" . $doing);
22          }
23        break;
```

Im weiteren Verlauf des Abschnitts wird zunächst die Umsetzung der anderen Funktionen in der Admin Applikation genauer erläutert und anschließend drei beispielhafte Bilder zur optischen Untermalung dargestellt. Die drei Funktionalitäten sind das Erstellen einer Veranstaltung, das Hochladen eines Veranstaltungsbildes, das Aktualisieren einer Veranstaltung und das Löschen einer Veranstaltung. Im Grunde sehen die Detailseiten dieser Aktionen im Hinblick auf Rubriken gleich aus, jedoch mit weniger Eingabefeldern. Die nächste Funktionalität ist das Hochladen eines Bildes für eine Veranstaltung. Diese Aktion hat sich im Laufe der Implementierung als sinnvoll erachtet. Die Umsetzung des Hochladens der Bilder für Rubriken, sowie Veranstaltungen wird im Anschluss an die Erläuterung des Aufbaus, der jeweiligen Detailseiten, genauer betrachtet. Im Gegensatz dazu ist die Darstellung dieser Funktion mittels folgendem Code realisiert:

Codeverzeichnis 29: Admin Applikation rechte Spalte 5 create-img-v

```
case 'create-img-v': // HINWEIS! Darstellung fuer die Funktion, ein Bild fuer eine Veranstaltung hochzuladen
  ?>
  <div class="container">
    <form method="post" action="/veranstaltungen" enctype="multipart/form-data" id="myform">
      <label for="file">Bild hochladen</label>
      <input type="file" id="file" name="file" required/>
      <label for="rubrik">Rubrik (z.B. Mentale Fitness):</label>
      <?php
        echo "<input required type=\"text\" id=\"rubrik_img\" name=\"rubrik\" pattern=\"$pattern\">";
      ?>
      <label for="titel">Titel:</label>
      <input required type="text" id="titel_img" name="titel">
      <input type="button" class="button" value="Upload" id="but_upload-v">
    </form>
    <span>
      Es koennen auschliesslich JPEG und JPG Dateien hochgeladen werden! <br>
        Bestehende Bilder fuer eine Kombination aus Rubrik und Titel werden bei erneutem hochladen ueberschrieben!
    </span>
  </div>
  <?php
  break;
```

Der nächste Reiter in der linken Spalte ist das Aktualisieren einer Veranstaltung. Dort werden die bisherigen Einträge zu einer Veranstaltung angezeigt und können mittels Eingabefelder aktualisiert werden. Es gibt keine Pflichtfelder und aktualisiert werden nur die Daten von gefüllten Feldern. Jedoch kann nur eine bestehende Rubrik, sowie für die Art nur *event* oder *kurs* eingetragen werden. Im ersten Teil des Aufbaus werden alle Veranstaltungen aus der Datenbank bezogen und als Dropdown-Menü dargestellt. Wird eine Veranstaltung ausgewählt, wird dessen ID in eine Variable gespeichert und der URL übergeben. Im Anschluss daran, wird diese ID ausgelesen und anhand dieser die einzelnen Felder mit der aktuellen Belegung ausgegeben. Dies ist folgendermaßen umgesetzt:

Codeverzeichnis 30: Admin Applikation rechte Spalte 6.1 update

```
case 'update': // HINWEIS! Darstellung fuer die Funktion, eine bestehende Veranstaltung zu aktualisieren
  ?>
  <div style="margin-left: auto; margin-right: auto; margin-top: 1rem; text-align: left;">
  <form action="" method="get">
    <select required id="select-veranstaltungen">
      <option disabled selected value>Veranstaltung auswaehlen</option>
      <?php
        $veranstaltungen = getVeranstaltungen();

        foreach ($veranstaltungen as $value)
        {
          echo "<option value=\"". $value['VeranstaltungsID'] . "\">" . $value['Titel'] . "</option>";
        }
      ?>
    </select>
    <input type="hidden" name="doing" value="update">
```

Wie bereits erwähnt, wird im nächsten Schritt die ID der gewählten Veranstaltung bezogen und anhand dieser die Eingabefelder, sowie Belegungen dargestellt. Dies passiert mit diesem Code:

Codeverzeichnis 31: Admin Applikation rechte Spalte 6.2 update

```
  <?php
    echo "<input type=\"hidden\" name=\"v_id\" value=\"" . $_GET['id'] . "\">";

    foreach ($veranstaltungen as $value)
    {
      if($value['VeranstaltungsID'] == $_GET['id'])
```

```
7         {
8           echo "<h2>" . $value['Titel'] . "</h2>";
9
10          echo "<label for=\"art\">Art (event -> einmalig, kurs -> mehrmalig):</
                label>";
11          echo "<input type=\"text\" id=\"art\" name=\"art\" pattern=\"^(event|kurs)
                $\">";
12          echo "<br><div style=\"max-width: 80vw; margin-left: 5px; color: gray\">"
                . $value['Art'] ."</div><hr>";
13
14          echo "<label for=\"rubrik\">Rubrik (z.B. Mentale Fitness):</label>";
15          echo "<input type=\"text\" id=\"rubrik\" name=\"rubrik\" pattern=\"^(
                Mentale Fitness|Psychische Fitness|Koerperliche Fitness|Kognitive
                Fitness|Kreative Angebote|Kulturelle Angebote|Arbeitsangebote|Spass
                und Spiel)$\">";
16          echo "<br><div style=\"max-width: 80vw; margin-left: 5px; color: gray\">"
                . $value['Rubrik'] ."</div><hr>";
17
18          echo "<label for=\"titel\">Titel:</label>";
19          echo "<textarea class='autoExpand' rows='3' data-min-rows='3' type=\"text
                \" id=\"titel\" name=\"titel\"></textarea>";
20          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Titel'] ."</div
                ><hr>";
21
22          echo "<label for=\"bereich\">Bereich:</label>";
23          echo "<textarea class='autoExpand' rows='3' data-min-rows='3' type=\"text
                \" id=\"bereich\" name=\"bereich\"></textarea>";
24          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Bereich'] ."</
                div><hr>";
25
26          echo "<label for=\"dozent\">Dozent*in:</label>";
27          echo "<textarea class='autoExpand' rows='3' data-min-rows='3' type=\"text
                \" id=\"dozent\" name=\"dozent\"></textarea>";
28          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Dozent'] ."</
                div><hr>";
29
30          echo "<label for=\"zeit\">Zeit:</label>";
31          echo "<textarea class='autoExpand' rows='3' data-min-rows='3' type=\"text
                \" id=\"zeit\" name=\"zeit\"></textarea>";
32          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Zeit'] ."</div
                ><hr>";
33
34          echo "<label for=\"ort\">Ort:</label>";
35          echo "<textarea class='autoExpand' rows='3' data-min-rows='3' type=\"text
                \" id=\"ort\" name=\"ort\"></textarea>";
36          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Ort'] ."</div><
                hr>";
37
38          echo "<label for=\"mitbringen\">Mitbringen:</label>";
39          echo "<textarea class='autoExpand' rows='3' data-min-rows='3' type=\"text
                \" id=\"mitbringen\" name=\"mitbringen\"></textarea>";
40          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Mitbringen'] ."
                </div><hr>";
41
42          echo "<label for=\"voraussetzungen\">Voraussetzungen:</label>";
43          echo "<textarea class='autoExpand' rows='3' data-min-rows='3' type=\"text
                \" id=\"voraussetzungen\" name=\"voraussetzungen\"></textarea>";
44          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Voraussetzungen
                '] ."</div><hr>";
45
46          echo "<label for=\"teilnehmer-max\">Teilnehmer Maximal(0 = keine
                Teilnehmerbegrenzung):</label>";
47          echo "<input type=\"number\" id=\"teilnehmer-max\" name=\"teilnehmer-max\"
                 min=\"0\">";
48          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Teilnehmer_Max'
                ] ."</div><hr>";
49
50          echo "<label for=\"kosten\">Kosten:</label>";
51          echo "<textarea class='autoExpand' rows='3' data-min-rows='3' type=\"text
                \" id=\"kosten\" name=\"kosten\"></textarea>";
52          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Kosten'] ."</
                div><hr>";
53
54          echo "<label for=\"details\">Details:</label>";
55          echo "<textarea class=\"autoExpand\" rows='3' data-min-rows='3' type=\"
                text\" id=\"details\" name=\"details\"></textarea>";
56          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Details'] ."</
                div><hr>";
57
58          echo "<label for=\"active\">Aktiv(0 = inaktiv/pausiert | 1 = aktiv):</
                label>";
59          echo "<input type=\"text\" id=\"active\" name=\"active\" pattern
                =\"[0-1]\">";
```

```
60          echo "<br><div class=\"veranstaltung-aktuell\">" . $value['Aktiv'] ."</div
                ><hr>";
61
62          echo "<br><input type=\"submit\" name=\"submit\" value=\"updaten\">";
63          break;
64        }
65      }
66    ?>
67  </form>
68  </div>
```

Mit dem Drücken des Knopfes werden die Inhalte der Felder in Variablen gesichert. Diese werden im letzten Schritt dieser Funktion ausgelesen und an eine Funktion übergeben, die jene Veranstaltung mit den neuen Werten aktualisiert. Wurden die Daten aktualisiert, wird darauf diese Detailseite neu geladen, jedoch nur mit der auf *update* gesetzten Variable *doing*. Diese Schritte sind wie folgt implementiert:

Codeverzeichnis 32: Admin Applikation rechte Spalte 6.3.1 update

```
1   <?php
2     if(isset($_GET['submit']))
3     {
4       $data = array();
5
6       $inputs = array(
7         'art'          => 'Art',
8         'rubrik'       => 'Rubrik',
9         'titel'        => 'Titel',
10        'bereich'        => 'Bereich',
11        'dozent'       => 'Dozent',
12        'zeit'         => 'Zeit',
13        'ort'          => 'Ort',
14        'mitbringen'     => 'Mitbringen',
15        'voraussetzungen' => 'Voraussetzungen',
16        'teilnehmer-max'  => 'Teilnehmer_max',
17        'kosten'       => 'Kosten',
18        'details'        => 'Details',
19        'active'       => 'Aktiv'
20      );
```

Codeverzeichnis 33: Admin Applikation rechte Spalte 6.3.2 update

```
1       foreach ($inputs as $key => $value)
2       {
3         $tmp = $_GET[$key];
4         if(strlen($tmp) > 0)
5         {
6           if($key != "teilnehmer-max" && $key != "active") { $data[$value] = $_GET[
                $key]; }
7           else { $data[$value] = intval($_GET[$key]); }
8         }
9       }
10
11      if(!empty($data))
12      {
13        updateVeranstaltung($_GET['v_id'], $data);
14        header("Refresh:0; url=?doing=" . $doing . "&id=0&index=0");
15      }
16    }
17  break;
```

Die Funktionalitäten hinter den einzelnen Funktionen, wie beispielsweise das Aktualisieren einer Veranstaltung, werden im weiteren Verlauf genauer erläutert. Soll eine Veranstaltung entfernt werden, da diese obsolet geworden ist, geschieht dies über den Reiter *Veranstaltung löschen*. Die dazugehörige Detailseite ist, wie nachgehend zu sehen, aufgebaut:

Codeverzeichnis 34: Admin Applikation rechte Spalte 7 delete

```
1   case 'delete': // HINWEIS! Darstellung fuer die Funktion, eine bestehende
        Veranstaltung zu loeschen
2     ?>
```

```
3     <div style="margin-left: auto; margin-right: auto; margin-top: 1rem; text-align:
          left;">
4     <form action="" method="get">
5       <?php
6         $veranstaltungen = getVeranstaltungen();
7
8         foreach ($veranstaltungen as $value)
9         {
10          echo "<input type=\"radio\" name=\"veranstaltung\" value=\"". $value['
                VeranstaltungsID'] . ";" . $value['Rubrik'] . ";" . $value['Titel'] .
                "\" style=\"vertical-align: middle;margin-right: 10px;\">" . $value['
                Art'] . "-" . $value['Rubrik'] . "-" . $value['Titel'] . "<br>";
11        }
12      ?>
13      <input type="hidden" name="doing" value="delete">
14      <input type="submit" name="submit" value="loeschen">
15    </form>
16    </div>
17    <?php
18      if(isset($_GET['submit']))
19      {
20        deleteVeranstaltung($_GET['veranstaltung']);
21        header("Refresh:0; url=?doing=" . $doing);
22      }
23    break;
```

Umgesetzt ist die Auswahl der zu löschenden Veranstaltung mittels Radio-Button. Von den Knöpfen kann nur einer auf einmal gewählt sein, sodass ein versehentliches Löschen nicht passieren kann. Nachdem eine Veranstaltung gelöscht wurde, wird, wie bei den anderen Funktionen auch, die Detailseite neu geladen.
Der gegensätzliche Fall tritt ein, wenn ein Teilnehmer nicht mehr zu einer Veranstaltung geht, kann ein Dozent diesen aus der Veranstaltung entfernen. Damit ist sichergestellt, dass der Platz für einen interessierten Klienten frei wird und jeder die Möglichkeit hat, an den angebotenen Veranstaltungen teilzunehmen. Es kann ausschließlich ein Teilnehmer auf einmal entfernt werden. Ebenfalls realisiert ist die Auswahl mit Hilfe eines Dropdown-Menüs, in dem alle Teilnehmer den jeweiligen Veranstaltungen zugeordnet sind. Gibt es keine Veranstaltung mit Teilnehmern, dann ist das Menü leer. Der Knopf kann nur dann gedrückt werden, beziehungsweise führt das Entfernen aus, wenn im Menü eine Auswahl vorgenommen ist. Der Teilnehmer kann folgendermaßen entfernt werden:

Codeverzeichnis 35: Admin Applikation rechte Spalte 8 teilnehmer

```
1   case 'teilnehmer': // HINWEIS! Darstellung fuer die Funktion, Teilnehmer aus einer
        bestehenden Veranstaltung zu entfernen
2     ?>
3     <div style="margin-left: auto; margin-right: auto; margin-top: 1rem; text-align:
          left;">
4     <form action="" method="get">
5       <select required id="select-teilnehmer-delete">
6         <option disabled selected value>Teilnehmer auswaehlen</option>
7         <?php
8           $veranstaltungen = getVeranstaltungen();
9
10          foreach ($veranstaltungen as $value)
11          {
12            if(strlen($value['Teilnehmer']) > 0)
13            {
14              $tmp_teilnehmer = getTeilnehmer($value['VeranstaltungsID'])[0]['
                    Teilnehmer'];
15
16              $teilnehmer = explode(';', $tmp_teilnehmer);
17
18              foreach ($teilnehmer as $temp)
19              {
20                if(strlen($temp) > 0)
21                {
22                  echo "<option value=\"". $temp . ";" . $value['VeranstaltungsID']
                        . "\">" . $value['Titel'] . ": " . $temp . "</option>"; } }
23            }
```

```
24          }
25        ?>
26      </select>
27      <input type="hidden" name="doing" value="teilnehmer">
28      <input type="hidden" name="teilnehmer" id="select-teilnehmer-teilnehmer">
29      <input type="hidden" name="id" id="select-teilnehmer-id">
30      <input type="submit" name="submit" value="loeschen">
31    </form>
32    </div>
33    <?php
34      if(isset($_GET['submit']))
35      {
36        deleteTeilnehmer($_GET['teilnehmer'], $_GET['id']);
37        header("Refresh:0; url=?doing=" . $doing);
38      }
39    break;
```

Die nächsten und zugleich letzten Reiter der linken Spalte dienen zur Verwaltung der Rubriken, welche die Veranstaltungen einteilen. Der erste Reiter dient lediglich zur Darstellung der Rubriken, die in der Datenbank vorhanden sind. Dies geschieht wie folgt:

Codeverzeichnis 36: Admin Applikation rechte Spalte 9 rubriken

```
1  case 'rubriken': // HINWEIS! Darstellung einer Uebersicht ueber alle Rubriken, die
       verfuegbar sind
2    ?>
3
4    <div style="margin-left: 50px;">
5
6    <?php
7      $rubriken = getRubriken();
8
9      foreach ($rubriken as $value)
10     {
11       echo "<h3>" . $value['Titel'] . "</h3><br>";
12     }
13   ?>
14   </div>
15   <?php
16   break;
```

Im Laufe der Zeit können neue Rubriken hinzu kommen, die ebenfalls einfach einzupflegen sein sollen. Dies kann mit Hilfe der nächsten Funktion ausgeführt werden. Zum Anlegen werden lediglich zwei Elemente benötigt. Zum einem ein Eingabefeld, in dem der Titel der neuen Rubrik eingetragen werden kann, sowie zum anderen einen Knopf, um diese an die Datenbank zu senden und zu speichern.
Der zugehörige Code ist der nachstehende:

Codeverzeichnis 37: Admin Applikation rechte Spalte 10 new-rubrik

```
1  case 'new-rubriken': // HINWEIS! Darstellung fuer die Funktion, eine neue Rubrik
       anzulegen
2    ?>
3    <div style="margin-left: auto; margin-right: auto; margin-top: 1rem; text-align:
         left; display: flex;">
4      <form action="" method="get">
5        <label for="titel" style="font-size: x-large;">Titel:</label>
6        <textarea class='autoExpand' rows='3' data-min-rows='3' required type="text"
             id="titel" name="titel"></textarea>
7
8        <input type="hidden" name="doing" value="new-rubriken">
9        <label for="submit"></label>
10       <input type="submit" name="submit" value="sichern">
11     </form>
12   </div>
13   <?php
14     if(isset($_GET['submit']))
15     {
16       newRubrik($_GET['titel']);
17       header("Refresh:0; url=?doing=" . $doing);
18     }
19   break;
```

Damit die neu erstellte Rubrik vernünftig in der Veranstaltungsapplikation dargestellt werden kann, wird ein Logo benötigt. Des Weiteren könnte es vorkommen, dass ein bestehendes Logo ausgetauscht werden soll. Um diesen beiden Funktionalitäten gerecht zu werden, ist die nachfolgende Funktion implementiert:

Codeverzeichnis 38: Admin Applikation rechte Spalte 11 create-img-r

```
1  case 'create-img-r': // HINWEIS!
2    ?>
3    <div class="container">
4      <form method="post" action="/rubriken" enctype="multipart/form-data" id="
           myform">
5        <label for="file">Bild hochladen</label>
6        <input type="file" id="file" name="file" required/>
7        <label for="rubrik">Rubrik (z.B. Mentale Fitness):</label>
8        <?php
9          echo "<input required type=\"text\" id=\"rubrik_img\" name=\"rubrik\"
               pattern=\"$pattern\">";
10       ?>
11       <input type="button" class="button" value="Upload" id="but_upload-r">
12     </form>
13     <span>Es koennen auschliesslich PNG Dateien hochgeladen werden! <br>
           Bestehende Bilder fuer eine Rubrik werden bei erneutem hochladen
           ueberschrieben!</span>
14   </div>
15   <?php
16   break;
```

Wichtig beim Hochladen der neuen Bilder ist, dass nur welche im Format PNG akzeptiert werden. Dies hat den Grund, dass Logos ohne Hintergrund dargestellt werden können. Ebenfalls wird damit die Auswahl des richtigen Bildes vereinfacht.
Abschließend ist der letzte Reiter der Spalte für das Löschen einer Rubrik zuständig, falls eine von den bestehenden obsolet geworden ist. Das Löschen ist nachstehend umgesetzt:

Codeverzeichnis 39: Admin Applikation rechte Spalte 12.1 delete-rubriken

```
1    case 'delete-rubriken': // HINWEIS! Darstellung fuer die Funktion, eine
         bestehende Rubrik zu loeschen
2      ?>
3      <div style="margin-left: auto; margin-right: auto; margin-top: 1rem; text-
           align: left;">
4        <form action="" method="get">
5        <?php
6          $rubriken = getRubriken();
7          foreach ($rubriken as $value) { echo "<input type=\"radio\" name=\"
               rubriken-del\" value=\"". $value['Titel'] . "\" style=\"vertical-align
               : middle;margin-right: 10px;\">" . $value['Titel'] . "<br>"; }
8        ?>
9          <input type="hidden" name="doing" value="delete-rubriken">
10         <input type="submit" name="submit" value="loeschen">
11       </form>
12     </div>
```

Codeverzeichnis 40: Admin Applikation rechte Spalte 12.2 delete-rubriken

```
1      <?php
2        if(isset($_GET['submit']))
3        {
4          deleteRubrik($_GET['rubriken-del']);
5          header("Refresh:0; url=?doing=" . $doing);
6        }
7      break;
8    default:
9      break;
10       }
11     ?>
12     </div>
13   </div>
14 </div>
```

Mit dem Löschen von Rubriken ist der Abschnitt über die Darstellung der jeweiligen Detailseiten abgeschlossen. Es folgen die nächsten beiden Abschnitte, die sich erstens mit

dem Routing und die Funktionalität hinter dem Hochladen der Bilder, sowie zweitens mit der Kommunikation zwischen Applikation und Datenbank befassen.
Damit das Hochladen der Bilder mittels einem *POST* funktioniert, müssen zwei neue Routen für jeweils Veranstaltungs- und Rubrikenbilder angelegt werden. Diese Routen sind folgendermaßen in der vorgesehenen *routes.php* Datei, die von der Nextcloud für Routing vorgegeben ist, umgesetzt:

Codeverzeichnis 41: Admin Applikation routes.php

```
1  <?php
2    return [
3      'routes' => [
4        ['name' => 'page#index', 'url' => '/', 'verb' => 'GET'],
5        ['name' => 'post#veranstaltungen', 'url' => '/veranstaltungen', 'verb' => '
             POST'],
6        ['name' => 'post#rubriken', 'url' => '/rubriken', 'verb' => 'POST'],
7      ]
8    ];
```

Es wird eine Funktion eines Controllers aufgerufen anhand einer URL, wie beispielsweise in Zeile 5 *post#veranstaltungen* zu sehen ist. Es wird vom *PostController* die Funktion *veranstaltungen* aufgerufen. Demnach ist, beziehungsweise sollte ein zugehöriger Controller implementiert werden, damit das Routing wie gewünscht funktioniert. Der *PostController* ist wie folgt aufgebaut:

Codeverzeichnis 42: Admin Applikation PostController

```
1  <?php
2    namespace OCA\FaehrbookEventsAdmin\Controller;
3
4    use OCP\IRequest;
5    use OCP\AppFramework\Controller;
6
7    class PostController extends Controller {
8      private $userId;
9
10     public function __construct($AppName, IRequest $request, $UserId){
11       parent::__construct($AppName, $request);
12       $this->userId = $UserId;
13     }
14
15     public function veranstaltungen() {
16       require_once("apps/faehrbookeventsAdmin/templates/content/upload.php");
17       return upload_v();
18     }
19
20     public function rubriken() {
21       require_once("apps/faehrbookeventsAdmin/templates/content/upload.php");
22       return upload_r();
23     }
24   }
```

Der *PostController* ruft jeweils eine bestimmte Funktion in der Datei *upload.php* auf. Diese liefert einen Rückgabewert zurück, welcher ebenfalls vom Controller weitergereicht wird. Dieser Wert wird von der zugehörigen JavaScript Datei empfangen, die den POST ausgeführt hat. Wie genau die JavaScript Datei aufgebaut ist und wie diese den POST handhabt folgt im weiteren Verlauf. Zunächst stehen die beiden Funktionen *upload_v* und *upload_r* im Fokus. Grundlegend sind die beiden Funktionen weitestgehend identisch, jedoch sind unterschiedliche Bildformate notwendig und die Pfade zum Abspeichern der Bilder ist ebenfalls unterschiedlich. Das Hochladen von Bilder für Veranstaltungen passiert mit dieser Funktion:

Codeverzeichnis 43: Admin Applikation Upload Veranstaltung

```
<?php
function upload_v()
{
  $rubrik = $_POST['rubrik'];
  $titel = $_POST['titel'];
  $verzeichnis = $rubrik . '_' . $titel . '.jpeg';
  $filename = $_FILES['file']['name'];
  $location = "apps/faehrbookevents/img/veranstaltungen/" . $filename;
  $uploadOk = true;
  $imageFileType = pathinfo($location,PATHINFO_EXTENSION);

  $valid_extensions = array("jpeg", "jpg");

  if( !in_array(strtolower($imageFileType),$valid_extensions) ) {
    $uploadOk = false;
  }

  if($uploadOk === false)
  {
    return 0;
  }
  else
  {
    if(move_uploaded_file($_FILES['file']['tmp_name'],$location))
    {
      rename($location, "apps/faehrbookevents/img/veranstaltungen/" . $verzeichnis
          );
      return "/apps/faehrbookevents/img/veranstaltungen/" . $verzeichnis;
    }
    else
    {
      return 0;
    }
  }
}
```

Bilder für Veranstaltungen können nur im Format *JPG* oder *JPEG* hochgeladen werden. Dies wird ebenfalls in der Funktion geprüft und eine Fehlermeldung wird ausgegeben, wenn ein anderes Bildformat gewählt ist. Gleiches gilt für Rubrikbilder, die jedoch ausschließlich im *PNG* Format akzeptiert werden.

Codeverzeichnis 44: Admin Applikation Upload Rubrik 1

```
function upload_r()
{
  $rubrik = $_POST['rubrik'];
  $verzeichnis = $rubrik . '.png';
  $filename = $_FILES['file']['name'];
  $location = "apps/faehrbookevents/img/rubriken/" . $filename;
  $uploadOk = true;
  $imageFileType = pathinfo($location,PATHINFO_EXTENSION);

  $valid_extensions = array("png");

  if(!in_array(strtolower($imageFileType),$valid_extensions))
  {
    $uploadOk = false;
  }
```

Codeverzeichnis 45: Admin Applikation Upload Rubrik 2

```
  if($uploadOk == false)
  {
    return 0;
  }
  else
  {
    if(move_uploaded_file($_FILES['file']['tmp_name'],$location))
    {
      rename($location, "apps/faehrbookevents/img/rubriken/" . $verzeichnis);
      return "/apps/faehrbookevents/img/rubriken/" . $verzeichnis;
    }
    else
    {
      return 0;
```

```
15        }
16      }
17    }
```

Die vorherigen Codeverzeichnisse 44 und 45 stellen die Implementierung des Hochladens für Rubrikbilder dar. Ebenfalls wird das Bildformat geprüft und gegebenenfalls eine Fehlermeldung ausgegeben. Beide Funktionen geben eine Meldung aus, wenn das Bild erfolgreich hochgeladen wurde. Das Hochladen wird angestoßen, wenn der jeweilige Knopf gedrückt wird. Die Funktionalität wird dem Knopf durch den Einsatz von JavaScript verliehen. Die Nextcloud bietet von Haus aus den Einsatz der JavaScript Erweiterung JQuery an, die asynchrone Anfragen handhabt. Mit JQuery kann ein POST abgefangen und bestimmte Aktionen anhand dessen ausgeführt werden. Wie in den letzten beiden Funktionen zu sehen, wird mittels einem Knopf ein POST Request ausgelöst, dass heißt die Bilddaten, sowie die Inhalte der Textfelder werden an die jeweiligen Routen geschickt. Die jeweiligen Routen sind für Veranstaltungen */veranstaltungen* und für Rubriken */rubriken*. Das Versenden des Requests übernimmt der folgende JavaScript Code mit dem Einsatz von JQuery. Zunächst wird für beide Funktionen eine *baseUrl* erstellt, die dazu dient, dass die Nextcloud den POST Request an die richtige Stelle leitet. Anschließend wird der gesamten Applikation eine Funktion zugeteilt, die jeweils die beiden einzelnen Funktionen den jeweiligen Elementen für das Hochladen zuordnet. Die Funktion wird ausgelöst, wenn der Knopf gedrückt wird und bezieht die Daten des Bildes und der Textfelder. Diese Daten werden dem POST mitgegeben. Je nach Antwort auf diesen Request, ist das Bild erfolgreich hochgeladen oder nicht. Anhand dieser Antwort wird ebenfalls eine Meldung ausgegeben, die den Erfolg des Requests wiedergibt. Nach erfolgreichem Hochladen, wird die Seite neu geladen.

Codeverzeichnis 46: Admin Applikation Upload Veranstaltung Post Request

```
1  var baseUrl = OC.generateUrl('/apps/faehrbookeventsAdmin')
2  $(document).ready(function(){
3  // Veranstaltungsbild
4    $("#but_upload-v").click(function(){
5      var fd = new FormData()
6      var files = $('#file')[0].files[0]
7      var rubrik = $('#rubrik_img').val()
8      var titel = $('#titel_img').val()
9
10     fd.append('file',files)
11     fd.append('rubrik',rubrik)
12     fd.append('titel',titel)
13
14     $.ajax({
15       url: baseUrl + '/veranstaltungen',
16       type: 'post',
17       data: fd,
18       success: function(response){
19         if(response != 0){
20           alert("file uploaded")
21           window.location.href = "?doing=create-img-v"
22         }else{
23           alert('file not uploaded')
24         }
25       },
26     })
27   })
```

Die Funktion für ein Rubrikbild funktioniert ähnlich der obigen, jedoch werden weniger Parameter benötigt, um ein Bild mit dem richtigen Namen zu speichern.

Codeverzeichnis 47: Admin Applikation Upload Rubrik Post Request

```
// Rubrikbild
  $("#but_upload-r").click(function(){
    var fd = new FormData()
    var files = $('#file')[0].files[0]
    var rubrik = $('#rubrik_img').val()

    fd.append('file',files)
    fd.append('rubrik',rubrik)

    $.ajax({
      url: baseUrl + '/rubriken',
      type: 'post',
      data: fd,
      success: function(response){
        if(response != 0){
          alert("file uploaded")
          window.location.href = "?doing=create-img-r"
        }else{
          alert('file not uploaded')
        }
      },
    })
  })
})
```

Der PostController reagiert auf diesen Aufruf und führt die jeweiligen Funktionen aus. Diese befinden sich in einer separaten Datei, bereits vorgestellt mit den Codeverzeichnissen 43 bis 45, und geben eine Rückmeldung über den Erfolg zurück. Dieser Rückgabewert wird vom Controller weiter an die JavaScript Funktion geleitet. Diese reagiert mit einer entweder positiven oder negativen Meldung.
Abschließend ist in diesem Kapitel die Kommunikation zwischen Applikation und Datenbank nachfolgend erläutert. Ebenfalls wird die PHP-Erweiterung MySQLi verwendet, um die Datenbankverbindung zu realisieren. Im Grunde sind die jeweiligen Funktionen ähnlich zu denen, die für die Veranstaltungsapplikation verwendet werden. Jedoch werden mehr Schreibe- statt Leseaktionen durchgeführt. Die erste Funktion dient auch dem ersten Reiter, dem Erstellen von Veranstaltungen. Dieser Funktion werden alle benötigten Daten zu einer Veranstaltung übergeben. Mit diesen wird anschließend ein SQL Statement erstellt, welches die Veranstaltung in die Datenbank schreibt. Bevor die Daten in das Statement gesetzt werden, wird überprüft, ob überhaupt Daten enthalten sind. Ist beispielsweise ein String kürzer als ein Zeichen und somit leer, wird stattdessen ein '-' gesetzt. Der nachstehende Code führt das Erstellen aus:

Codeverzeichnis 48: Admin Applikation Datenbankfunktionen 1.1

```
function newVeranstaltung($data)
{
  $connection = new mysqli(servername, username, password, database);

  if($connection->connect_error)
  {
    $connection->close();
    return "error";
  }
```

Im ersten Teil der Funktion wird überprüft, ob eine Verbindung zur Datenbank aufgebaut werden kann. Ist dies nicht der Fall bricht die Funktion ab und gibt *error* zurück. Ist die Verbindung jedoch erfolgreich zur Datenbank hergestellt worden, durchläuft die Funktion den *else* Teil folgendermaßen:

Codeverzeichnis 49: Admin Applikation Datenbankfunktionen 1.2

```
    else
    {
      $veranstaltung = array();

      foreach ($data as $key => $value)
      {
        if($key !== "Teilnehmer_Max" || $key !== "Aktiv")
        {
          if(strlen($value) > 0)
          {
            $veranstaltung[$key] = $value;
          }
          else
          {
            $veranstaltung[$key] = '-';
          }
        }
        else
        {
          $veranstaltung[$key] = $value;
        }
      }

      $sql = "insert into Faehrbook_veranstaltungen (
        Art,
        Rubrik,
        Titel,
        Bereich,
        Dozent,
        Zeit,
        Ort,
        Mitbringen,
        Voraussetzungen,
        Teilnehmer_Max,
        Kosten,
        Details,
        Aktiv)
          values
          (
            '" . $data['Art'] . "',
            '" . $data['Rubrik'] . "',
            '" . $data['Titel'] . "',";

            $sql .= "'" . $veranstaltung['Bereich'] . "',";
            $sql .= "'" . $veranstaltung['Dozent'] . "',";
            $sql .= "'" . $veranstaltung['Zeit'] . "',";
            $sql .= "'" . $veranstaltung['Ort'] . "',";
            $sql .= "'" . $veranstaltung['Mitbringen'] . "',";
            $sql .= "'" . $veranstaltung['Voraussetzungen'] . "',";
            $sql .= "" . $veranstaltung['Teilnehmer_Max'] . ",";
            $sql .= "'" . $veranstaltung['Kosten'] . "',";
            $sql .= "'" . $veranstaltung['Details'] . "',";
            $sql .= "" . $veranstaltung['Aktiv'] . "";
            $sql .= ");";

      $result = $connection->query($sql);
      $connection->close();
      return $result;
    }
  }
```

Nachdem eine Veranstaltung erstellt wurde, können sich die Einzelheiten im nach hinein ändern, wie beispielsweise die Zeit oder der Ort. Um dies auch innerhalb der Datenbank zu aktualisieren, ist die nachstehende Funktion implementiert. Diese bekommt ebenfalls die Daten übergeben und zunächst wird eine Datenbankverbindung aufgebaut und geprüft. Ist die Prüfung erfolgreich, wird das SQL Statement aus den übergebenen Daten zusammengebaut und anschließend ausgeführt. Danach wird die Verbindung, wie bei allen Datenbankfunktionen, wieder geschlossen. Als Rückgabewert wird ein Boolean, also true beziehungsweise false, zurückgegeben, abhängig von der Antwort der Datenbank.

Codeverzeichnis 50: Admin Applikation Datenbankfunktionen 2

```
function updateVeranstaltung($veranstaltungsid, $data)
{
  $connection = new mysqli(servername, username, password, database);
```

```
4
5     if($connection->connect_error)
6     {
7       $connection->close();
8     }
9     else
10    {
11      $sql = "UPDATE Faehrbook_veranstaltungen SET ";
12
13      foreach ($data as $key => $value)
14      {
15        if(is_string($value))
16        {
17          $sql  .= $key . " = '" . $value . "',";
18        }
19        else
20        {
21          $sql  .= $key . " = " . $value . ",";
22        }
23      }
24
25      $sql = substr($sql, 0, -1);
26      $sql .= " WHERE VeranstaltungsID = " . $veranstaltungsid;
27      $result = $connection->query($sql);
28      $connection->close();
29      return $result;
30    }
31  }
```

Ist eine Veranstaltung obsolet geworden, kann diese entfernt werden. Dafür wird der Funktion die VeranstaltungsID, dir Rubrik und der Titel übergeben. Anhand der ID wird das SQL Statement zu Löschung innerhalb der Datenbank ausgeführt. Im gleichen Zug wird das zugehörige Bild anhand der Rubrik und des Titel vom Server entfernt. Dies ist folgendermaßen umgesetzt:

Codeverzeichnis 51: Admin Applikation Datenbankfunktionen 3

```
1   function deleteVeranstaltung($veranstaltung)
2   {
3     $tmp = explode(";", $veranstaltung);
4     $veranstaltungsid = $tmp[0];
5     $rubrik = $tmp[1];
6     $titel = $tmp[2];
7
8     $connection = new mysqli(servername, username, password, database);
9
10    if($connection->connect_error)
11    {
12      $connection->close();
13    }
14    else
15    {
16      $sql = "DELETE FROM Faehrbook_veranstaltungen WHERE VeranstaltungsID =
            $veranstaltungsid";
17      $result = $connection->query($sql);
18      unlink("apps/faehrbookevents/img/veranstaltungen/" . $rubrik . "_" . $titel .
            ".jpeg");
19      $connection->close();
20      return $result;
21    }
22  }
```

Im Umkehrschluss kann ein Teilnehmer aus einer Veranstaltung entfernt werden, falls dieser von sich aus nicht der Veranstaltung verlässt. Um diesen Teilnehmer zu entfernen, muss einerseits in der Veranstaltung die Teilnehmerliste, sowie die Teilnehmerzahl aktualisiert und andererseits die Veranstaltung aus der jeweiligen Benutzertabelle gelöscht werden. Da die einzelnen Teilnehmer in einem String stehen, muss dieser aufgesplittet und wieder ohne den entfernten Teilnehmer zusammengesetzt werden. Der Funktion müssen somit die UserID und VeranstaltungsID übergeben werden, damit das Entfernen durchgeführt werden kann. Diese Funktion ist nachstehend realisiert:

Codeverzeichnis 52: Admin Applikation Datenbankfunktionen 4

```
1  function deleteTeilnehmer($userid, $veranstaltungsid)
2  {
3    $connection = new mysqli(servername, username, password, database);
4
5    if($connection->connect_error)
6    {
7      $connection->close();
8    }
9    else
10   {
11     $sql = "SELECT Teilnehmer_Current from Faehrbook_veranstaltungen where
           VeranstaltungsID = '$veranstaltungsid'";
12     $result = $connection->query($sql);
13     $current = $result->fetch_assoc()['Teilnehmer_Current'];
14     $current = $current - 1;
15
16     $sql = "SELECT Teilnehmer from Faehrbook_veranstaltungen where
           VeranstaltungsID = '$veranstaltungsid'";
17     $result = $connection->query($sql);
18     $teilnehmer = $result->fetch_assoc()['Teilnehmer'];
19
20     $tmp = explode(';', $teilnehmer);
21     $new_teilnehmer = '';
22
23     foreach($tmp as $value)
24     {
25       if($value !== $userid)
26       {
27         $new_teilnehmer .= $value . ';';
28       }
29     }
30
31     $sql = "Update Faehrbook_veranstaltungen set Teilnehmer = '$new_teilnehmer'
           where VeranstaltungsID = '$veranstaltungsid'";
32     $connection->query($sql);
33
34     $sql = "Update Faehrbook_veranstaltungen set Teilnehmer_Current = $current
           where VeranstaltungsID = '$veranstaltungsid'";
35     $connection->query($sql);
36
37     $sql = "Delete FROM Faehrbook_$userid where Events = '$veranstaltungsid'";
38     $connection->query($sql);
39
40     $connection->close();
41   }
42 }
```

Die letzten beiden Funktionen dienen zum Anlegen und Löschen von Rubriken. Es existieren weitere Funktionen, wie beispielsweise das erhalten aller Rubriken, doch diese werden nicht genauer erläutert, da diese denen entsprechen, welche in der Veranstaltungsapplikation ihren Einsatz finden. Um eine neue Rubrik zu erstellen, wird lediglich dessen Titel in die Datenbank geschrieben. Demnach benötigt die Funktion nur den Titel der neuen Rubrik und ist folgendermaßen implementiert:

Codeverzeichnis 53: Admin Applikation Datenbankfunktionen 5.1

```
1  function newRubrik($rubriktitel)
2  {
3    $connection = new mysqli(servername, username, password, database);
4
5    if($connection->connect_error)
6    {
7      $connection->close();
8      return "error";
9    }
```

Wie bei allen anderen Datenbankfunktionen auch, wird zunächst die Verbindung geprüft und bei positiver Rückmeldung, wird das Erstellen der neuen Rubrik ausgeführt.

Codeverzeichnis 54: Admin Applikation Datenbankfunktionen 5.2

```
1    {
```

```
2       $sql = "insert into Faehrbook_Rubriken (Titel)
3           values ('" . $rubriktitel . "');";
4       $result = $connection->query($sql);
5       $connection->close();
6       return $result;
7     }
8   }
```

Im Gegensatz zum Erstellen, führt die letzte Funktion das Löschen einer bestehenden Rubrik aus. Dieser Funktion wird lediglich der Titel der Rubrik übergeben und anhand diesem aus der Datenbank entfernt. Ebenfalls wird im gleichen Zug das zugehörige Bild vom Server gelöscht. Dies ist wie folgt im Code umgesetzt:

Codeverzeichnis 55: Admin Applikation Datenbankfunktionen 6

```
1   function deleteRubrik($rubriktitel)
2   {
3     $connection = new mysqli(servername, username, password, database);
4     if($connection->connect_error){
5       $connection->close();
6     }else{
7       $sql = "DELETE FROM Faehrbook_Rubriken WHERE Titel = '$rubriktitel'";
8       $result = $connection->query($sql);
9       $connection->close();
10      unlink("apps/faehrbookevents/img/rubriken/" . $rubriktitel . ".png");
11      return $result;
12    }
13  }
```

Mit dieser letzten Funktion ist die Implementierung der beiden Applikationen, die für das Faehrbook benötigt werden, abgeschlossen. Das nächste Kapitel befasst sich mit dem Ergebnis und gibt einen Ausblick auf mögliche Ver- und Nachbesserungen innerhalb der Applikationen, sowie der dahinter stehenden Codebasen. Die nachfolgenden Bilder stehen stellvertretend für die Umsetzung der zweiten Spalte der Admin Applikation:

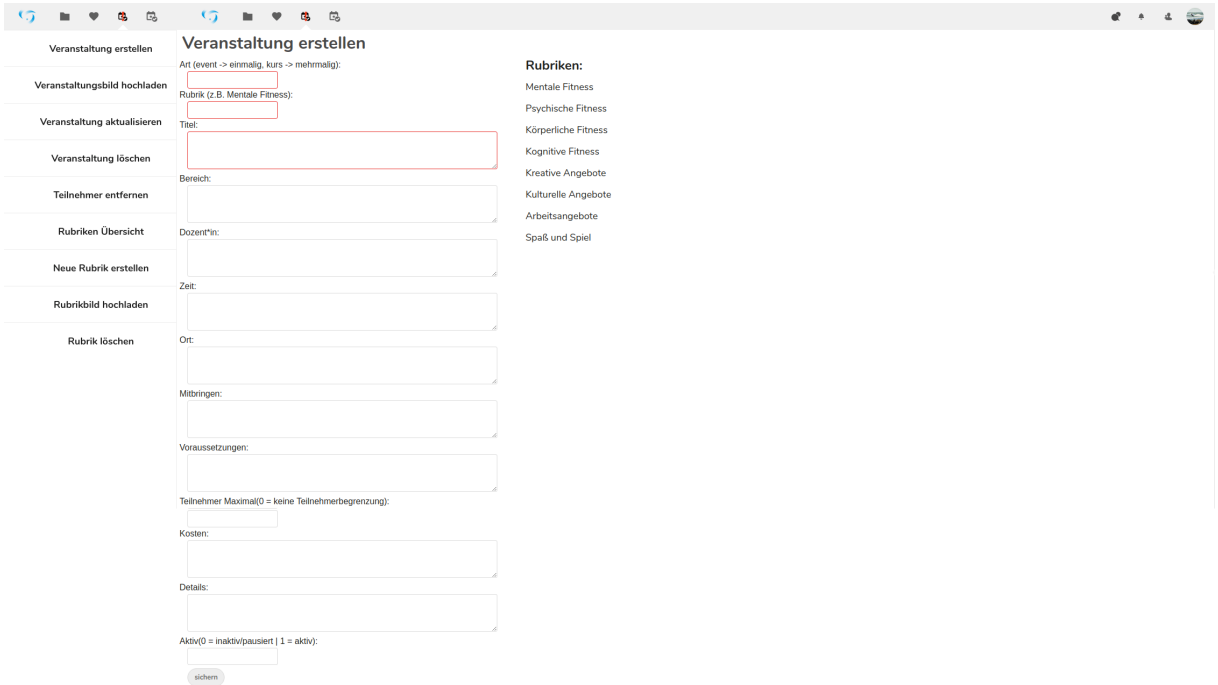


Abbildung 12: Admin Applikation Funktion create

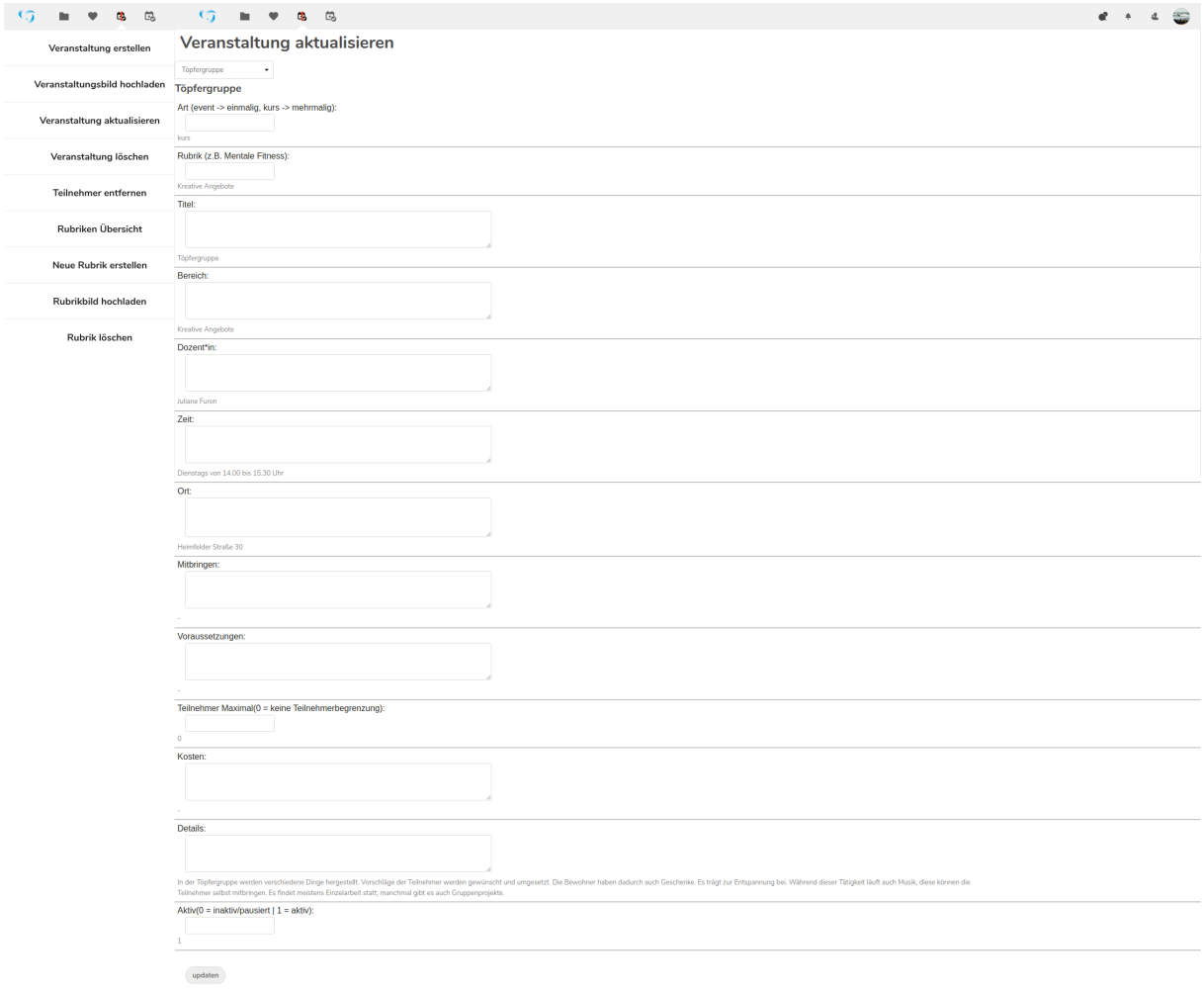


Abbildung 13: Admin Applikation Funktion update

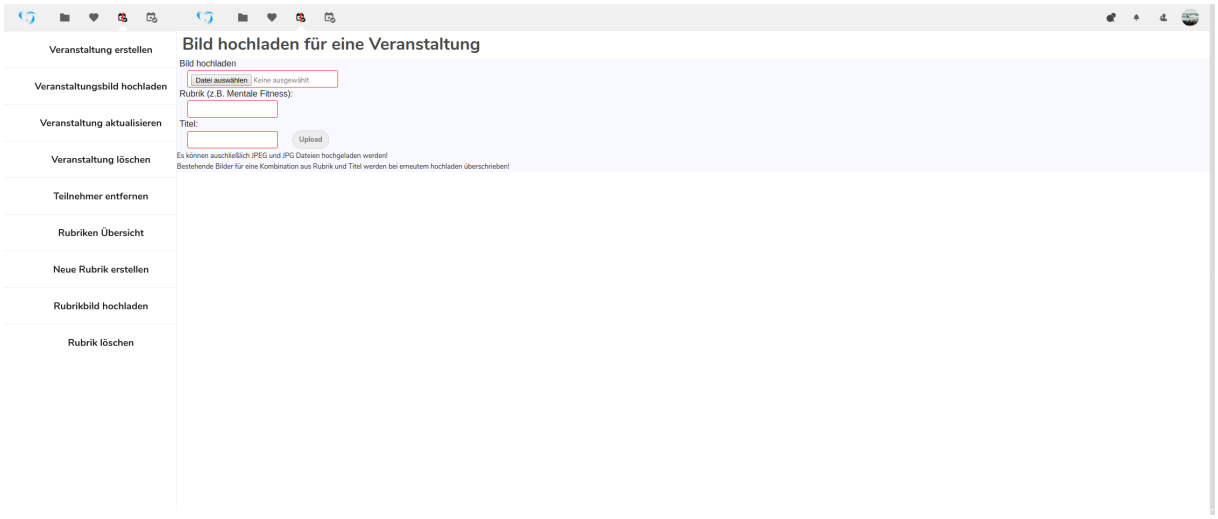


Abbildung 14: Admin Applikation Funktion create-img-v

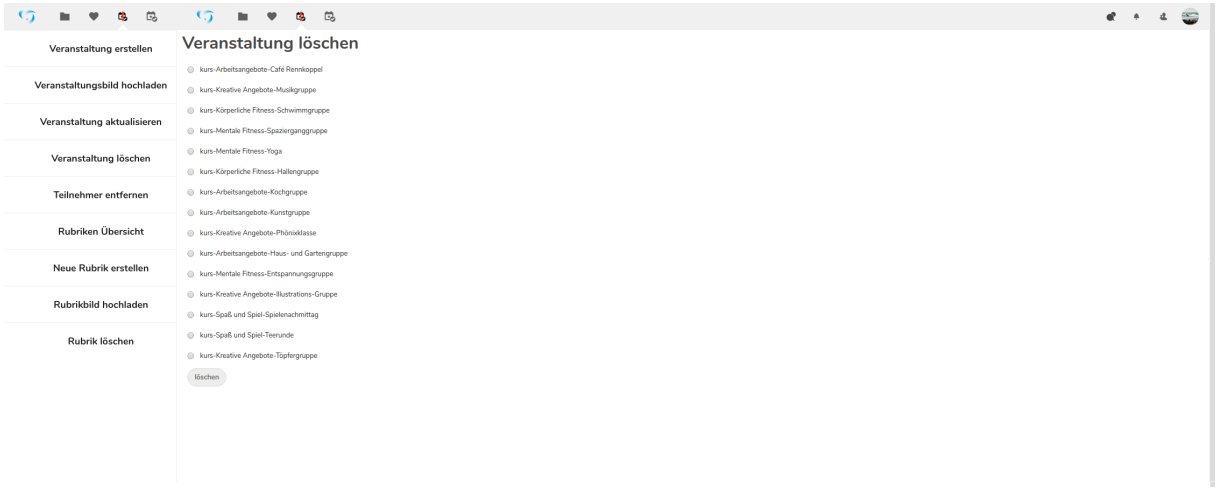


Abbildung 15: Admin Applikation Funktion delete

5 Diskussion

Dieses Kapitel befasst sich mit der Diskussion des Ergebnisses der Implementierung und gibt einen Ausblick auf mögliche Optimierungen. Im ersten Unterkapitel ist das Ergebnis der Implementierung vorgestellt und diskutiert, im Hinblick auf die Anforderungen an die Applikation. Ebenfalls ist bei der Diskussion ein Vergleich zwischen Konzeption und Implementierung inbegriffen. Im darauffolgenden Unterkapitel gibt es einen Ausblick auf mögliche Erweiterungen der Funktionalitäten der Applikation, sowie Nach-, beziehungsweise Verbesserungen der jeweiligen Codebasen. Dieses Kapitel dient grundlegend zur Reflexion des Projektes im Bezug auf die Gegenüberstellung von Anforderungen und Umsetzungen.

5.1 Ergebnis

In diesem Kapitel geht es um das Ergebnis der Implementierung. Vorweg ist festzuhalten, dass die Konzeption vollständig umgesetzt ist und darüber hinaus haben sich, wie zuvor dargestellt, weitere Funktionen im Verlauf der Implementierung ergeben, die ebenfalls umgesetzt sind. Um die Administration von der Darstellung für Benutzer abzugrenzen, sind zwei Applikationen entstanden. Diese können innerhalb der Nextcloud für verschiedene Benutzergruppen freigegeben werden. Dies vereinfacht deutlich die Abgrenzung, sodass Klienten keinen Zugriff auf die administrativen Funktionen erhalten. Die grundlegende Idee, Veranstaltungen darzustellen und zu verwalten ist umgesetzt. Die Benutzer der Veranstaltungsapplikation können sich alle Veranstaltungen der Faehre anzeigen lassen und erhalten so einen Überblick. Eingeteilt sind die Veranstaltungen in die zwei Oberkategorien *Kurse* und *Events*. Kurse sind mehrmalige und Events einmalige Veranstaltungen, die sich wiederum in verschiedene Kategorien, wie beispielsweise Mentale Fitness oder Arbeitsangebote, aufteilen. Mittels drei Spalten lässt sich innerhalb der App navigieren. Es gibt ebenfalls einen Reiter, der dem jeweiligen Benutzer die Veranstaltungen anzeigt, an denen er teilnimmt. Somit ist das Zu-, beziehungsweise Absagen zu Veranstaltungen für die Klienten möglich. Dies ist neben einer simplen und übersichtlichen Darstellung der Veranstaltungen die Hauptfunktion, welche von Seiten der Faehre gewünscht ist. Damit ist die Veranstaltungsapplikation in ihren Anforderungen vollständig umgesetzt. Die zugehörige administrative Applikation ist komplexer aufgebaut und verfügt über mehr Funktionalitäten. Einteilen lassen sich die Funktionen in drei Kategorien, die erste befasst sich mit dem Administrieren der Veranstaltungen, die zweite mit der Verwaltung von Rubriken und die letzte handhabt das Hochladen der zugehörigen Bilder von Veranstaltungen und Rubriken. In der ersten Kategorie können neue Veranstaltungen erstellt und bestehende aktualisiert werden. Gegebenenfalls lassen sich obsolete Veranstaltungen löschen und werden somit dem Benutzer nicht mehr angezeigt. Im Grunde gelten die gleichen Funktionen für Rubriken, jedoch lassen sich diese nicht aktualisieren. Stattdessen kann ein Administrator sich die verfügbaren Rubriken anzeigen lassen. Ein grundlegender

Teil der Darstellung sind Bilder. Im administrativen Bereich lassen sich diese jeweils für Veranstaltungen und Rubriken hochladen. Erweiternd zu diesen drei Kategorien ist eine Funktion umgesetzt, die es ermöglicht, einzelne Teilnehmer einer Veranstaltung zu entfernen. Dies dient dazu den Teilnehmer bei Nichterscheinen abzumelden, sodass kein Platz unnötigerweise belegt wird. Mit dieser Funktion sind alle administrativen Anforderungen erfüllt.
Abschließend ist festzuhalten, dass alle grundlegenden Anforderungen erfüllt sind. Dem hinzufügend ergaben sich weitere Funktionalitäten im Laufe der Implementierung, die ebenfalls umgesetzt worden sind. Somit entspricht das Ergebnis der Konzeption und dem grundlegenden Gedanken hinter dieser Anwendung.

5.2 Ausblick

Im Zuge der Entwicklung ergeben sich einige Ver- und Nachbesserungen, die aufgrund der vorgegebenen Zeit nicht mehr den Weg ins Projekt gefunden haben. Dieses Kapitel befasst sich mit den möglichen Optimierungen, die in Zukunft eingepflegt werden könnten. Der erste Abschnitt geht genauer auf Optimierungen der Codebasen ein, die ohne großen Aufwand umgesetzt werden können. Die darauf folgenden Optimierungen verändern den Aufbau der Applikationen von Grund auf, was wiederum mit deutlich mehr Aufwand einhergeht. Der letzte Abschnitt behandelt Funktionen, die nicht notwendig für die Applikationen sind, jedoch die Funktionalität erhöhen und gegebenenfalls die Ergonomie verbessern.
Die jeweiligen Codebasen der Applikationen können in jedem Fall verbessert und überarbeitet werden. In erster Linie kann immer die Ergonomie des Codes verbessert werden, durch beispielsweise Einrückung, Bezeichnung der Variablen oder Kapselung in Funktionen. Diese Optimierungen finden lediglich auf der Seite des Entwicklers statt und haben keinen Einfluss auf die Anwendung an sich. Die folgenden Optimierungen auf Codeebene verändern die Funktionalität und Handhabung der Applikationen. Die erste Möglichkeit der Verbesserung bezieht sich auf die Kommunikation der Anwendungen mit der Datenbank. Wie im Kapitel 4 mehrfach zu sehen, werden die SQL Anfragen als Strings erstellt und einzeln an die Datenbank gesendet. Diese Strings können manipuliert werden und ermöglichen so einen SQL-Injection Angriff. Dabei wird das SQL Statement so verändert, dass der Angreifer beispielsweise an sensible Daten, wie Passwörter, gelangt. Um dies zu verhindern können sogenannte *prepared Statements* verwendet werden, die eine Manipulation des Strings verhindern. Das SQL Statement wird zuvor festgelegt und mit Platzhaltern versehen. Die Datenbank prüft vor Ausführung des Statements die Gültigkeit der gesetzten Parameter. Eine weitere Möglichkeit die Kommunikation mit der Datenbank zu optimieren ist, dass die Anfragen zunächst gesammelt werden, sofern eine Funktion mehrere aufweist, wie beispielsweise das Entfernen eines Teilnehmers aus einer Veranstaltung. Anschließend werden die gesammelten Anfragen als sogenanntes *Batch* ausgeführt. Dies kann zu weniger Datenverkehr zwischen Applikation und Datenbank führen. Ebenso können Anfragen

effizienter ausgeführt werden, da diese nicht nacheinander abgearbeitet werden müssen. Das Datenbankmanagementsystem kann selbst eine effizientere Reihenfolge für das Abarbeiten der Anfragen wählen. In dieser Implementierung der Applikationen öffnet jede Funktion, die mit der Datenbank kommuniziert eine eigene Verbindung und schließt diese am Ende. Unter Umständen kann es dazu kommen, dass mehrere Verbindungen gleichzeitig geöffnet werden, die bei hohem Datenverkehr die Datenbank ausbremsen könnten. Um dies zu verhindern kann das Entwurfsmuster *Singleton* auf die Verbindung angewendet werden. Das Entwurfsmuster besagt, dass nur genau eine Instanz existieren kann. Somit lassen sich multiple Verbindungen zur Datenbank verhindern. Die eine Instanz einer Verbindung wird global innerhalb der Applikation zur Verfügung gestellt. Ebenfalls wird mit diesem Entwurfsmuster verhindert, dass eine Verbindung offen bleibt, wenn vergessen wird, diese am Ende einer Funktion zu schließen. Um nicht ständig mit der Datenbank zu kommunizieren, können bereits abgefragte Daten im Cache gespeichert werden, was zu verringerten Ladezeiten führen kann.
Des Weiteren kann das Routing innerhalb der Applikationen deutlich verbessert und optimiert werden. Aktuell wird lediglich für den Upload der Bilder das Routing verwendet. Die einzelnen Reiter der jeweiligen Anwendungen können in eigene Routen und Dateien ausgelagert werden. Ebenfalls wird damit das Einbauen von neuen Funktionalitäten vereinfacht und übersichtlicher. Die Sicherheit der Applikationen wird ebenso erhöht, da sensible, beziehungsweise interne Daten, wie beispielsweise die IDs der einzelnen Veranstaltungen, dem Nutzer nicht mehr sichtbar gemacht werden.
Die nächste Kategorie von Optimierungen bezieht sich auf die Struktur des Aufbaus der Applikationen. Frameworks haben kein Einsatz in dieser Implementierung gefunden, welche jedoch verwendet werden können. Es existieren zahlreiche JavaScript Erweiterungen, die für eine Umsetzung eingesetzt werden können. Die letzte Kategorie befasst sich mit Funktionen, die hilfreich, aber nicht notwendig sind. Einer dieser Funktionen ist beispielsweise eine Übersicht aller aktuellen Veranstaltungen innerhalb der Admin Applikation. Angezeigt werden Informationen, wie maximale und aktuelle Teilnehmerzahl, sowie die jeweiligen Teilnehmer. Dies ermöglicht dem Dozenten einen Überblick über die Veranstaltung und insbesondere der Teilnehmer zu erhalten. Ebenfalls hilfreich ist eine Kalenderübersicht, um so die einzelnen Termine besser im Blick zu haben. Im Bezug auf die einzelnen Termine, insbesondere die einmaligen Events, ist es praktisch, wenn vergangene Termine sich selbstständig aus der Datenbank entfernen. Hierfür muss die Tabelle der Veranstaltungen angepasst werden, sodass die Zeiten nicht mehr als String, sondern als Datum gesichert werden. Des Weiteren muss ein Hintergundjob erstellt werden, der in einem bestimmten Intervall die Veranstaltungen durchläuft, deren Datum mit dem aktuellen vergleicht und anhand der Differenz die Veranstaltung entfernt. Mit dieser Funktionalität wird der Verwaltungsaufwand für den Administrator verringert. Führt man das Erstellen einer Veranstaltung und das Hochladen des passenden Bildes zusammen, kann ebenso der Aufwand für den Administrator vereinfacht und verringert werden.

Im Laufe der Zeit können sich noch weitere Funktionalitäten oder Verbesserungen ergeben. Ein wichtiges Thema, welches in diesem Projekt vernachlässigt ist, ist das Testen der Applikationen. Mit Testen ist gemeint, dass das die Codebasen auf ihrer korrekten Funktionalität geprüft werden. Dies kann unter Anderem durch Frameworks umgesetzt werden, jedoch beschränkt sich das Testen in diesem Fall auf das direkte Ausprobieren der Funktionen innerhalb der Anwendungen. Da die Anzahl der Funktionalitäten klein und die Komplexität simpel gehalten ist, kann auf ein Framework verzichtet werden. Es können Funktionalitäten hinzukommen, die deutlich komplexer strukturiert sein können, als aktuelle. Daher bietet es sich an, vor Einführung neuer Funktionen, Tests einzubauen, die eine sichere Benutzung der Anwendungen garantieren. Ein möglicher Ansatz ist das sogenannte Test Driven Development zum Testen der Funktionen. Dabei handelt es sich um eine Entwicklungsstrategie, die den Ansatz verfolgt, dass erst der Test und darauf basierend die Funktion implementiert wird.
Zusammenfassend ist festzuhalten, dass die Applikationen Potential für zahlreiche Verbesserungen bieten, jedoch keine Notwendigkeit sind. Weitere Möglichkeiten der Nachbesserung ergeben sich ebenso mit der Benutzung dieser Anwendungen, bei der Fehler auftreten können, die bei der Konzeption und Entwicklung unentdeckt geblieben sind.

6 Fazit

Dieses Kapitel schließt die Bachelorarbeit ab und zieht ein Fazit zu dieser. Vorweg ist das Projekt sehr spannend, aber gleichzeitig auch herausfordernd. Aufgrund dessen ist dieses Kapitel in zwei Abschnitte aufgeteilt. Der erste Abschnitt geht genauer auf die spannenden Aspekte des Projekts ein und der zweite befasst sich mit den Herausforderungen, die das Projekt ausmachen.

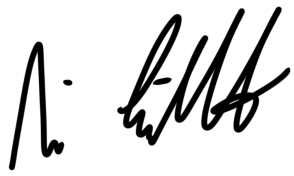
Zunächst ist die enge und notwendige Zusammenarbeit mit dem Kunden sehr spannend, da man das Projekt für jemand anderen umsetzt und dessen Ansprüchen an die Anwendung gerecht werden möchte. Hinzukommt, dass man im ständigen Austausch mit dem Kunden den aktuellen Stand mitteilt und auf gegebenenfalls gewünschte Änderungen schnellstmöglich reagiert, um nicht unnötig Zeit für obsolete Funktionalitäten zu verschwenden. Ein weiterer spannender Aspekt ist das Zusammenspiel der verschiedenen Technologien die zum Einsatz kommen. Dadurch konnte bestehendes Wissen angewendet und erweitert werden. Dieses Wissen ist hilfreich für die Umsetzung von zukünftigen Nextcloud Anwendungen, aber ist darüber hinaus für andere Projekte von Nutzen. Die gesammelte Erfahrung hilft allgemein beim Entwickeln von Anwendungen, sowie beim Kontakt mit Auftraggebern und Kunden. Das Erlernen von neuen Technologien stellt gleichzeitig eine Herausforderung dar. Das vorhandene Wissen reicht unter Umständen nicht aus und somit muss neues angeeignet werden. Zudem muss zunächst die grundlegende Struktur einer Nextcloud Applikation verstanden werden, bevor die Implementierung starten kann. Ein weiterer herausfordernder Aspekt ist, dass man vor einem unerwarteten Problem steht und dieses lösen muss. Diese Herausforderung macht gleichzeitig das Projekt sehr spannend, da die eigene Kreativität gefordert ist, jenes Problem zu lösen. Spannend aber auch herausfordernd ist das Umsetzen der Wünsche des Kunden. Ebenfalls ist während der Umsetzung von Anforderungen Kreativität gefordert.

Abschließend ist festzuhalten, dass das Projekt sehr spannend und die Umsetzung herausfordernd war. Die Möglichkeit neue Technologien zu lernen war ebenfalls ein sehr interessanter Aspekt des Projekts. Die Idee bis hin zur Konzeption, sowie Umsetzung sind erfolgreich abgeschlossen. Das Ergebnis entspricht dem Gedanken hinter den Anwendungen. Trotz auftretenden Herausforderungen während der Umsetzung, ist das Projekt erfolgreich abgeschlossen und das aus dem Studium erworbene Wissen konnte erfolgreich, praktisch angewandt werden.

7 Eidesstattliche Versicherung

Preikschat, Marius Kai
Name, Vorname

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem Titel **Konzeption und Implementierung einer webbasierten Applikation zur Darstellung und Verwaltung von Veranstaltungen** selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.



Oberhausen, 20.8.2019

8 Verzeichnisse

Codeverzeichnis

Literatur

[Kontor Consulting] [online]https://kontor.consulting [03. Juli 2019, 14:18]

[Die Faehre E.V.] [online]http://www.diefaehre-hamburg.de/ [28. Juni 2019, 11:02]

[Nextcloud] [online]https://nextcloud.com/de/ [28. Juni 2019, 13:45]

[ownCloud] [online]https://owncloud.org/ [05. Juli 2019 13:32]

[Nextcloud - About] [online]https://nextcloud.com/de/about/ [05. Juli 2019 13:33]

[Westfälische Hochschule Logo] [Bild]https://www.w-hs.de/typo3conf/ext/whs/Resources/Public/Images/Pagelayout/w-hs_pagelogo.png [16. Juli 2019, 14:37]

[Datenbanken und SQL] Schicker, Edwin: Datenbanken und SQL: Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL, 5. Aufl., Springer Vieweg, Wiesbaden 2017.

[Objektorierte Programmierung mit JavaScript] Bewersdorff, Jörg: Objektorientierte Programmierung mit JavaScript: Direktstart für Einsteiger, 2. Aufl., Springer Vieweg, Wiesbaden 2018.

[Webtechnologien] Bühler, Peter; Schlaich, Patrick; Sinner, Dominik: Webtechnologien: JavaScript - PHP - Datenbank (Bibliothek der Mediengestaltung), 1. Aufl., Springer Vieweg, Berlin 2018.

[PHP Info] https://www.php.net/manual/de/intro-whatis.php [17. Juli 2019, 9:31]

[HTML5 und CSS3] Bühler, Peter; Schlaich, Patrick; Sinner, Dominik:HTML5 und CSS3: Semantik-Design-Responsive Layouts (Bibliothek der Mediengestaltung), 1. Aufl., Springer Vieweg, Berlin 2017.

[MySQLi] [online]https://www.php.net/manual/de/intro.mysqli.php [18. Juli 2019, 10:15]

[Nextcloud Skeleton App] [online]https://apps.nextcloud.com/developer/apps/generate [24. Juli 2019, 9:09]

[Nextcloud Developer Intro] [online]https://docs.nextcloud.com/server/15/developer_manual/app/intro.html [24. Juli 2019, 9:12]

[Nextcloud JavaScript Frameworks] [online]https://docs.nextcloud.com/server/15/developer_manual/app/tutorial.html#adding-javascript-and-css [24. Juli 2019, 9:30]

[Visual Studio Code] [online]https://code.visualstudio.com/ [24. Juli 2019, 9:35]

[Visual Studio Code Erweiterung SFTP] [online]https://marketplace.visualstudio.com/items?itemName=liximomo.sftp [24. Juli 2019, 9:37]

[Nextcloud Design Guideline Content] [online]https://docs.nextcloud.com/server/15/developer_manual/design/list.html [27. Juli 2019, 14:00]

[SFTP] [online]https://www.wise-ftp.de/know-how/ftp_und_sftp_protokoll.htm [06. August 2019, 13:44]