Frankfurt University of Applied Sciences

Fachbereich 2 – Studiengang Engineering Business Information Systems

Bachelorthesis Thema

Entwicklung und Implementierung einer Integrationsmöglichkeit von Nextcloud in die Desktop-as-a-Service Anwendung oneye

Eingereicht zum Erlangen des akademischen Grads

Bachelor of Science

Autor: Pascal Kersten Müller

Matrikel-Nummer: 1221590

Referent: Prof. Dr. Christian Baun

Korreferent: Prof. Dr. Thomas Gabel

EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen sind, sind als solche kenntlich gemacht. Die Arbeit hat in gleicher Form noch keiner anderen Prüfbehörde vorgelegen.

Ort, Datum: Offenbach am Main, 08.11.2021

Unterschrift: 

DANKSAGUNG

Hiermit bedanke ich mich vor allem bei meiner Familie und allen Bekannten, die mir das Studium ermöglicht und mich in dieser Zeit begleitet und unterstützt haben. Ohne diese Unterstützung wäre ich nicht bis zu diesem Punkt gekommen.

Weiterhin möchte ich mich bei Herrn Prof Dr. Christian Baun bedanken, der sich bereit erklärt hat, als Hauptreferent zur Verfügung zu stehen. Zusätzlich hat er mich bei der Wahl des Themas unterstützt. Auch Herrn Prof Dr. Thomas Gabel möchte ich dafür danken, dass er sich als Korreferent zur Verfügung gestellt hat.

ZUSAMMENFASSUNG

Diese Bachelorarbeit beschäftigt sich mit dem Design und der Implementierung einer Schnittstelle zwischen dem Nextcloud-System und dem oneye-System auf Basis einer prototypischen Anwendung innerhalb des oneye-Systems. Dazu zählt das Bereitstellen einer Konfigurationsoberfläche für den Benutzer sowie das Umsetzen verschiedener Methoden zur Synchronisation der Dateien. Die Anwendung ermöglich dem Benutzer, Daten zwischen beiden Systemen synchron zu halten und in den jeweiligen Systemen zu bearbeiten.

ABSTRACT

This bachelor thesis is about the design and implementation of an interface between the nextcloud system and the oneye system. The base for the interface is a prototypical application of the oneye system. The application provides a configuration interface for the user and implements different methods for the synchronisation of the data. The application makes it possible to keep data of both systems equal without losing the ability to edit the data.

Inhaltsverzeichnis

ABKÜRZUNGSVERZEICHNIS

WebDAV	= Web Distributed Authoring and Versioning
PHP	= PHP: Hypertext Preprocessor
http	= Hypertext Transfer Protocol
JSON	= JavaScript Object Notation
XML	= Extensible Markup Language
HTML	= HyperText Markup Language
API	= Application Programming Interface
DTD	= Document Type Definition
YAML	= YAML Ain’t Markup Language
URL	= Uniform Resource Locator
RFC	= Request for Comments
OCS	= Open Collaboration Services
VFS	= Virtual File System

ABBILDUNGSVERZEICHNIS

1 EINLEITUNG

In der heutigen Welt verfügt jedes Gerät über Internet und ist in der Lage, einen Browser zu starten. Auch die Synchronisation von Daten kann von überall durchgeführt werden. Daher haben die Menschen den Anspruch, dass ihre Daten überall und zu jederzeit verfügbar sind. Natürlich besteht dann die Erwartung, dass diese Daten auch innerhalb einer Desktop-as-a-Service-Anwendung zur Verfügung stehen. Besonders im Zuge der COVID-19-Pandemie durch vermehrtes Homeoffice hat die Nutzung solcher Systeme zugenommen [1]. Viele Unternehmen und Institutionen waren daher gezwungen, das Arbeiten von Zuhause aus zu ermöglichen. Den Anwendern müssen dadurch auch die Daten zur Verfügung stehen. Aufgrund dessen besteht die Notwendigkeit, eine Verbindung zwischen einer Desktop-as-a-Service-Anwendung und einer Anwendung, die für die Bereitstellung der Daten verantwortlich ist, zu schaffen. Bisher existiert keine Lösung, die mit Open-Source-Software umgesetzt ist und somit keine Lizenzkosten verursacht.

1.1 ZIELSETZUNG

Die Zielsetzung dieser Bachelorarbeit besteht darin, ein prototypisch, aber funktionsfähiges Programm innerhalb des oneye-System zu entwickeln. Die Anwendung soll eine übersichtliche Konfigurationsoberfläche sowie die Möglichkeit zur Synchronisation von Dateien und Ordnern aus einem Nextcloud-System beinhalten. Als Einschränkung wird vorgegeben das, dass führende System das oneye-System ist. Als Desktop-as-a-Service Anwendung soll hierüber die Benutzerinteraktion erfolgen. Dabei ist die Voraussetzung, dass die Systeme sich händisch sowie automatisch synchronisieren. Alle Dateisystemoperationen, die innerhalb des oneye-System möglich sind, sollen in der Anwendung abgebildet sein. Damit sich die Anwendung möglichst nahtlos in das oneye-System integriert.

Der Quellcode soll verständlich und übersichtlich sein, da er später als Open-Source zur Weiterentwicklung des oneye-System zur Verfügung stehen soll. Zusätzlich sollen alle Teilabschnitte der Integration kommentiert sein, damit sich andere Entwickler einarbeiten und zur Verbesserung der Anwendung beitragen können.

2 STAND DER TECHNIK

In diesem Kapitel sind alle verwendeten und benötigten Technologien aufgelistet und beschrieben. Zusätzlich sind Alternativen aufgezeigt und es erfolgt der Vergleich zu den verwendeten Systemen.

2.1 NEXTCLOUD

Bei Nextcloud handelt es sich um eine Sammlung von Software, welche ursprünglich für das Speichern von Dateien auf einem selbst bereitgestellten Server entwickelt ist. Dabei ist die Software kostenfrei und unter einer Open-Source-Lizenz lizenziert. Sie ermöglicht den Zugriff auf die Daten per Webinterface oder über verschiedene andere Dienste wie beispielsweise WebDAV. Darüber hinaus hat sich Nextcloud zu einer Kollaborationssoftware entwickelt. Das bedeutet, dass auf dem ursprünglichen System aufbauend weitere Komponenten für beispielsweise Chat zwischen Benutzern, Teilen von Dateien oder Kalenderverwaltung integriert sind. [2] Bei all diesen Erweiterungen achtet das Entwicklerteam der Firma Nextcloud darauf, dass aktuelle Sicherheitsanforderungen wie Verschlüsselung oder Zwei-Faktor-Authentifizierung korrekt umgesetzt sind und dadurch ein adäquater Sicherheitsstandard für die bereitgestellten Dateien umgesetzt ist. [3] Aufgrund des Funktionsumfangs von Nextcloud konkurriert die Software direkt mit Produkten wie Google Drive von Google oder OneDrive von Microsoft. Dabei liegt der Vorteil von Nextcloud darin, dass der Betreiber die komplette Kontrolle über die Daten behält. Es muss keine fremde Infrastruktur zum Verwenden der Software benutzt werden. Ein weiterer Vorteil ist, dass die Installation auf allen gängigen Betriebssystemen sowohl für die Serversoftware wie auch die Clientsoftware umsetzbar ist. Durch die Clientsoftware ist eine Synchronisation der Daten auf die lokalen Systeme wie Laptop oder Handy der Nutzer möglich. Die Nutzer haben in einer dauerhaft vernetzten Welt überall Zugriff auf ihre gespeicherten Daten. Zusätzlich sind sowohl größere als auch kleinere Anwendungsfälle umsetzbar. Beispielsweise ist es möglich, ein Nextcloud-System für eine Familie zu Hause auf einem Einplatinencomputer wie dem Raspberry Pi bereitzustellen. Dieses bietet eine kostengünstige Alternative zu den konkurrierenden Systemen, welche oft mit laufenden Kosten abhängig vom verwendeten Speicherplatz finanziert werden. Im

Gegensatz zu kleineren Installationen gibt es auch die Möglichkeit der Installation eines Clusters für beispielsweise eine Universität, welches wesentlich mehr Kapazitäten benötigt, um allen Nutzern eine zufriedenstellende Benutzererfahrung zu bieten. Auch moderne Einsatzmethoden durch Container wie Docker sind möglich und führen zu mehr Flexibilität. Durch differente Möglichkeiten der Virtualisierung können die verfügbaren Ressourcen in verschiedenen Szenarien effizient genutzt werden und schaffen eine vielfältige Einsatzmöglichkeit für Nextcloud. [4]

2.2 ONEYE

Bei dem oneye-System handelt es sich um eine Desktop-as-a-Service-Anwendung, welche dem Benutzer einen Desktop im Webbrowser zur Verfügung stellt.

Abbildung 2.2.1: Desktop-as-a-Service Anwendung, Quelle: eigenes Bild

Der Benutzer hat so die Möglichkeit, seinen personalisierten Desktop von verschiedenen Geräten aus zu nutzen. Dabei hat er Zugriff auf seine Daten und Anwendungen, ohne dass er sich um die Synchronisation aller Inhalte kümmern muss. [5]

Das oneye-System unterstützt simultan die Anmeldung mehrere Anwender. Jeder Nutzer erhält einen eigenen virtuellen Desktop. Es baut auf einem Webserver auf und ist so unabhängig von der zugrunde liegenden Hardware und dem Betriebssystem. Solange für das Betriebssystem ein Webserver mit Unterstützung der Programmiersprache PHP verfügbar ist, kann auch das oneye-System darauf betrieben werden. Lizenziert ist das System als Open-Source-Software. Dadurch kann jeder Entwickler an dem System mitarbeiten und es aktiv weiterentwickeln. Die Nutzung des oneye-Systems ist bereits von einem Thin-Client aus möglich, da für das Anzeigen der Webseite nur eine geringe Rechenleistung erforderlich ist. Besonders in Bereichen, in denen nicht jedem Benutzer ein Fat-Client zur Verfügung steht, beispielsweise aus Kostengründen, kann so ein guter Kompromiss zwischen Funktionalität und Anschaffungskosten geschaffen werden. Solche Anwendungsfelder findet man in Bildungseinrichtungen oder gemeinnützigen Einrichtungen. [6]

2.3 ALTERNATIVEN ZUM ONEYE-SYSTEM

Im Folgenden werden alternative Systeme und Technologien zum oneye-System beschrieben und eingeordnet.

2.3.1 Virtuelle Maschinen

Eine virtuelle Maschine stellt dieselben Funktionen wie ein physischer Rechner dem Nutzer zur Verfügung. Dabei können mehrere virtuelle Maschinen von einem physischen Rechner zur Verfügung gestellt werden. Die virtuelle Maschine wird als Gastsystem bezeichnet und der physische Rechner als Hauptsystem. Pro Hauptsystem können mehrere Gastsysteme mittels softwaretechnischer Kapselung des Gastsystems von dem Hauptsystem den Benutzern zur Verfügung stehen. Jede virtuelle Maschine führt eine eigene Instanz eines Betriebssystems aus. Dabei können auf einem Hauptsystem verschiedene virtuelle Maschinen mit unterschiedlichen Betriebssystemen betrieben werden. [7] Ein Hauptsystem, welche virtuellen Maschinen verwaltet und zur Verfügung stellt, trägt die Bezeichnung Hypervisor. Die Ressourcen des Hypervisors, wie beispielsweise Arbeitsspeicher und Rechenleistung verteilen sich auf die zu betreibenden virtuellen Maschinen. [8]

Im Vergleich zum oneye-System werden pro Instanz wesentlich mehr Ressourcen benötigt, da auch das Betriebssystem des Gastsystems zum Betrieb Ressourcen benötigt. Dadurch sind pro Hauptsystem weniger Gastinstanzen möglich. Die finanziellen Kosten steigen bedingt durch höhere Anforderungen an die Ressourcen. Ein Vorteil besteht darin, dass die Gastsysteme voneinander getrennt sind. Daher kann es nicht zu negativen Wechselwirkungen zwischen ihnen kommen. Beispielsweise kann die Performance des Webservers beim oneye-System durch einen einzelnen Benutzer negativ beeinflusst werden. Dieses ist bei virtuellen Maschinen, die fest zugewiesenen Ressourcen haben, nicht möglich. Der Hypervisor trägt Sorge dafür, dass jedem System die richtigen Ressourcen zu jeder Zeit zur Verfügung stehen.

2.3.2 Remote-App-Systems

Weitere Möglichkeiten der Virtualisierung bieten Remote-App-Systems. Dabei handelt es sich um Systeme, welche nicht ein komplettes Betriebssystem kapseln, sondern einzelne Anwendungen stehen dem Benutzer gekapselt oder virtualisiert zur Verfügung. Der Benutzer kann damit Anwendungen verwenden, die nicht lokal auf seinem Computer ausführbar sind. Diese sind direkt von einem Server für den Benutzer bereitgestellt. [9] Beschriebene Systeme finden Anwendung, wenn Programme nicht direkt auf dem Computer des Benutzers ausführbar sind oder der Zugriff auf bestimmte Ressourcen nur über einen beschränken Zugriff möglich ist. Für den Benutzer stellt sich die Anwendung wie eine normal lokal installierte Anwendung dar. Er kann daher problemlos mit dieser interagieren und es erfolgt keine Einschränkung des Funktionsumfangs. Der Vorteil einer Remote-App besteht darin, dass sie sich nahtlos in die Benutzererfahrung des lokalen Systems einfügt. Damit sinkt die Komplexität für den Benutzer und auch ungeübte Benutzer haben die Möglichkeit, die Anwendung ohne potenzielle Schwierigkeiten zu nutzen. [10]

Im Vergleich zum oneye-System stellt ein Remoteapp-System nur einzelne Anwendungen zur Verfügung. Der Benutzer benötigt dabei trotzdem einen vollwertigen Desktop. Der Vorteil des oneye-System ergibt sich daraus, dass nur ein Browser benötigt wird. So ist eine Verwendung eines günstigen Laptops wie beispielsweise einem Chromebook auch

möglich. Vorteile bietet ein Remote-App-System bei der Vielfalt der verfügbaren Anwendungen. Abhängig vom Betriebssystem stehen mehr Anwendungen zur Verfügung als für das oneye-System. Dieses deckt zwar die grundlegenden Anforderungen ab, aber spezielle Software steht nicht zur Verfügung oder wurde noch nicht für das oneye-System entwickelt.

2.4 VISUAL STUDIO CODE

Mit Visual Studio Code ist es möglich, die eingesetzten Systeme zu bearbeiten. Mit Bearbeitung der auszuführenden Dateien innerhalb des oneye-System ist es möglich, dem oneye-System eine Anwendung hinzuzufügen, welche die Kommunikation zum Nextcloud-System übernimmt. Visual Studio Code kann kostenlos heruntergeladen und installiert werden.

Visual Studio Code unterstützt alle gängigen Programmiersprachen und stellt dabei alle wichtigen Komponenten einer Entwicklungsumgebung zur Verfügung. Zu diesen gehören beispielsweise IntelliSense, Möglichkeiten zum Debuggen und die Integration des Versionsverwaltungssystems Git. Zusätzlich kann es durch ein Plug-in System erweitert werden. Durch diese Erweiterungen des Hauptprogramms erfolgt eine Unterstützung der Benutzer bei der Bearbeitung der Dateien. Die Vielfalt der Plug-ins reicht von visuellen Anpassungen bis zu aktiver Unterstützung beim Programmieren, wie beispielsweise zur Verfügung stellen von häufig verwendete Programmcodeausschnitte. [11]

2.5 PHP

Bei PHP handelt es sich um eine Skriptsprache, welche besonders im Bereich der Webentwicklung Anwendung findet. Mithilfe ihrer ist es möglich, dynamisch generierte Inhalte auf einer Webseite anzuzeigen und abhängig der Anfrage des Benutzers anzupassen. PHP wurde 1995 entwickelt und ist in der Programmiersprache C geschrieben. [12] Ein Interpreter übersetzt den Quellcode in ausführbaren Code. Bei der Entwicklung wurde darauf geachtet, das bestehende Nachteile andere Skriptsprachen nicht auch in PHP zum Tragen kommen. Parallel sind Vorteile aus anderen Sprachen integriert worden. Zu ihnen zählen geringe Serverbelastung, Trennung von Layout und Code sowie

Browserunabhängigkeit [13]. Durch die Integration des Interpreters für PHP in die gängigen Webserver erfreut sie sich einer weiten Verbreitung und Popularität. Bei der Anwendung ist ausschlaggebend, dass die Ausführung des Quellcodes auf dem Server stattfindet. Der Browser ist an dieser nicht beteiligt und hat auch keinen Zugriff auf die Ausführung. Daher ist beispielsweise die Ausführung von Datenbankzugriffen, welche auf sensible Daten zugreifen, in PHP problemlos möglich. Für den Benutzer oder Browser stellt sich die Anfrage wie eine normale http-Anfrage dar.

Wie bereits in Punkt 2.2 erwähnt, ist das oneye-System in PHP geschrieben. Daher erfolgt auch die Programmierung der Verknüpfung zu Nextcloud in dieser Skriptsprache. Zu beachten ist, dass das oneye-System nicht mit der aktuellen Version von PHP kompatible ist. Es wird eine Installation der PHP-Version 5 benötigt, damit keine Kompatibilitätsprobleme entstehen. Passend zum Open-Source-Gedanken des oneye-Systems ist auch PHP eine Open-Source-Software und ermöglicht so die Erweiterung durch beispielsweise Module für einen speziellen Anwendungsfall. Auch entstehen keine Lizenzierungskosten oder Kosten für ein Abonnement bei der Nutzung von PHP, dieses würde auch dem Open-Source-Gedanken nicht entsprechen

2.6 JSON

JSON oder auch „JavaScript Object Notation“ ist ein Datenformat, welches zur Speicherung oder Austausch von Daten verwendet wird. Anwendung findet es vorwiegend bei der Entwicklung von Webanwendungen und API-Anwendungen. Durch das Speichern von Daten in lesbarer Textform und mit einer selbsterklärenden Syntax ist es leicht verständlich. [14]

```
{
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
}
```

Abbildung 2.6.1: Beispiel JSON Datenstruktur, Quelle: https://www.w3schools.com/whatis/whatis_json.asp

Wie in Abbildung 2.6.1 zu erkennen, erfolgt das Speichern der Daten immer als Schlüssel-Wert-Paar. Als Wert können verschiedene Datentypen wie Boolean, String oder Array abgespeichert werden. Dadurch ist es möglich komplexe Datenstrukturen, mit Verschachtelungen, in Textform abzubilden. [15]

PHP stellt wie andere Skript- oder Programmiersprachen Funktionen zur Verfügung, um Objekte oder Variablen verschiedener Typen in das oder von dem Datenformat JSON zu konvertieren. So ist es möglich, diese Daten zu speichern und diese bei einer neuen Anfrage erneut zur Verarbeitung innerhalb eines Skriptes zu laden.

2.7 XML

Die Abkürzung XML steht für „Extensible Markup Language“ und bezeichnet eine Auszeichnungssprache. Beim Beschreiben von hierarchischen strukturierten Daten findet sie Anwendung, um diese in Textform abzuspeichern. Zusätzlich findet das Abspeichern in lesbarer Form für einen Menschen oder eine Maschine statt. Besonders bei Protokollen zur Übertragung und Austausch von Daten ist die Verwendung von XML beliebt. [16]

Ähnlich zu einer HTML-Datei erfolgt der logische Aufbau einer XML-Datei mit sogenannten Startzeichen und Endzeichen. Die zu speichernden Daten sind durch Elemente innerhalb der XML-Datei repräsentiert. Bei Ihnen ist auch eine Verschachtelung ineinander möglich. Zusätzlich können Elemente mit Attributen ergänzt werden, um Eigenschaften dieser zu repräsentieren.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<element1>
     <sub-element1>Element Inhalt</sub-element1>
     <sub-element2>
          <sub-element3>Element Inhalt</sub-element3>
     </sub-element2>
</element1>
```

Abbildung 2.7.1: Beispiel XML, Quelle: eigenes Bild

Wie in Abbildung 2.7.1 dargestellt beginnt eine XML-Datei immer mit einer „Document Type Definition“, abgekürzt DTD. Diese beinhaltet beispielsweise, welche Version des XML Standards verwendet wird sowie die Codierung des gesamten Dokuments. Die DTD ist notwendig, damit ein Parser direkt erkennen kann, dass eine XML-Datei vorliegt und wie diese zu verarbeiten ist. Folgend sind die einzelnen gespeicherten Elemente in hierarchischerer Struktur abgebildet. [17]

Im Vergleich zu der JSON-Datenstruktur sind die einzelnen Knoten oder Elemente benannt. Auch bietet XML durch die Möglichkeit, Attribute den Elementen hinzuzufügen, zusätzliche Flexibilität, welche in besonderen Anwendungsfällen von Vorteil ist. Ein Nachteil, der dabei entsteht, ist, dass XML mehr Speicherplatz benötigt. Sowohl XML wie auch JSON benötigen einen Parser zum Einlesen bei der Verarbeitung der Daten.

3 DESIGN

In diesem Kapitel ist beschrieben, welche Lösungsansätze Verwendung finden und warum diese gewählt wurden. Darüber hinaus werden alternative Ansätze beleuchtet und eingeordnet.

3.1 EINRICHTUNG DER TEST- UND ENTWICKLUNGSUMGEBUNG

Dieses Unterkapitel beschreibt Komponenten, welche zur Umsetzung und Entwicklung Grundvoraussetzungen sind. Die Entwicklung erfolgt auf einem Linux basierten Betriebssystem, daher beziehen sich die geschilderten Punkte auf die Verwendung eines solchen Systems.

3.1.1 Nextcloud Docker

Für die Entwicklung einer Anwendung, welche eine Schnittstelle zwischen Nextcloud und dem oneye-System herstellt, ist es notwendig, dass eine Nextcloud-Instanz zum Testen zur Verfügung steht. Diese Testinstanz ist durch Containerisierung umgesetzt. Mit Docker beziehungsweise Docker Compose ist es möglich, eine Nextcloud-Instanz schnell und reproduzierbar zu erzeugen. Bei Docker Compose handelt es sich um eine Software, welche mit einer YAML Datei alle Grundkonfigurationsparameter der benötigten Services erstellt und konfiguriert. Zur Verwendung von Docker Compose ist eine Installation von Docker notwendig. Die Installation erfolgt über ein Skript, welches über die Webseite https://get.docker.com/ zur Verfügung steht [18] . Nach der Installation erfolgt mittels des Befehles „docker compose up“ das Einlesen und Ausführen der Docker Compose Datei[1]. In dieser sind die benötigten Docker Images für den Datenbankserver sowie die Nextcloud-Anwendung definiert. Auch beinhaltet diese die Umgebungsvariablen wie beispielsweise Zeitzone, Netzwerkkonfiguration und persistente Speicherorte. Sollte es im Laufe der Entwicklung zu Problemen mit der Nextcloud-Anwendung kommen, kann diese durch die Docker Compose Datei neu erstellt werden. Dabei gehen keine Daten verloren, außer der Anwender löscht diese explizit. Die Grundkonfiguration ist weiterhin in der Docker-Compose-Datei festgeschrieben und ist ein weiteres Mal aufrufbar. Der Vorteil

[1] Siehe Anhang, S. 56 [23]

dieser Lösung besteht darin, dass eventuelle Fehlkonfigurationen oder Fehlverhalten der programmierten Anwendung durch ein erneutes Erstellen zu korrigieren sind. Sofern eine manuelle Korrektur innerhalb der Nextcloud-Anwendung nicht möglich oder nicht erfolgreich ist. Auch für eine spätere Überführung in eine produktive Umgebung ergeben sich Vorteile. So ist das Weiterverwenden einer Docker-Compose-Datei innerhalb einer Cluster Lösungen wie „Docker Swarm“ oder „Kubernetes“ möglich. Dadurch sieht dieser Ansatz bereits Möglichkeiten für eine Skalierung bei steigender Last innerhalb eines produktiven Systems vor.

3.1.2 Oneye-System mit XAMPP

Als zweite Komponente innerhalb der Entwicklungsumgebung kommt XAMPP zum Einsatz. Bei XAMPP handelt es sich um ein Paket von Programmen, welches einen Apache-Webserver mit bereits integriertem PHP-Interpreter beinhaltet. Des Weiteren ist ein Datenbankserver (MariaSQL) und ein Interpreter für die Programmiersprache Perl beinhaltet. Es stehen verschiedene Versionen mit unterschiedlichen Versionen des PHP-Interpreters zur Verfügung. Das Paket kann kostenfrei heruntergeladen und installiert werden. [19] Beim Auswählen der Version ist eine Version auszuwählen, welche eine unterstützte PHP-Version für das oneye-System beinhaltet. Die Installation erfolgt mittels „.run“ Datei, welche alle nötigen Abhängigkeiten und die Programme des Paketes selbstständig installiert. Nach der abgeschlossenen Installation steht eine Benutzeroberfläche zur Verfügung, über die das Starten und Stoppen der einzelnen Dienste erfolgt. Nach dem Starten des Apache-Webservers wird das oneye-System in das „htdocs“ Verzeichnis des Webservers abgelegt. Dabei ist darauf zu achten, dass die richtigen Berechtigungen gesetzt sind. Sonst kann es zu Fehlern kommen und das oneye-System kann nicht ordnungsgemäß starten. Das Öffnen der Webseite bzw. des oneye-Systems kann über einen modernen Webbrowser erfolgen. Die Entwickler des oneye-Systems empfehlen die Verwendung von Google Chrome, Mozilla Firefox, Opera oder den Internet Explorer (Edge) [6] . Innerhalb der Test- oder Entwicklungsumgebung ist der Webserver, welcher das oneye-System zur Verfügung stellt, lokal auf dem Computer installiert. Daher erfolgt der Aufruf über die URL „localhost“. Auch eine Verschlüsselung des Aufrufes ist in

diesem Szenario nicht vorgesehen. Kommt es zu einer Überführung in ein produktives System sind die negativen Punkte zu bedenken und zu vermeiden. Eine Verschlüsselung des Aufrufs, sowie eine Lastverteilung bei produktiven Systemen sind zu bedenken und umzusetzen. Andernfalls entstehen erheblich Sicherheitsrisiken für die Daten der Benutzer und die Betriebssicherheit. Für Webserver gibt es dafür verschiedene Ansätze wie „Content Delivery Networks“ oder „Loadbalancer“.

3.2 ANWENDUNGEN INNERHALB DES ONEYE-SYSTEM

Die Schnittstelle zwischen dem oneye-System und dem Nextcloud-System ist mit einer Anwendung auf Basis des oneye-Systems umgesetzt. Dafür gibt es seitens des oneye-System Anforderungen an den Aufbau auf Dateiebene. Befolgt man diese Anforderungen, dann integriert sich die Anwendungen in das oneye-System und kann bereits vordefinierte Funktionen und Menüs verwenden. Dadurch sind grundsätzlich nur geringe Anpassungen am oneye-System notwendig.

Eine Anwendung innerhalb des oneye-Systems besteht grundsätzlich aus drei Dateien. Die erste Datei ist die „info.xml“ welche die Grundattribute wie Name, Kategorie, Version, Beschreibung, Autor, Lizenz, Typ und Icon beinhaltet. Ist diese Datei korrekt hinterlegt, dann listet das oneye-System die Anwendung innerhalb der Systemsteuerung als Anwendung. Ausgehend von der Systemsteuerung ist ein direkter Aufruf der Anwendung möglich. Beim Aufruf der Anwendung kommen die zwei weiteren Dateien zur Anwendung. Die Datei „app.eyecode“ bestimmt die grundsätzlichen Funktionen und die grafische Darstellung der Anwendung. Innerhalb dieser werden vordefinierte grafische Elemente des oneye-Systems hinzugefügt und konfiguriert. Mithilfe dieser ist eine einheitliche Gestaltung umsetzbar. Dabei sind Parameter verfügbar, um beispielsweise die Größe und Ausrichtung der Elemente anzupassen. In der letzten Datei der „events.eyecode“, sind Funktionen definiert die ausgeführt werden, wenn Events ausgehend von der Benutzeroberfläche stattfinden. Beispielsweise wenn ein Benutzer einen Knopf innerhalb der Benutzeroberfläche betätigt, dann kommt es zur Auslösung eines Events welches seine Funktionalität innerhalb der „events.eyecode“ definiert hat. Diese drei Dateien bilden das Grundkonstrukt einer Anwendung innerhalb des oneye-Systems. Auch

die Anwendung zu Nextcloud ist mittels dieser drei Dateien umgesetzt. Darüber hinaus ist die Einbindung weitere Dateien möglich. Dieses dient dann zur Übersichtlichkeit und logischen Trennung von verschiedenen Programmteilen. Aufgrund des Umfangs der Schnittstelle ist dieses während der Umsetzung der Schnittstelle notwendig. Die drei Dateien sind innerhalb des oneye-System in einem Ordner, der den Anwendungsnamen trägt im Unterverzeichnis „apps“ abgelegt. An dieser Stelle liegen alle Anwendungen, die innerhalb des oneye-Systems zur Verfügung stehen.

3.3 Mögliche Lösungsansätze Synchronisation

Das folgende Kapitel beschreibt die potenziellen Lösungsansätze bezüglich der Synchronisation von Dateien und Metadaten zwischen den einzelnen Systemen. Dabei werden Vor- und Nachteile sowie potenzielle Schwierigkeiten thematisiert.

3.3.1 WebDAV-Protokoll

Ein Ansprechen des Nextcloud-Systems ist mit dem WebDAV-Protokoll, ohne weitere Plug-ins oder Erweiterungen möglich. Dabeistehen die laut RFC 4918 definierten Methoden zur Verfügung. Bei dem WebDAV-Protokoll handelt es sich um ein Protokoll, welches eine Erweiterung zu dem http/1.1 Protokoll ist. Unter Anwendung dieses Protokolls ist es möglich Dateioperationen auf ein Dateisystem des Webservers durchzuführen. Dabei implementiert das WebDAV-Protokoll alle gängigen Dateioperationen wie beispielsweise Erstellen, Löschen, Verschieben oder Kopieren von Dateien. Möglich ist dieses, indem die Dateioperationen verschiedenen „Headern“ und Anfragetypen innerhalb des http/1.1 Protokoll zugeordnet werden. Ergänzend wird die URL zur Identifikation des Pfades oder der Datei verwendet, umgesetzt ist dieses durch ein zuordnen der Ressourcen (Dateien, Pfade usw.) zu einer URL. Zusätzlich unterscheidet das WebDAV-Protokoll zwischen Dateien und Ordnern. So benötigen manche Anfragen bedingt durch die zu bearbeitende Ressource verschiedene Parameter. [20]

Der Vorteil des WebDAV Protokoll liegt in der Anwendung. Es ist einfach anzuwenden, da es http-Anfragen verwendet zum Aufrufen der Ressourcen. Innerhalb von PHP ist die Umsetzung einer http-Anfrage gut umsetzbar. Dabei können auch die benötigen Optionen wie Methode oder „Header“ individuell für die Anfrage gesetzt werden. Das Weitern ist die

Struktur der Anfragen einfach und übersichtlich gehalten, sodass eine solche gut parametrisierbar ist. Nachteile des WebDAV-Protokolls liegen bei der Authentifizierung, da es keine Möglichkeit bietet, nur einen Hash oder Token zur Authentifizierung zu senden. Es ist immer nötig, den Benutzernamen und das Passwort zu übermitteln. Daher ist die Sicherheit der Benutzerzugangsdaten abhängig von der Verschlüsselung des Transports, die innerhalb eines produktiven Systems verwendet wird. Sollte es Schwachstellen oder Probleme bei der Transportverschlüsselung geben, sind die Benutzerzugangsdaten im Klartext auslesbar, welches ein erhebliches Sicherheitsrisiko darstellt.

3.3.2 Nextcloud-API

Die zweite Möglichkeit, Daten von dem Nextcloud-System zu beziehen, besteht in der „Open Collaboration Services“ API, auch OCS API genannt. Dieses ist eine eigens für Nextcloud konzipierte API-Schnittstelle. Sie ist in der Dokumentation des Nextcloud-Systems definiert. Ähnlich dem WebDAV-Protokoll stellt sie Methoden zur Interaktion mit dem Filesystem zur Verfügung. Darüber hinaus beinhaltet sie Methoden zur Interaktion mit dem Benutzerobjekt und verschiedenen weiteren Nextcloud-Features. [21] Eine weitere Gemeinsamkeit besteht darin, dass auch bei dieser Schnittstelle die Aufrufe zur Kommunikation auf Basis des http-Protokolls konzipiert sind. Im Vergleich zum WebDAV Protokoll ist diese Schnittstelle speziell für das Nextcloud-System konzipiert und bietet daher nicht die Universalität wie das WebDAV-Protokoll. Desweitern fehlen grundlegende Dateisystem-Operationen. Die Schnittstelle ist primär für die Verknüpfung von verschiedenen Nextcloud-Systemen sowie für die Verwaltung von Einstellungen innerhalb des Nextcloud-Systems implementiert.

3.3.3 Verwendete Variante der Synchronisation

Die zu erstellende Anwendung wird als Basis die WebDAV-Schnittstelle des Nextcloud-Systems nutzen. Aufgrund der Vor- und Nachteile stellt das WebDAV-Protokoll die benötigen Funktionen standardisiert zur Verfügung. Der Zielsetzung folgend liegt der Fokus auf der Synchronisation der Dateien und Ordner zwischen dem Nextcloud-System und dem oneye-System, daher werden zusätzlich Funktionen der OCS-API nicht benötigt. Passend zu den Dateisystem Operationen, die innerhalb des oneye-Systems vorstellbar sind, sind die http-Anfragen mit den korrekten Optionen zu erstellen. Es ist notwendig, dass alle Dateisystemoperationen abgebildet sind, da nur so eine vollständige Abbildung des Zustands des Dateisystems auf beiden Systemen möglich ist.

3.4 Konzeption der zu synchronisierenden Dateien

Zu bedenken sind verschiedene Szenarien der Synchronisation. Sie unterscheiden sich in der Anzahl der Dateien und dem Umfang. In den folgenden Unterpunkten werden die verschiedenen Anwendungsfälle und benötigten Funktionen erläutert und beschrieben. Die Synchronisation soll in einen vordefinierten Ordner innerhalb des oneye-Systems erfolgen. Dieser Ordner befindet sich in einem Pfad, welcher für jeden Benutzer einzigartig ist. Der Ordner trägt den Namen „Nextcloud“.

3.4.1 Vollständige Synchronisation

Das erste Szenario beschreibt die vollständige Synchronisation der Daten. Dabei sind alle Verzeichnisse mit den beinhaltenden Dateien zu übertragen. Ein Beispiel stellt dabei die initiale Synchronisation der beiden Systeme dar. Es ist davon auszugehen, dass schon Daten im Nextcloud-System hinterlegt sind. Diese bereits erstellten Daten müssen auf das oneye-System übertragbar sein. Darüber hinaus muss es möglich sein, die vollständige Synchronisation zu jeder Zeit ein weiteres Mal zu starten. Hierdurch werden beide Verzeichnisse erneut auf den gleichen Stand gebracht. Zu bedenken dabei ist, dass eventuelle Konflikte oder Unterschiede zwischen den Systemen Berücksichtigung finden. Solche Zustände sind programmatisch zu lösen.

Die Lösung von Konflikten erfolgt durch die Markierung der Datei mit dem älteren Bearbeitungsdatum als Konflikt. Die Datei mit dem jüngeren Bearbeitungsdatum erhält den

ursprünglichen Dateinamen. Beispielsweise erhält die ältere Datei den Dateinamen „Aufstellung.txt_conflict“ und die neuere Datei behält den ursprünglichen Namen „Aufstellung.txt“. So wird der Inhalt nicht verworfen und eine Korrektur kann gegebenenfalls durch den Benutzer erfolgen.

Des Weiteren ist zu bedenken, dass potenzielle Änderungen in den beiden Systemen wie beispielsweise das Erstellen einer neuen Datei oder das Verzeichnisses in beide Richtungen synchronisiert werden müssen. Damit solch eine Synchronisation erfolgen kann, muss die Anwendung wissen, welche Änderungen ausgehend von der letzten Synchronisation stattgefunden haben. Es ist notwendig, eine Liste des Dateisystems zum Zeitpunkt der letzten vollständigen Synchronisation vorzuhalten. Die Liste des Dateisystems mit allen Dateien lässt sich gut mit JSON-Datenstruktur abbilden. Diese Datei ist bei der nächsten vollständigen Synchronisation notwendig und ist daher innerhalb des Dateisystems des oneye-Systems abgespeichert.

Die vollständige Synchronisation folgt einem vordefinierten Ablaufplan, damit gewährleistet ist, dass alle Dateien korrekt verarbeitet werden. Bei der Verarbeitung sind Listen zu erstellen, die genau beschreiben, welche Daten entfernt, hinzugefügt oder verändert wurden. Nur so ist sicherzustellen, dass auch alle Änderungen erkannt und umgesetzt werden. Die Abarbeitung der Listen erfolgt in der vordefinierten Reihenfolge.

3.4.2 Eventbasierte Dateisystem Ereignisse

Das zweite Szenario beschreibt die Synchronisation einzelner Dateien. Dieses erfolgt auf Grundlage von Events, welche durch Ereignisse innerhalb des Dateisystems entstehen. Dabei ist klar definiert, welche Aktion auf die Datei ausgeführt ist. Alle verfügbaren Operationen des oneye-Systems sind durch die Anwendung zu unterstützen. Da keine Schnittstelle für eine Anwendung zu dem oneye-System existiert, ist diese zu erstellen. Bei der Erstellung der Schnittstelle sind alle Dateisystem-Operationen des oneye-Systems zu bedenken und umzusetzen. Diese Schnittstelle muss möglichst generisch umgesetzt sein, damit potenziell auch weitere Programme die Events abonnieren können. Angestrebt ist, eine Schnittstelle zu schaffen, auf die sich Programme aufschalten können und mit Parameter den eigenen Aufruf bei Auftreten eines Events bestimmen. Auch ist es möglich,

die Dateisystemevents zu deabonnieren. Die Verarbeitung der Events findet innerhalb der Anwendung statt. Es ist eine Funktion implementiert, welche die Änderungen zum Nextcloud-System synchronisiert und so die Systeme synchron hält. Dieser Ansatz bietet den Vorteil, dass keine größeren Datenmengen zu verarbeiten sind. Es sind weniger Ressourcen notwendig, um die Systeme dauerhaft synchron zu halten. Außerdem ist die Fehlerwahrscheinlichkeit reduziert, da nicht alle Daten in der Synchronisation Berücksichtigung finden, sondern einzig und allein solche, die unmittelbar verändert wurden. Von Nachteil ist dieses Verfahren nur, wenn das oneye-System nicht das führende System ist. Denn Änderungen innerhalb des Nextcloud-Systems werden nicht in Richtung des oneye-Systems synchronisiert. Da die Zielsetzung diese Variante für die Anwendung, welche im Rahmen dieser Bachelorarbeit konzipiert ist, ausschließt, ist dieser Nachteil nicht relevant.

3.5 VIRTUELLES DATEISYSTEM

Innerhalb des oneye-Systems stehen zwei Dateisysteme zur Verfügung. Diese unterscheiden sich in der Art, wie Dateien gespeichert werden. Bei dem sogenannten „real" Dateisystem werden die Daten direkt auf dem Dateisystem des Webservers gespeichert. Dabei gibt es keine zusätzlich Abstraktionsschicht. Im Gegensatz dazu steht das „virtuell" Dateisystem des oneye-Systems. Dieses trennt den Inhalt einer Datei von den Informationen wie Dateiname, Erstellungsdatum und weiteren Attributen der Datei.

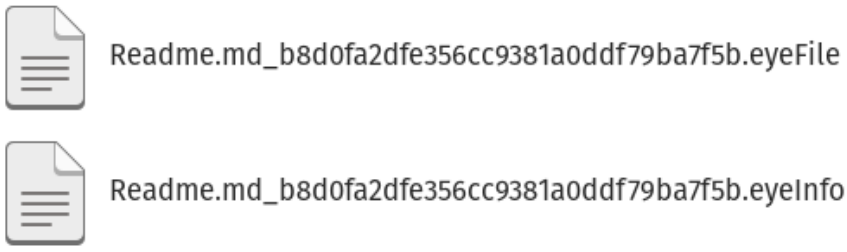

Abbildung 3.5.1: Virtuelles Dateisystem, Quelle: eigenes Bild

Wie in Abbildung 3.5.1 zu sehen, trägt die Datei mit den zusätzlichen Informationen die Dateiendung „eyeInfo" und die Datei, welche den Inhalt enthält, die Endung „eyeFile". Der

reale Dateiname der beiden setzt sich aus dem virtuellen Dateinamen und einer MD5 Hashsumme zusammen. Hierdurch ist eine Zuordnung jedes Paares zueinander innerhalb des virtuellen Dateisystems problemlos möglich. Der Vorteil, welcher sich aus der Verwendung des virtuellen Dateisystems für die Anwendung ergibt, ist, dass innerhalb der „eyeInfo“ Datei eigene Attribute hinzugefügt werden können. Die „eyeInfo“ Datei verwendet den Aufbau einer XML-Datei. Diese Struktur wird so modifiziert, dass die Information über die letzte Synchronisation einer Datei gespeichert ist. Dieses ermöglicht zusätzlich zum Bearbeitungsdatum auf der realen Dateiebene zu speichern, wann die Datei durch die Anwendung verarbeitet wurde.

Ein weiterer Vorteil besteht darin, dass zur Verarbeitung von Daten mittels des virtuellen Dateisystems bereits Funktionen verfügbar sind. Diese sind in das oneye-System integriert und beinhalten das Abfangen von Fehler und Optimierungen auf das oneye-System. Die Anwendung folgt damit dem Standard, wie Anwendungen innerhalb des oneye-Systems Dateien verarbeiten sollten.

3.5.1 Verlinkung oder Kopieren von Dateien

Auch wenn die Dateien über das virtuelle Filesystem des oneye-Systems abgelegt sind, stellt sich die Frage, ob alle Dateien vollumfänglich zu synchronisieren sind oder ob eine Verlinkung ausreichend ist. Der Unterschied beider Varianten besteht darin, dass bei einer Verlinkung erst zum Zeitpunkt der Bearbeitung eine vollständige Synchronisation der Datei aus dem Nextcloud-System erfolgt. Vorher werden nur „Links“ auf die Dateien erstellt. Dieses hat den Vorteil, dass nicht alle Dateien direkt auf dem oneye-System gespeichert sind und dadurch keinen Speicherplatz verwenden. Aber im Gegensatz dazu besteht der Nachteil, dass beim Bearbeiten einer Datei innerhalb des oneye-Systems diese eventuell erst aus dem NextCloud-System nachgeladen wird. Daraus resultierende Ladezeiten führen zu einer schlechteren Benutzererfahrung. Zur Implementierung einer Anwendung mit Verlinkung ist zusätzlich zur Anwendung eine neue Funktion innerhalb des virtuellen Filesystems notwendig. Ist die Anwendung so gestaltet, dass die zu synchronisierenden Dateien direkt komplett geladen und abgespeichert werden, dann verhalten sich die Vorteile und Nachteile genau entgegengesetzt.

Es überwiegen zwei Vorteile aufseiten des Ansatzes die Daten komplett zu kopieren. Diese sind, dass der Benutzer sofort mit den synchronisierten Dateien arbeiten kann, ohne durch eventuelle Ladezeiten beeinflusst zu werden. Sowie das keine weiteren Anpassungen im virtuellen Dateisystem, einer zentralen Komponente des oneye-Systems, durchzuführen sind. Damit entsteht auch kein Risiko eines Performanceeinbruchs oder ähnlicher Nebeneffekte einer solchen Veränderung.

4 IMPLEMENTIERUNG

Innerhalb dieses Kapitels ist der Hauptteil der Implementierung und damit dieser Arbeit beschrieben. Dabei ist beschrieben, wie die Anwendung technisch auf Grundlage der Designanforderungen implementiert ist. Zusätzlich ist mit Bildschirmaufnahmen gezeigt, wie der Verwendungsablauf der Anwendung durch einen Benutzer ist. Der erstellte Quellcode ist im Anhang hinterlegt. Integriert in das oneye-System ist dieser im Internet unter https://github.com/pascal-k-mueller/oneye abrufbar (Stand: 10.11.2021). Zusätzlich sind zum Installieren des modifizierten oneye-Systems die Installationsdateien unter https://github.com/pascal-k-mueller/installer zu finden (Stand: 10.11.2021).

4.1 ERWEITERUNG VIRTUELLES FILESYSTEM MIT EVENTS

Damit Events durch das virtuelle Filesystem, nachfolgend VFS genannt, des oneye-Systems generierbar sind, sind drei Funktionen notwendig. Es ist nötig, eine Funktion zu implementieren, welche die generierten Events verarbeitet und die abonnierten Anwendungen mit Parameter aufruft. Des Weiteren sind Funktionen zum Abonnieren und Deabonnieren der Events notwendig. Die Informationen, welche Anwendung die Events abonniert hat, sind dauerhaft für das VFS verfügbar zu sein. Daher erfolgt eine Speicherung auf zwei Ebenen. Für eine persistente Speicherung sind diese Informationen mittels einer JSON-Datei innerhalb des realen Dateisystems abgespeichert. Um die Anzahl an Lesezugriffen auf diese Datei zu reduzieren, sind die Daten zusätzlich innerhalb einer Variable mittels „Shared Memory“ gespeichert. Daher ist es nicht notwendig, dass bei jedem generierten Event die JSON-Datei erneut gelesen werden muss. Die jeweiligen Funktionen tragen Sorge dafür, dass beide Datenstände synchron sind.

Das Abonnieren der Events ist mit der „subscribeEvents“ Funktion des VFS möglich. Der Aufruf dieser muss mittels eines Arrays als Parameter erfolgen. Innerhalb dieses Arrays ist angegeben, um welche Anwendung es sich handelt und mit welchen Parametern diese Anwendung gestartet werden soll, damit die Events verarbeitet werden. Beispielhaft sieht dieser Aufruf wie in der folgenden Abbildung dargestellt aus.

```
//subscribe to fileevents
vfs('suscribeEvents',array("nextcloud","file_events"));
```

Figure 4.1.1: Abonnieren von Events, Quelle: eigenes Bild

In diesem Beispiel abonniert die Anwendung Nextcloud die Dateievents und erwartet den Parameter „file-events“ beim Aufruf. So ist es möglich, dass die Anwendung mittels eines Parameters entscheidet, welcher Teil auszuführen ist. Auch trägt die Funktion „subscribeEvents“ sorge dafür, dass bei einem Neustart des oneye-Systems bestehende Abonnements geladen sind.

Der Gegenspieler zu der bereits beschriebenen Funktion stellt die Funktion „removeEvents“ dar. Durch diese ist es Anwendungen möglich, die Events zu Deabonnieren. Der Aufruf dieser Funktion erfolgt gleich wie bei „subscribeEvents“. Wichtig dabei ist, dass genau die gleichen Angaben wie beim Abonnieren zu verwenden sind. Dadurch ist es auch möglich, dass eine Anwendung sich mehrfach auf die Events abonnieren kann. Voraussetzung ist, dass unterschiedliche Parameter zum Aufruf der Anwendung verwendet sind. Je nach Anwendung könnte multiples Abonnieren notwendig sein, um verschiedene Szenarien abbilden zu können. Für die Nextcloud-Anwendung ist dieses nicht notwendig, aber bei den Dateievents ist hierdurch ein möglichst universeller Ansatz umgesetzt.

Die dritte Funktion ist die Funktion „events“. Die Ausführung dieser Funktion erfolgt beim Auftreten eines Events. Sie liest aus der Variablen aus dem Shared Memory Segment aus, welche Anwendungen aufzurufen sind. Danach ruft die Funktion mithilfe des Kommandos „proc“ die einzelnen Anwendungen auf. Das Kommando „proc“ wird durch das oneye-System zur Verfügung gestellt. Es startet einen neuen Prozess mit den übergebenen Parametern. Dieses ermöglicht der Anwendungen mittels der vorher hinterlegten Parameter zu starten.

```
foreach ($subscribedApps as $sub) {
    proc('launch',array($sub[0],array($sub[1],array($method, $affectedFile))));
}
```

Figure 4.1.2: Ausführen von Events, Quelle: eigenes Bild

Wie in Abbildung zu sehen, findet zusätzlich zu dem hinterlegten Parameter die Übergabe eines Arrays statt. Diese beinhaltet, welche Dateioperation das Event ausgelöst hat, also welche Methode auf das reale Dateisystem durchgeführt wurde und welche Datei betroffen ist.

Eine weitere Änderung ist noch innerhalb des VFS notwendig. Bei dieser handelt es sich um die Anpassung aller Funktionen, die Dateioperationen ausführen. In diesen muss hinterlegt werden, dass die Funktion „events“ aufzurufen ist. Wichtig ist dabei, dass dieses zuerst geschieht, sofern die Dateioperation fehlerfrei durchgeführt ist. Die Methode, welche an die Funktion „events“ übergeben wird, entspricht dem Funktionsnamen der Dateioperation des VFS. Dadurch ergeben sich 13 verschiedene Dateioperationen, die mittels Dateisystemevents innerhalb des oneye-Systems überwachbar sind.

4.2 GLOBALE VARIABLEN

Zu Beginn der Anwendung innerhalb der Datei app.eyecode, sind globale Variablen definiert. Diese finden immer wieder innerhalb der Anwendung Verwendung. Daher sind sie einmalig definiert. Es sind drei globale Variablen definiert. Sie beinhalten den Pfad zu Konfigurationsdateien und die URL des verwendeten NextCloud-Systems.

```
global $config;
global $nextcloudURL;
global $nextcloudPath;
$config = eyeXML('getXMLconfig',array('nextcloud','conf.xml'));
$nextcloudURL = $config['nextcloud'][0]['url'][0];
$nextcloudPath = um('getCurrentUserDir') . 'files/Nextcloud' . '/';

include_once(EYE_ROOT.'/'.APP_DIR.'/nextcloud/func'.EYE_CODE_EXTENSION);
```

Abbildung 4.2.1: Globale Variablen, Quelle: eigenes Bild

Wie in Abbildung 4.1.1 zu sehen sind globale Variablen innerhalb von PHP mittels des Schlüsselworts „global“ zu definieren. Sie müssen erst deklariert werden, damit im nächsten Schritt eine Initialisierung stattfinden kann. So beschreibt es die Dokumentation. [22] Bei der Initialisierung werden Funktionen des oneye-Systems verwendet, um das persönliche Verzeichnis des aktuellen Benutzers zu erhalten und um eine

Konfigurationsdatei einzulesen, welche im XML-Format abgespeichert ist. Im späteren Programmverlauf kann durch „$GLOBALS['XYZ']“ der Zugriff auf die globale Variable erfolgen. Zusätzlich ist zu sehen, dass mit „include_once“ das Laden einer weiteren Datei erfolgt. Diese Datei beinhaltet die Funktionen der Anwendung. Diese sind ausgelagert, um die Übersichtlichkeit in der Hauptdatei der Anwendung zu wahren.

4.3 Benutzerinterface zur Konfiguration

Für die Interaktion des Benutzers mit der Anwendung ist ein Benutzerinterface implementiert. Das oneye-System stellt bereits Komponenten zur Verfügung, welche verwendet werden. Das Benutzerinterface zur Konfiguration umfasst die Punkte „Nextcloud URL“, „Benutzername“ und „Password“. Mit diesen Informationen ist die Synchronisation zwischen beiden Systemen möglich. Zusätzlich gibt es einen Knopf zum Speichern der Konfiguration. Der Nutzer kann das Konfigurationsmenü über die Systemsteuerung des oneye-Systems aufrufen. Bei den verschiedenen Komponenten des Benutzerinterface ist es möglich, über das setzten von Eigenschaften wie beispielsweise Höhe, Breite, Position und Zentrierung diese anzupassen. Die Verknüpfung der Komponenten erfolgt mithilfe einer Hierarchie. Jede Komponente ist zu einem übergeordneten Element verknüpft. Hervorzuheben ist dabei das Eingabefeld für das Passwort.

```
$myTextboxPassword = new Textbox(array(
    'father' => 'Nextcloud_Window_Content',
    'name' => 'Nextcloud_Password_Textbox',
    'width' => 150,
    'x' => 20,
    'y' => 125,
    'password' => 1
));
$myTextboxPassword->show();
```

Abbildung 4.3.1: Eingabefeld Passwort, Quelle: eigenes Bild

Bei diesem ist es notwendig, dass das Passwort nicht sichtbar ist. Dadurch soll es nicht möglich sein, das Passwort direkt vom Bildschirm abzulesen. Mittels des Parameters „password“ ist dieses möglich. Durch das setzten werden dem Benutzer Punkte angezeigt,

anstelle des Passworts in Klartext. Nachdem eine Komponente definiert ist, muss dem oneye-System mitgeteilt werden, dass diese anzuzeigen ist. Durch den Aufruf der Methode „show()“ erkennt das oneye-System, das die Komponente für den Benutzer sichtbar ist. Wie die in Abbildung 4.2.1 gezeigte Komponente sind alle Benutzerinterfacekomponenten innerhalb des oneye-Systems aufgebaut. Mittels des Stichworts „new“ gefolgt von dem Typ der Komponente ist die Erstellung eingeleitet. Darauffolgend findet die Definition der Eigenschaften statt. Nachdem die vorherigen Schritte erfolgt sind, ist die Komponente bereit und kann dem Benutzer angezeigt werden.

Es ist implementiert, dass beim Laden des Interface zur Konfiguration bestehende Konfigurationen des Benutzers dargestellt werden. So wird dem Benutzer ermöglich, die bestehende Konfiguration beispielsweise bei einer Änderung des Passworts zu bearbeiten, ohne die Daten komplett neu eintragen zu müssen. Zur Umsetzung dieser Funktion sind die Komponentenattribute durch die gespeicherten Daten aus der Konfigurationsdatei angereichert.

4.3.1 Speichern der Konfiguration

Das Speichern der Konfiguration erfolgt durch das Drücken des Knopfes zum Speichern. Dieses führt zu einem Event innerhalb des oneye-Systems. Beim Erstellen der „Button“ Komponente erfolgt mittels des Namens eine Verknüpfung zu einer vordefinierten Funktion innerhalb der „events.eyecode“ Datei. Der Name dieser Funktion entspricht einem vordefinierten Pattern. Dieses Schema beinhaltet den Anwendungsnamen und den Namen der „Button” Komponente. In dem Fall des Knopfes zum Speichern ist der Name der Komponente „Nextcloud_Save_Button“. Daraus resultiert das der Name der verknüpften Funktion „Nextcloud_on_Nextcloud_Save_Button“ sein muss. Wie zu erkennen, ist lautet das Schema „ANWENDUNGSNAME_on_NAMEDESBUTTONS“. Daraus resultiert das der Name einer Komponente eindeutig sein muss, da sonst keine Verknüpfung möglich ist.

Nach Aufruf der Funktion durch das oneye-System ist zu prüfen, ob bereits eine Konfiguration existiert. Wenn diese existiert, aktualisiert die Anwendung diese, andernfalls erstellt die Anwendung mit den neuen Konfigurationsparametern die Konfigurationsdatei. Dabei ist wichtig, dass im Fall einer neuen Konfiguration innerhalb der Konfigurationsdatei

der Zeitpunkt der letzten vollständigen Synchronisation auf „0“ gesetzt ist. Hierdurch weiß die Anwendung, dass bisher noch keine vollständige Synchronisation ausgeführt wurde. Nachdem die Konfigurationsdatei hinterlegt ist, kann theoretisch eine Synchronisation durchgeführt werden. Zusätzlich werden die Events des Dateisystems abonniert, damit die Verarbeitung von Änderungen durch die Anwendung stattfindet.

4.4 UNTERSTÜTZENDE FUNKTIONEN

In dem folgenden Kapitel sind Funktionen beschrieben, welche Unterstützende oder wiederkehrende Abläufe auslagern. Durch die Auslagerung erhält die Anwendung mehr Übersichtlichkeit und Struktur. Sie sind alle innerhalb der „func.eyecode“ Datei hinterlegt und werden daher beim Starten der Anwendung geladen.

4.4.1 Funktion zur Ausführung einer WebDAV-Anfrage

Diese Funktion wickelt die Erstellung und Durchführung von WebDAV-Anfragen ab. Mit Parametrisierung und Rückgabewert ist es möglich, alle nötigen WebDAV-Anfragen zu generieren und zu verarbeiten.

Der Aufruf der Funktion erfolgt mittels Parameter. Dabei ist zu unterschieden zwischen Parametern, die zwingend und optional zu übergeben sind. Zwingend erforderlich sind die gewünschte WebDAV-Methode und die aufzurufende URL zu übergeben. Als optional ist es möglich, eine Datei und ein Ziel zu übergeben. Je nach WebDAV-Methode sind verschiedene Elemente nötig. Anhand der bereitgestellten Parameter entscheidet sich innerhalb der Funktion, welcher Aufruf durchzuführen ist. Resultierend aus dieser Entscheidung erfolgt die Erzeugung der WebDAV-Anfrage. Die Erstellung der Anfrage erfolgt in drei Schritten. Zuerst sind die sogenannten „Options“ zu definieren. Diese beinhalten, dass es sich um eine http-Anfrage handelt sowie welche http-Methode, http-Header und Inhalt diese Anfrage transportiert. Im zweiten Schritt erfolgt das Erstellen des Kontextes der Anfrage durch den Befehl „stream_context_create“.

Im letzten Schritt findet die tatsächliche Anfrage mittels des Befehles „file_get_contents“ statt. Dieser Befehl biete zusätzlich zum Öffnen von Dateien die Möglichkeit, auch URLs aufzurufen. Voraussetzung dafür ist, dass die „fopen Wrappers“ verfügbar sind. Diese müssen in der Konfiguration von PHP aktiviert sein (PHP Doku). Mit Durchführung dieses Befehls ist die Anfrage beendet. Der Rückgabewert der Anfrage ist auch gleichzeitig der Rückgabewert der gesamten Funktion.

Innerhalb der Nextcloud-Anwendung unterstützt diese Funktion alle benötigten WebDAV-Methoden. Bei Ihnen handelt es sich um die Methoden „PUT“, „MOVE“, „GET“, „PROPPATCH“, „PROPFIND“, „MKCOL“, „DELETE und „COPY“ [20]. Besonders zu erwähnen ist die Methode „PROPPATCH“. Die Verwendung dieser Methode erfolgt, damit für Dateien innerhalb des Nextcloud-Systems ein Attribut gesetzt ist, welches die Zeit beinhaltet, wann eine Änderung dieser Datei zuletzt durch das oneye-System stattfand. Notwendig ist dieses Attribut, damit eventuelle Laufzeitverzögerungen nicht dazu führen, dass eine Datei als geändert oder neuer betrachten wird. Dadurch könnte es möglicherweise zu einer erneuten Synchronisation dieser Datei kommen, obwohl keine Änderungen erfolgt sind.

```
webDAVRequest("COPY", $urlFrom, "", $urlTo);
webDAVRequest("PROPPATCH", $urlTo, $filenameTo);
```

Abbildung 4.4.1: Aufruf WebDAV Funktion, Quelle: eigenes Bild

In Abbildung 4.4.1 sind zwei typische WebDAV-Aufruf der Nextcloud-Anwendung abgebildet. Im ersten Aufruf ist die Methode „COPY“ als Parameter übergeben. Gefolgt von den beiden URLs, welche definieren, wohin die Datei zu kopieren ist. Diese sind mittels der Variablen „$urlFrom“ und „$urlTo“ zu übergeben. Der dritte Parameter ist leer bzw. beinhaltet einen leeren String, da die „COPY“ Methode zwei Dateien innerhalb des Nextcloud-Systems kopiert und nicht lokal im oneye-System auf eine Datei zugreift. Ein Beispiel für solch einen Aufruf ist das Event des Dateisystems, welches beim Kopieren einer Datei auftritt. Im Vergleich dazu zeigt der zweite Aufruf eine weitere Methode. Dabei findet der Aufruf der beschriebenen „PROPPATCH“ Methode statt. Um die Zeit der letzten

Änderung zu hinterlegen, muss der Zugriff auf eine lokale Datei erfolgen. Daher ist als dritter Parameter die Variable „$filenameTo“ übergeben. Sie beinhaltet den Pfad zur lokalen Datei. Beim Betrachten beider Aufrufe zusammen erschließt sich der Zusammenhang. Zuerst findet der Kopiervorgang statt und anschließend ist sicher zu stellen, dass auch der Zeitpunkt der letzten Änderung beider Dateien gleich ist. Da sonst die Möglichkeit besteht, dass die zuvor beschriebenen Probleme auftreten.

4.4.2 Funktion zur Abbildung eines Verzeichnisses als Array

Beim Synchronisieren der Daten durch die Nextcloud-Anwendung, ist es immer wieder notwendig, dass zur Verarbeitung von Verzeichnissen und ihrer Inhalte diese einzulesen ist. Sinnvoll ist es, diese innerhalb eines Arrays zu speichern, damit die ursprünglichen Verzweigungen und Strukturen erhalten bleiben. Mittels dieser Funktion ist es möglich, ganze Verzeichnisse von beliebiger Tiefe in einer Variablen des Typs Array abzubilden. Problematisch ist, dass innerhalb des oneye-Systems keine Funktion existiert, die Inhalte eines Verzeichnisses rekursiv ausliest und zurückgibt. Das oneye-System stellt nur eine Funktion zur Verfügung, die auf eine Ebene beschränkt ist. Dieses führt dazu, das darunterliegende Dateien oder Verzeichnisse nicht durch diese Funktion berücksichtigt sind. Dieses Problem ist mittels einer rekursiven Funktion zu lösen. PHP unterstützt rekursive Funktionen, dieses bedeutet, dass eine Funktion sich selbst erneut aufruft und anschließend den Rückgabewert direkt verarbeiten kann. Durch die Nutzung dieser Möglichkeit ist es möglich, rekursiv die Verzeichnisse zu erfassen und anschließend in die Variable zu speichern. Technisch erfolgt der Aufruf der Funktion wie bei einer normalen Funktion mit „funktionsName(Parameter)“ innerhalb der Anwendung. Innerhalb der Funktion erfolgt mittels des Aufrufs „vfs('getDirContent',array($dir))“ das Auslesen der Verzeichnisse und Daten der aktuellen Ebene. Im Folgenden ist für jedes der gefundenen Objekte geprüft, ob es sich um ein Verzeichnis oder eine Datei handelt. Handelt es sich um ein Verzeichnis, erfolgt der rekursive Aufruf der Funktion mit dem neuen Verzeichnis als Parameter. Innerhalb der nächsten Verschachtelung der Funktion erfolgen genau die gleichen Schritte wie beschrieben, bis keine weiteren Verzeichnisse vorhanden sind. Die sich ergebende Struktur ist nach Beendigung der Funktion innerhalb des Arrays gespeichert. Besonders zum Abgleichen von Änderungen im Dateisystem oder beim

Erkennen von Änderungen ausgehend vom Nextcloud-System ist eine solche Möglichkeit der Abbildung von Dateisystemen notwendig. Dieses ermöglicht effizienter das Durchführen der beschriebenen Prozesse, ohne mehrmals auf die gleiche Ebene des Dateisystems zugreifen zu müssen.

4.4.3 Funktion zum Prüfen eines Pfades innerhalb eines Arrays

Eine weitere Funktion, die innerhalb der Anwendung benötigt ist, stellt die Funktion zum Prüfen von Pfaden innerhalb eines Arrays zur Verfügung. In PHP gibt es bereits eine Funktion, die einen ähnlichen Zweck verfolgt. Der entscheidende Unterschied besteht darin, dass diese Funktion nur in der ersten Dimension nach einem Wert sucht. Bei der Suche nach einem Pfad sind nacheinander mehrere Dimensionen bzw. Ebenen zu durchsuchen. Zusätzlich kann es sein, dass für jede Dimension ein anderer Wert gesucht ist. Die für die Anwendung implementierte Funktion erwartet als Eingabeparameter ein Array, welches zum Durchsuchen verwendet wird, sowie einen Pfad, der innerhalb des Arrays zu finden ist. Der übergebene Pfad entspricht dem Dateipfad innerhalb des Dateisystems. Dadurch ist es möglich, im Programmablauf genau zu prüfen, ob ein Pfad bzw. eine Datei weiterhin existiert oder nicht. Zusätzlich können gefundene Dateien oder Pfade aus einem Array eliminiert werden, um Dateien oder Pfade zu erhalten, welche noch nicht vorhanden sind.

Die Verarbeitung erfolgt mittels einer rekursiven Funktion, welche zusätzlich als Parameter eine Referenz zu einer Variablen übergeben bekommt. Mit dieser Referenz ist es möglich, aus der letzten Verschachtelung den finalen Rückgabewert zurückzugeben und weiterzuverarbeiten. Mittels „explode“ erfolgt die Umwandlung des Pfades zu einem Array. Das erste Element dieses Arrays stellt den gesuchten Schlüssel für die gegenwärtige Dimension dar. Wenn in der aktuellen Dimension der Schlüssel zu finden ist, dann erfolgt der rekursive Aufruf der Funktion. Dieses geschieht nur, wenn noch weitere Elemente innerhalb des Arrays des gesuchten Pfades existieren. Beinhaltet dieses keine weiteren Elemente mehr und alle vorherigen konnten ermittelt werden, dann gilt der Pfad als vollständig aufgelöst bzw. gefunden und der Rückgabewert ist „true“. Sollte die Erkennung eines Schlüsselelements nicht erfolgen, dann gilt der Pfad als nicht gefunden. In diesem

Fall gibt die Funktion den Rückgabewert „false“ aus. Durch diesen Rückgabewert besteht an der Stelle des Aufrufs die Möglichkeit, eine Entscheidung zu treffen, wie weiter zu verfahren ist. Beispielsweise findet diese Prozedur Verwendung in der Funktion „deleteChanges“, welche für die Löschung von Elementen im Nextcloud-System auf Basis von Änderungen im lokalen Dateisystem des oneye-Systems zuständig ist. Diese Funktion ist im nächsten Abschnitt beschrieben.

4.4.4 Funktion zum Entfernen lokaler Löschungen

Dieser Abschnitt beschreibt die Implementierung der Funktion „deleteChanges“. In dieser erfolgt das Löschen von Daten innerhalb des Nextcloud-System. Bedingt durch lokale Änderungen ist es notwendig, die Änderungen zum Nextcloud-System zu replizieren. Unter Zuhilfenahme der Funktion „checkArray“ werden Änderungen erkannt und daraus resultierend umgesetzt. Als Parameter sind zwei Arrays nötig. Ein Array repräsentiert den aktuellen Istzustand und das andere den Zustand nach der letzten Synchronisation. Anhand dieser beiden Arrays erfolgen die Vergleiche und das Rausfiltern der Abweichungen. Mit einer „foreach“ Schleife kommt es zur Prüfung jedes Schlüssels auf Existenz. Sollte dieser nicht im Istzustand vorhanden sein, dann erfolgt mittels der Funktion „webDAVRequest“ die Löschung des korrespondierenden Elements im Nextcloud-System. Dadurch finden die Identifikation und Angleichung der Unterschiede Schritt für Schritt statt.

Da das Dateisystem in einem multidimensionalen Array abgebildet ist, müssen erneut nach und nach die einzelnen Dimensionen durchsucht werden. Wie bereits bei vorausgegangenen Funktionen erfolgt dieses durch Rekursion. Auch die Funktion „deleteChanges“ ruft sich selbst auf, damit eine Verarbeitung des gesamten Arrays stattfinden kann. Dabei erfolgt die Änderung der Ebene, wenn es zur Feststellung eines weiteren Arrays innerhalb einer Dimension kommt.

Ein Rückgabewert ist nicht vorgesehen. Da die Funktion die notwendigen Anweisungen selbst durchführt und alle nötigen Schritte inkludiert sind. Es muss keine Auswertung oder Ähnliches erfolgen, was den Programmablauf an der Stelle des Aufrufes beeinflusst.

4.4.5 Funktion zum Aktualisieren der Informationen einer Datei

Das VFS stellt keine Funktion zur Verfügung, damit ein Aktualisieren der Informationen der eyeinfo-Datei erfolgen kann. Für die Synchronisation ist in dieser Datei gespeichert, wann die letzte Synchronisation dieser Datei stattgefunden hat. Besonders für die Verarbeitung von existierenden Dateien und dem Erkennen von Konflikten ist diese Information notwendig. Daher ist mittels der Funktion „updateVFS“ die Funktionalität geschaffen, um das XML der „eyeinfo“ Datei sowie das Änderungsdatum der „eyeFile“ Datei anzupassen.

Durch den ersten Parameter erfolgt die Übergabe der betreffenden Datei an die Funktion. Der zweite Parameter erwartet ein Array, welches die Informationen beinhaltet, die im XML zu hinterlegen sind. Zu diesen zählt die Zeit, zu der die Datei das letzte Mal modifiziert worden ist, sowie die Zeit der letzten Synchronisation. Zusätzlich kann noch die Änderung der Eigenschaft erfolgen, welche beschreibt, welche Anwendung zur Erstellung der Datei verwendet worden ist.

```
//Creating the xml information
if($fileInfoChanged) {
    $file = getRealFileName($filePath);
    file_put_contents($file . '.' . EYEOS_INFO_EXT, eyeXML('array2xml',array($fileInfo)));
}
if($modified) {
    touch($file . '.' . EYEOS_FILE_EXT, $modified);
}
```

Abbildung 4.4.1: Modifizierung VFS Information, Quelle: eigenes Bild

Zu Beginn des Vorgangs liest die Funktion die Daten vor der Aktualisierung ein und konvertiert diese aus dem XML-Format in ein Array. Im weiteren Verlauf aktualisiert die Funktion die einzelnen Felder des Arrays mit den neuen Werten. In Abbildung 4.4.1 ist zu sehen, dass nach der Prüfung auf Änderungen in den Informationen der Datei, das Array mittels des Befehles „eyeXML“ die Konvertierung in das XML-Format durchgeführt wird. Dabei unterscheidet das Programm, ob ausschließlich eine Änderung innerhalb des XML erfolgt ist oder ob auch das Bearbeitungsdatum der Datei anzupassen ist. Sollte dieses benötigt sein, dann passiert dieses unter Verwendung des Befehls „touch“. Dieser setzt das Bearbeitungsdatum der jeweiligen Datei auf den per Parameter übergebenen Wert.

4.5 VOLLSTÄNDIGE SYNCHRONISATION

In dem folgenden Kapitel ist beschrieben, wie die vollständige Synchronisation implementiert ist. Dabei sind die verschiedenen Phasen dieser dargestellt. Zusätzlich ist behandelt, wie der Aufruf erfolgt und die Integration in die Nextcloud-Anwendung des oneye-Systems. Auch Aspekte wie die Erhaltung von Daten für die zukünftigen Synchronisationen sind beschrieben.

Der Ablauf der vollständigen Synchronisation besteht aus mehreren Schritten, die nacheinander abzuarbeiten sind. Zu Beginn dieser Schritte steht das Deabonnieren der Dateievents, da bei der vollständigen Synchronisation viele Events entstehen und diese nicht durch die Nextcloud-Anwendung zu verarbeiten sind. Die Verarbeitung erfolgt innerhalb der vollständigen Synchronisation. Nachdem der Vorgang abgeschlossen ist, sind diese natürlich erneut zu abonnieren. Zuerst führt die Anwendung die Funktion zum Entfernen lokaler Löschungen auf Basis des aktuellen Dateisystems und dem Stand des Dateisystems der letzten Synchronisation aus. Darauf folgt das Abrufen einer Dateiliste aus dem Nextcloud-System. Anhand dieser Dateiliste erfolgt der weitere Ablauf der Synchronisation. Die Dateiliste beinhaltet alle Daten, die im Nextcloud-System hinterlegt sind. Dabei sind auch die notwendigen Attribute wie die URL bzw. der Pfad, das letzte Synchronisationsdatum und Bearbeitungsdatum vorhanden. Im nächsten Schritt erfolgt die Prüfung, welche Dateien im Nextcloud-System nicht mehr vorhanden sind. Diese Erkenntnis kann aus den Daten der letzten Synchronisation und der aktuellen Dateiliste des Nextcloud-Systems gewonnen werden. Es ist davon auszugehen, dass am Ende der letzten Synchronisation beide Systeme auf dem gleichen Stand sind. Daher sind die Unterschiede programmatisch zu ermitteln. Das Programm prüft als Nächstes für jede Datei, ob diese bereits existiert. Im Falle, dass die Datei bereits existiert, kommt es zur Prüfung, ob diese modifiziert ist. Dabei ist zu unterscheiden welche der Dateiversionen in den beiden Systemen die aktuelle ist. Sollte nicht verlässlich nachprüfbar sein, welche die aktuellere Version der Datei ist, dann erfolgt die Konfliktverarbeitung. Sollte die Datei dagegen noch nicht existieren, dann ist sie aus dem Nextcloud-System zum oneye-System zu synchronisieren. Die Datei wird innerhalb des oneye-Systems angelegt und der Inhalt vom Nextcloud-System kopiert. Wenn diese Schritte für jede Datei aus der Dateiliste des

Nextcloud-Systems abgeschlossen sind, dann löscht die Anwendung alle Dateien im oneye-System, welche im Nextcloud-System seit der letzten Synchronisation entfernt wurden. Nun steht nur noch die Synchronisation der im oneye-System neu erstellten Dateien und Pfade aus. Durch das Erkennen der Unterschiede zwischen dem lokalen Datenbestand nach der letzten Synchronisation und dem aktuellen lokalen Datenbestand des oneye-Systems können diese identifiziert werden. Wichtig dabei ist, dass darauf zu prüfen ist, dass es keine Überschneidungen mit Dateien des Nextcloud-Systems gibt. Diese Prüfung ist gegen die aktuelle Dateiliste des Nextcloud-Systems möglich. Zum Schluss der vollständigen Synchronisation steht die Erstellung einer aktuellen Dateiliste der synchronisierten Dateien im oneye-System, da diese zum Vergleichen für den nächsten Durchlauf benötigt wird.

4.5.1 Aufruf vollständige Synchronisation

Der Aufruf erfolgt über die „app.eyecode“ Datei. Wird beim Aufruf der Anwendung der Parameter „sync“ übergeben, dann führt die Anwendung eine vollständige Synchronisation durch. Dieser Aufruf kann ausgehend von einer anderen Anwendung innerhalb des oneye-Systems erfolgen oder ausgehend von der Benutzeroberfläche. Damit der Aufruf über die Benutzeroberfläche stattfinden kann, mussten Anpassungen an dieser durchgeführt werden. Im Datei-Explorer des oneye-Systems ist eine Schaltfläche zum Aufruf der Synchronisation hinzugefügt worden. Hierdurch hat der Benutzer eine Möglichkeit den Aufruf schnell und intuitiv zu veranlassen. Der Aufruf der vollständigen Synchronisation erfolgt nur manuell nach Veranlassung durch den Benutzer.

4.5.2 Verarbeitung der WebDAV Anfrage PROPFIND

Im zweiten Schritt der vollständigen Synchronisation verarbeitet die Anwendung unteranderem auf Basis der Liste von Dateien aus dem Nextcloud-System die Änderungen der Dateien. Der Abruf dieser Liste ist mittels der Funktion „webDAVRequest“ mit dem Parameter „PROPFIND“ und der URL des Nextcloud-Systems möglich. Darauf erhält das Programm ein XML-Dokument, welches alle Daten beinhaltet. Die Programmiersprache PHP stellt die Funktion „simplexml_load_string zur Verfügung. Mit dieser liest das Programm die Datei in eine Variable ein. Durch diese Funktion ist das erhaltene XML in einer Variablen des Typs Objekt der Klasse „SimpleXMLElement“ gespeichert. Daher ist

es möglich, direkt auf die einzelnen Elemente des XML-Dokuments zuzugreifen. Die Weiterverarbeitung dieser Elemente ist mit einer „foreach“ Schleife umgesetzt. Daher erfolgt die bereits beschriebene Verarbeitung jedes Elements nacheinander. Wichtig dabei ist, dass die Attribute der einzelnen Elemente im Namensraum „d“ gespeichert sind. Daher ist dieser beim Zugriff auf das XML-Element mit anzugeben. Andernfalls kann nicht auf die benötigten Attribute zugegriffen werden.

Bei den Attributen ist zu beachten, in welchem Format diese vorliegen. Besonders bei verschiedenen Zeitformaten kann es zu Problemen kommen. In der Nextcloud-Anwendung sind Zeiten im UTC Zeitstempel-Format angeben. In den Attributen des XML ist dieses nicht der Fall. Dort sind die Zeiten mittels Zeitzone aufgeführt.

```
$childLastModified = $child->children($namespaces['d'])->propstat->prop->getlastmodified;
$timeLastModified = (DateTime::createFromFormat('D, d M Y H:i:s e', $childLastModified))->getTimestamp();
```

Abbildung 4.5.1: Angleichung Zeitformat, Quelle: eigenes Bild

Um Probleme beim Vergleichen oder im späteren Programmablauf zu verhindern, findet eine Angleichung statt. Wie in Abbildung 4.5.1 zu sehen, wird die Angleichung durch die „DateTime“ Klasse mit der Methode „createFromFormat“ durchgeführt. Zuerst ruft das Programm das Attribut ab, welches die Daten bezüglich der Zeit beinhaltet. Danach können mit der Eingabe eines Formats, welches das Zeitformat der Daten aus dem XML repräsentiert, die Zeitdaten in das UTC Zeitstempel-Format umgewandelt werden.

4.5.3 Behandeln von Existierenden Dateien

Bei der Verarbeitung von existierenden Dateien ist hauptsächlich der Vergleich der Bearbeitungszeitpunkte ausschlaggebend für die Entscheidung, welche Datei aktueller ist. Als zusätzlicher Faktor spielt der Zeitpunkt der letzten Synchronisation eine Rolle. Damit eindeutig festzustellen ist, welche Datei aktueller ist, muss diese Datei sowohl aktueller sein als die korrespondierende Datei in dem andern System, und gleichzeitig darf nur in einem System seit der letzten Synchronisation eine Veränderung an der Datei durchgeführt worden sein. Sobald diese Entscheidung getroffen ist, kann die Aktualisierung der älteren Datei mit dem Inhalt der neueren stattfinden. Hierfür lädt das Programm mit der Funktion „webDAVRequest“ entweder diesen herunter oder hoch. Zusätzlich passt die Anwendung die Bearbeitungszeit der Datei in beiden Systemen an, damit bei der nächsten Synchronisation die Datei nicht erneut zu synchronisieren ist, sofern sich keine Änderungen ergeben haben.

4.5.4 Behandeln von Konflikten

Sollte es nicht möglich sein, bei existierenden Dateien, festzustellen, welche Datei aktueller ist, dann ist die Lösung dieses Konfliktes der nächste Schritt. Beispielsweise entsteht ein Konflikt, wenn in beiden Systemen Änderungen an einer Datei vorgenommen sind, ohne dass eine Synchronisation zwischenzeitlich stattgefunden hat. In diesem Fall kann die Anwendung nicht feststellen, welche Datei aktueller ist. Daher ist als Lösungsansatz implementiert, dass die Datei mit dem älteren Bearbeitungsdatum den Zusatz „_conflict“ erhält. Zusätzlich werden beide Dateien mit dem jeweiligen System synchronisiert. Dadurch ist der Konflikt gelöst und eine Korrektur kann durch den Benutzer erfolgen. Technisch ist diese Implementierung mit Funktionen des VFS und der Funktion „webDAVRequest“ umgesetzt. Der Vergleich der Bearbeitungszeiten ist mit einer „if“ Anweisung umgesetzt. In dieser evaluiert die Anwendung, welche Bearbeitungszeit größer ist. Da das UTC Zeitstemple-Format verwendet ist, kann der Vergleich der Werte direkt erfolgen. Eine weitere Umwandlung der Werte ist nicht notwendig.

4.5.5 Konsistenz der Daten (Dateilisten)

Wie in den vorherigen Kapiteln erwähnt führt die Anwendung mehrere Vergleiche der Daten durch. Damit dieses durch das Programm möglich ist muss die Struktur der Dateien und Pfade innerhalb des Dateisystems zum Zeitpunkt der letzten Synchronisation gespeichert werden. Es muss nur das Speichern der Struktur durchgeführt werden, da weitere Attribute und Eigenschaften direkt zusammen mit der Datei durch das VFS gespeichert sind. Das Speichern der Daten erfolgt durch eine JSON Datei. Diese Datei bildet die Struktur des Dateisystems der zu synchronisierenden Daten ab. Am Ende jeder vollständigen Synchronisation aktualisiert die Anwendung diese Datei.

4.6 VERARBEITUNG VON DATEISYSTEMEVENTS

Nachdem sich ein Dateisystemevent ereignet hat, ist die Anwendung mit dem Parameter „file_events“ aufzurufen. Intern erfolgt darauf der Aufruf der Funktion „fileEvents“ dabei werden die Parameter des Aufrufs der Anwendung weitergereicht. Die Funktion erkennt mit einem „Switch Case“ welche Operation auf das Dateisystem erfolgt ist. Je nach Operation setzt die Anwendung die notwendigen Änderungen im Nextcloud-System um. Besonders dabei ist, dass zuerst eine Prüfung durchgeführt wird, ob die Dateisystemevents die synchronisierten Daten betreffen. Sollte dieses nicht der Fall sein, dann verwirft die Anwendung das Event und es kommt zu keinen weiteren Schritten. Erst wenn die Anwendung erkennt, dass synchronisierte Daten betroffen sind, dann führt sie die benötigten Anweisungen aus. Für jede Event-Methode ist dediziert geregelt, welche Anweisungen durchzuführen sind. Sollte die Event-Methode der Anwendung nicht bekannt sein, dann erfolgen keine weiteren Schritte.

Abbildung 4.6.1: Dateiliste Synchronisation, Quelle: eigenes Bild

In der Dateiliste, in welcher die JSON-Dateistruktur verwendet ist, repräsentiert ein Schlüssel-Wert-Paar eine Datei. Wie in der Abbildung zu erkennen, ist der Dateiname sowohl als Schlüssel und Wert hinterlegt. Sonderzeichen führen nicht zu Problemen, da bei der Erstellung der Datei mit der Funktion „json_encode“ diese in das Unicode-Format umgewandelt werden. Bei der Konvertierung in die entgegensetzte Richtung mit der Funktion „json_decode“ ersetzt die Funktion die Unicode Zeichen mit den korrespondierenden Zeichen. Ordner die Inhalt enthalten sind, als JSON Objekt dargestellt. Im Gegensatz dazu sind Ordner ohne Inhalt als leere Arrays dargestellt. Gespeichert ist diese Dateiliste im Benutzerordner des betreffenden Benutzers. Dadurch können auch mehrere Nutzer parallel die Nextcloud-Anwendung verwenden.

4.7 Ablauf der Verwendung durch den Anwender

Vor der ersten Verwendung durch den Benutzer muss die Konfiguration der Anwendung durgeführt sein. Die Konfigurationsoberfläche ist über das Anwendung Menü des oneye-Systems aufrufbar. Eingeordnet ist es in der Kategorie „File Management“, wie in der folgenden Abbildung zu sehen.

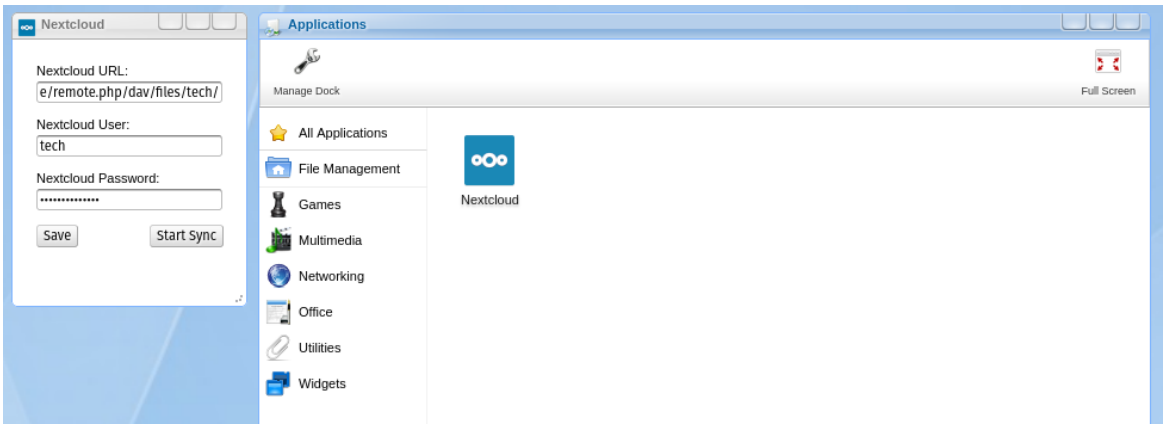

Abbildung 4.7.1: Konfigurationsoberfläche Nextcloud-Anwendung, Quelle: eigenes Bild

Des Weiteren ist in Abbildung 4.7.1 auf der linken Seite die Konfigurationsoberfläche der Nextcloud-Anwendung zu sehen. In dieser muss der User die URL des Nextcloud-Servers und seinen Benutzernamen sowie das Passwort für das Nextcloud-System hinterlegen. Versucht der Nutzer die Nextcloud-Anwendung vorher zu verwenden, erhält er eine Fehlermeldung, die ihn auffordert, zuerst die Konfigurationsdaten zu hinterlegen.

Nachdem die Konfigurationsdaten hinterlegt sind, kann der Benutzer entweder direkt aus der Konfigurationsoberfläche oder über den DateiExplorer die erste Synchronisation starten. Bei dieser legt die Anwendung in dem Home-Verzeichnis des Benutzers den Ordner Nextcloud an, sofern dieser noch nicht vorhanden ist. In diesen werden alle Dateien synchronisiert.

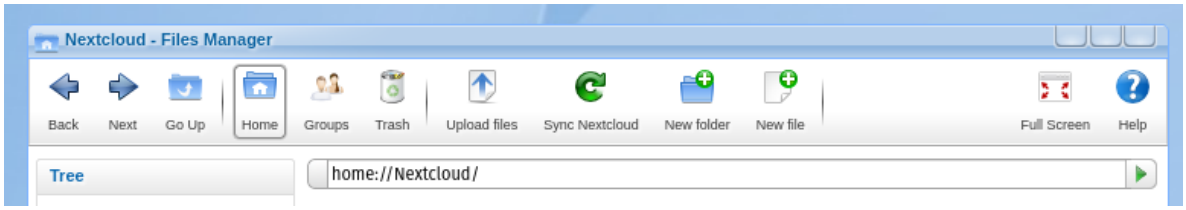

Abbildung 4.7.2: Modifizierte Symbolleiste, Quelle: eigenes Bild

Um dem Benutzer die Möglichkeit zu geben die Daten innerhalb des Nextcloud-Ordners schnell zu synchronisieren, ist die Symbolleiste mit einer Schaltfläche erweitert worden. Beim Benutzen der Schaltfläche „Sync Nextcloud“ führt die Anwendung eine vollständige Synchronisation durch. Zusätzlich zu den manuellen Methoden erfolgt automatisiert direkt beim Einloggen des Benutzers eine Synchronisation durch das System.

5 FAZIT

Die Implementierung einer prototypischen Anwendung zur Herstellung einer Schnittstelle zwischen dem oneye-System und Nextcloud-System konnte erfolgreich umgesetzt werden. Die gesetzten Zielsetzungen sind eingehalten. Auch sind die grundsätzlichen Anforderungen an die Anwendung eingehalten worden. Durch das WebDAV-Protokoll stand eine gute Lösung zur Verfügung, um die Synchronisation der Daten durchzuführen. Der Quellcode ist übersichtlich gestaltet und mit Kommentaren versehen. Daher sollte einer späteren Erweiterung der Anwendung nichts im Weg stehen. Auch konnten in den Tests keine Probleme bei der Performance festgestellt werden. Allerdings kann hierzu nur eine verlässliche Aussage getroffen werden, wenn ein groß angelegter Test zur tatsächlichen Last auf dem Server erfolgen würde.

Als schwieriger als erwartet stellte sich der Umgang mit dem oneye-System heraus. Für das System existiert keine öffentliche Dokumentation, die im Detail beschreibt, welche Komponenten das System zur Verfügung stellt. Daher dauerte die Einarbeitungsphase länger. Trotzdem konnte mit der Nextcloud-Anwendung eine zufriedenstellende Integration der Schnittstelle in das System erreicht werden. Auch das aufsetzten der Testumgebung stellte eine Herausforderung dar. Trotzdem das sich Techniken wie Docker und Webserver schon lange etabliert haben, stellt, das Betreiben dieser einen gewissen Aufwand dar. Ein rein lokales Testen ist nicht möglich gewesen, da zum Testen des gesamten Ablaufs beide Systeme benötigt werden. Trotzdem kann am Ende ein positives Resümee gezogen werden. Nachdem die beschriebenen Probleme gelöst waren, konnte dank der einfachen Verständlichkeit von PHP die Entwicklung ohne Probleme erfolgen.

5.1 AUSBLICK

In der vorgegebenen Zeit war es nicht möglich, alle Ideen zu perfektionieren, es gibt einige Prozesse, die in Zukunft verbessert oder ergänzt werden können. Innerhalb des Abschnittes der vollständigen Synchronisation kann der Quellcode weiter optimiert und übersichtlicher gestaltet werden. Eine gewisse Grundübersichtlichkeit besteht, diese ist aber noch nicht optimal. Auch die Auslagerung komplexer Vorgänge wäre möglich, damit eine Verschlankung des Quellcodes daraus resultiert.

In Zukunft könnte es hilfreich sein, die OCS API des Nextcloud-Systems einzubinden und dadurch den Funktionsumfang der Anwendung zu erweitern. Die Benutzer hätten dadurch die Möglichkeit Dokumente und Ordner direkt aus dem oneye-System heraus mit anderen Benutzern zu teilen. Die Daten wären weiterhin innerhalb des Nextcloud-Systems gespeichert und müssten dieses nicht verlassen. Auch ein doppeltes Abspeichern müsste nicht erfolgen. Daraus resultierende Datenredundanz auf beiden Systemen könnte vermieden werden.

Weiteren Verbesserungsbedarf besteht im Bereich der Algorithmus zum Finden von Änderungen im Dateisystem. Das Abarbeiten einzelner Dateien auf Basis des Änderungsdatums kann bei großen Datenmengen Probleme verursachen. Daher ist es sinnvoll, mit einem Hash-Verfahren aus mehreren Werten zu arbeiten. So wäre es auch möglich nicht jede Datei einzeln zu betrachten, sondern ganze Ordner mit einem Durchlauf zu verarbeiten.

Nachdem die Anwendung und die Anpassungen innerhalb des oneye-Systems weiter getestet wurden, ist es denkbar diese dem oneye-Projekt bereitzustellen. Dadurch würde das oneye-Projekt erweitert werden und neue Funktionen erhalten. Durch diesen Beitrag würde man zu einer Erweiterung eines Open-Source-Projektes beitragen.

6 ANHANG

6.1 APP.EYECODE

```
<?php
/*
  ___  _ __   ___ _   _  ___
 / _ \| '_ \ / _ \ | | |/ _ \
| (_) | | | |  __/ |_| |  __/
 \___/|_| |_|\___|\__, |\___|
                  |___/

oneye is released under the GNU Affero General Public License Version 3 (AGPL3)
 -> provided with this release in license.txt
 -> or via web at www.gnu.org/licenses/agpl-3.0.txt

Copyright © 2005 - 2010 eyeos Team (team@eyeos.org)
             since 2010 Lars Knickrehm (mail@lars-sh.de)
*/
global $config;
global $nextcloudURL;
global $nextcloudPath;
$config = eyeXML('getXMLconfig',array('nextcloud','conf.xml'));
$nextcloudURL = $config['nextcloud'][0]['url'][0];
$nextcloudPath = um('getCurrentUserDir') . 'files/Nextcloud' . '/';

include_once(EYE_ROOT.'/'.APP_DIR.'/nextcloud/func'.EYE_CODE_EXTENSION);

function Nextcloud_run($params = '') {
    if($params[0] == "sync") {
        fullsync($params);
    } else if ($params[0] == "file_events") {
        fileEvents($params);
    } else {
        $myWindow = new Window(array(
            'cent' => 1,
            'father' => 'eyeApps',
            'height' => 210,
            'name' => 'Nextcloud_Window',
            'title' => 'Nextcloud',
            'width' => 200
        ));
        $myWindow->show();

        $myLabelURL = new Label(array(
            'father' => 'Nextcloud_Window_Content',
            'name' => 'Nextcloud_URL_Label',
            'text' => 'Nextcloud URL:',
            'x' => 20,
            'y' => 20
        ));
        $myLabelURL->show(0);

        $myTextboxURL = new Textbox(array(
            'father' => 'Nextcloud_Window_Content',
            'name' => 'Nextcloud_URL_Textbox',
            'width' => 150,
            'x' => 20,
            'y' => 35
        ));
```

```
57          $myTextboxURL->show();
58
59          $myTextboxURL->focus();
60
61          $myLabelUser = new Label(array(
62              'father' => 'Nextcloud_Window_Content',
63              'name' => 'Nextcloud_User_Label',
64              'text' => 'Nextcloud User:',
65              'x' => 20,
66              'y' => 65
67          ));
68          $myLabelUser->show(0);
69          $myTextboxUser = new Textbox(array(
70              'father' => 'Nextcloud_Window_Content',
71              'name' => 'Nextcloud_User_Textbox',
72              'width' => 150,
73              'x' => 20,
74              'y' => 80
75          ));
76          $myTextboxUser->show();
77
78          $myLabelPassword = new Label(array(
79              'father' => 'Nextcloud_Window_Content',
80              'name' => 'Nextcloud_Password_Label',
81              'text' => 'Nextcloud Password:',
82              'x' => 20,
83              'y' => 110
84          ));
85          $myLabelPassword->show(0);
86          $myTextboxPassword = new Textbox(array(
87              'father' => 'Nextcloud_Window_Content',
88              'name' => 'Nextcloud_Password_Textbox',
89              'width' => 150,
90              'x' => 20,
91              'y' => 125,
92              'password' => 1
93          ));
94          $myTextboxPassword->show();
95
96          //
97          $myButton = new Button(array(
98              'caption' => 'Save',
99              'father' => 'Nextcloud_Window_Content',
100             'name' => 'Nextcloud_Save_Button',
101             'x' => 20,
102             'y' => 155
103         ));
104         $syncButton = new Button(array(
105             'caption' => 'Start Sync',
106             'father' => 'Nextcloud_Window_Content',
107             'name' => 'Nextcloud_Sync_Button',
108             'x' => 115,
109             'y' => 155
110         ));
111         // The Textbox need to be a friend of the Button:
112         // So every time the user clicks the Button the Textbox
113         // text will be sent.
```

```
114         // A button can have as many friends as you want.
115         $myButton->addFriend($myTextboxURL);
116         $myButton->addFriend($myTextboxUser);
117         $myButton->addFriend($myTextboxPassword);
118         $myButton->show();
119         $syncButton->show();
120         //load config data
121         if(!(empty(trim($GLOBALS['config']['nextcloud'][0]['url'][0])) &&
    empty(trim($GLOBALS['config']['nextcloud'][0]['username'][0])) &&
    empty(trim($GLOBALS['config']['nextcloud'][0]['password'][0])))) {
122             $GLOBALS['Nextcloud_URL_Textbox']->setText($GLOBALS['config']
    ['nextcloud'][0]['url'][0]);
123             $GLOBALS['Nextcloud_User_Textbox']->setText($GLOBALS['config']
    ['nextcloud'][0]['username'][0]);
124             $GLOBALS['Nextcloud_Password_Textbox']->setText($GLOBALS['config']
    ['nextcloud'][0]['password'][0]);
125         }
126     }
127
128 }
129
130 function Nextcloud_end($params = '') {
131     eyeWidgets('unserialize',$params);
132 }
133 ?>
```

6.2 EVENTS.EYECODE

```
<?php
/*
  ___  _ __   ___ _   _  ___
 / _ \| '_ \ / _ \ | | |/ _ \
| (_) | | | |  __/ |_| |  __/
 \___/|_| |_|\___|\__, |\___|
                  |___/

oneye is released under the GNU Affero General Public License Version 3 (AGPL3)
 -> provided with this release in license.txt
 -> or via web at www.gnu.org/licenses/agpl-3.0.txt

Copyright © 2005 - 2010 eyeos Team (team@eyeos.org)
            since 2010 Lars Knickrehm (mail@lars-sh.de)
*/

// First we create the NAMEOFAPP_on_NAMEOFBUTTON function.
// In this case, the application is Nextcloud and the
// button is called "Nextcloud_Button":
function Nextcloud_on_Nextcloud_Save_Button($params = '') {
    //get initial data
  $file = um('getCurrentUserDir').CONF_USER_DIR.'/nextcloud/conf.xml';
  //check if config exists
  if(!vfs('real_fileExists',array($file))) {
    //create config if not exists
    vfs('mkdir', array(um('getCurrentUserDir').CONF_USER_DIR.'/nextcloud'));
        vfs('real_create',array($file));
        $myF['nextcloud'] = array();
        eyeXML('setXMLfile',array($file,$myF));
    }
  //read config
    $content = eyeXML('getXMLconfig',array('nextcloud','conf.xml'));
  //set lastsync to never
  $lastSync = 0;
  if($content['nextcloud'][0]['lastSync'][0] != 0) {
    $lastSync = $content['nextcloud'][0]['lastSync'][0];
  }
  //set config attributes
  $content['nextcloud'][0]['url'][0] =
trim($GLOBALS['Nextcloud_URL_Textbox']->text);
    $content['nextcloud'][0]['username'][0] =
trim($GLOBALS['Nextcloud_User_Textbox']->text);
    $content['nextcloud'][0]['password'][0] =
trim($GLOBALS['Nextcloud_Password_Textbox']->text);
  $content['nextcloud'][0]['lastSync'][0] = $lastSync;
  //write config
  $myXml = eyeXML('array2xml',array($content));
    $fp = vfs('real_open',array($file,'w'));
    fwrite($fp,$myXml);
    fclose($fp);
  //subscribe to fileevents
  vfs('suscribeEvents',array("nextcloud","file_events"));
  //infrom user
  eyeX('messageBox',array('content' => 'Saved'));
}

function Nextcloud_on_Nextcloud_Sync_Button($params = '') {
```

```
  proc('launch',array('nextcloud',array("sync")));
}
// If the application we are developing needs to send / receive
// messages, it is necessary to update its contents through the
// eyeWidget's "updateContent" method.
// It is an automatic function and you can just copy it tp
// your applications, which use messages by simply changing the
// app's name in the function:
function Nextcloud_on_Message($params = '') {
    eyeWidgets('updateContent',$params);
}
// The NAMEOFAPP_on_Close() function will be executed every time
// the user closes the application. We recommended you to always send
// the close message to remove the application from the processes table.
function Nextcloud_on_Close($params = '') {
    proc('end');
}
?>
```

6.3 FUNC.EYECODE

```php
<?php
/*
  ___  _ __   ___ _   _  ___
 / _ \| '_ \ / _ \ | | |/ _ \
| (_) | | | |  __/ |_| |  __/
 \___/|_| |_|\___|\__, |\___|
                  |___/

oneye is released under the GNU Affero General Public License Version 3 (AGPL3)
 -> provided with this release in license.txt
 -> or via web at www.gnu.org/licenses/agpl-3.0.txt

Copyright © 2005 - 2010 eyeos Team (team@eyeos.org)
             since 2010 Lars Knickrehm (mail@lars-sh.de)
*/


/*****************************************************************************
//      Full Sync Function
*****************************************************************************/
function fullsync($params) {
        if(!(empty(trim($GLOBALS['config']['nextcloud'][0]['url'][0])) &&
empty(trim($GLOBALS['config']['nextcloud'][0]['username'][0])) && empty(trim($GLOB
['nextcloud'][0]['password'][0])))) {
            $lastLocalDataPath = um('getCurrentUserDir') . 'files/oldLocalData.jso
            vfs('removeEvents',array("nextcloud","file_events"));
            vfs('mkDir',array($GLOBALS['nextcloudPath']));
            //-------------------------- Step 1 ----------------------------

            //get file list locally
            $localData = dirToArray($GLOBALS['nextcloudPath']);

            //get last sync json
            $lastLocalData = json_decode(file_get_contents($lastLocalDataPath), tr
            if($lastLocalData) {
                //delete changes
                deleteChanges($localData, $lastLocalData);
            }

            //-------------------------- Step 2 ----------------------------
            //get file list from nextcloud
            $urlPath = parse_url($GLOBALS['nextcloudURL'], PHP_URL_PATH);
            $nextcloudSyncXML = simplexml_load_string(webDAVRequest("PROPFIND",
$GLOBALS['nextcloudURL']));
            //prepare array for comparison of changes
            $flattenArrayLastData = array();
            flattenArray($lastLocalData, $flattenArrayLastData);
            $flattenArrayActualData = array();
            flattenArray($localData, $flattenArrayActualData);
            $arrayDiff = array_diff($flattenArrayActualData, $flattenArrayLastData
            //sync new or modified files
            $namespaces = $nextcloudSyncXML->getNamespaces(true);
            foreach($nextcloudSyncXML->children($namespaces['d']) as $child) {
                $childHref = $child->children($namespaces['d'])->href;
                $cleanString = str_replace($urlPath, "", $childHref);
                $sURL = $GLOBALS['nextcloudURL'] . $cleanString;
```

```
54                $childLastModified =
$child->children($namespaces['d'])->propstat->prop->getlastmodified;
55                $timeLastModified = (DateTime::createFromFormat('D, d M Y H:i:s e'
$childLastModified))->getTimestamp();
56                $oneyesync = $child->children($namespaces['d'])->propstat->prop->or
57                if((!is_Null($oneyesync)) && $oneyesync > $timeLastModified) {
58                    $timeLastModified = $oneyesync;
59                }
60                if($cleanString != "") {
61                    $path = eyeFiles('cleanPath',array($GLOBALS['nextcloudPath']))
62                    $path[0] .= urldecode($cleanString);
63                    //check which files got deleted online
64                    $key = rtrim(urldecode($cleanString), "/");
65                    if(key_exists($key , $flattenArrayLastData)) {
66                        unset($flattenArrayLastData[$key]);
67                    }
68                    if( substr($childHref, -1) == "/"){
69                        //create folder
70                        vfs('mkDir',array(urldecode($path[0])));
71                    } else {
72                        //create file
73                        if(!vfs('fileExists', array($path[0]))) {
74                            if (vfs('create',array($path[0]))) {
75                                $contents = webDAVRequest("GET", $sURL);
76                                vfs('writeFile',array($path[0],$contents));
77                                webDAVRequest("PROPPATCH", $sURL, $path[0]);
78                            } else {
79                                //file creation failed
80                            }
81                        } else {
82                            //handling already existing files
83                            $file = getRealFileName($path[0]);
84                            $fileModifiedTime = filemtime($file . '.' . EYEOS_FILE_
85                            $fileInfo = vfs('readInfo', array($path[0]));
86                            if($timeLastModified > $fileModifiedTime && $fileModifi
$fileInfo['eyeFile'][0]['synced'][0] && $timeLastModified > $fileInfo['eyeFile']
[0]['synced'][0]) {
87                                //download modified file
88                                $contents = webDAVRequest("GET", $sURL);
89                                vfs('writeFile',array($path[0],$contents));
90                                //sync modified date
91                                updateVfs($path[0],array($timeLastModified, time()
92                            } else if ($timeLastModified < $fileModifiedTime &&
$fileModifiedTime > $fileInfo['eyeFile'][0]['synced'][0] && $timeLastModified <=
$fileInfo['eyeFile'][0]['synced'][0]) {
93                                //upload modified file
94                                webDAVRequest("PUT", $sURL, $path[0]);
95                                updateVfs($path[0],array($fileModifiedTime, time()
96                            } else if ($timeLastModified > $fileInfo['eyeFile'][0]
&& $fileModifiedTime > $fileInfo['eyeFile'][0]['synced'][0]) {
97                                //handle conflics
98                                $newPath = substr_replace($path[0], "_conflict",
strrpos($path[0], '.'), 0 );
99                                $newURL = substr_replace($sURL, "_conflict", strrpo
'.'), 0 );
100                               if($timeLastModified > $fileModifiedTime) {
101                                   //move local file to x_conflict
102                                   vfs('copy',array( $path[0], $newPath));
```

```
                                    //download newer file
                                    vfs('writeFile',array($path[0],webDAVRequest("(
$sURL)));
                                    updateVfs($path[0],array($timeLastModified, tir
                                    //upload local file
                                    webDAVRequest("PUT", $newURL, $newPath);
                                } else if ($timeLastModified < $fileModifiedTime) {
                                    //rename online file
                                    webDAVRequest("MOVE", $sURL, "", $newURL);
                                    //upload local file
                                    webDAVRequest("PUT", $sURL, $path[0]);
                                    //download online file
                                    if (vfs('create',array($newPath))) {
                                        vfs('writeFile',array($newPath,webDAVReques
$newURL));
                                        //sync modified date
                                        updateVfs($newPath,array($timeLastModified
time()));
                                    }
                                }
                            }

                        }
                    }
                }
            }
            //delete online removed files
            foreach ($flattenArrayLastData as $key => $value) {
                    $path = $GLOBALS['nextcloudPath'].$value;
                    if(vfs('isdir',array($path))) {
                        vfs('rmdir',array($path));
                    } else {
                        vfs('delete',array($path));
                    }
            }
            //create local added files
            uasort($arrayDiff, function($a, $b) {
                return strlen($a) <=> strlen($b);
            });
            foreach ($arrayDiff as $key => $value) {
                if(count($nextcloudSyncXML->xpath('d:response/d:href[.="'.$urlPath
/parent::*')) == 0) {
                    $path = $GLOBALS['nextcloudPath'].$value;
                    if(vfs('isdir',array($path))) {
                        $sURL = $GLOBALS['nextcloudURL'] . $value;
                        webDAVRequest("MKCOL", $sURL);
                    } else {
                        $sURL = $GLOBALS['nextcloudURL'] . $value;
                        webDAVRequest("PUT", $sURL, $path);
                    }
                }
            }
            //update ui stuff
            if($params[1] != null){
                eyeFiles('update',array($params[1]));
            } else {
                eyeFiles('update',array('home:///'));
```

```
            }
            //save local list xml
            vfs('real_create',array($lastLocalDataPath));
            //get file list locally
            $localData = dirToArray($GLOBALS['nextcloudPath']);
            vfs('real_putFileContent',array($lastLocalDataPath,json_encode($localDa
            //set last sync time
            $GLOBALS['config']['nextcloud'][0]['lastSync'][0] = time();
            eyeXML('setXMLconfig', array('nextcloud', 'conf.xml', $GLOBALS['config
            //inform user about finished sync
            eyeX('messageBox',array('content' => 'Nextcloud Sync finished!'));
            vfs('suscribeEvents',array("nextcloud","file_events"));
        } else {
            eyeX('messageBox',array('content' => 'Nextcloud Sync is not configured
        }
        proc('end');
}

/*****************************************************************************
//      FileEvent Function
/*****************************************************************************

function fileEvents($params) {
    if(!(empty(trim($GLOBALS['config']['nextcloud'][0]['url'][0])) &&
empty(trim($GLOBALS['config']['nextcloud'][0]['username'][0])) && empty(trim($GLOBA
['nextcloud'][0]['password'][0])))) {
        switch ($params[1][0]) {
            case 'create':
                if (str_contains($params[1][1][0], $GLOBALS['nextcloudPath'])) {
                    $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextclo
$params[1][1][0]);
                    webDAVRequest("PUT", $url, $params[1][1][0]);
                    webDAVRequest("PROPPATCH", $url, $params[1][1][0]);
                }
                break;
            case 'writeFile':
                if (str_contains($params[1][1][0], $GLOBALS['nextcloudPath'])) {
                    $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextclo
$params[1][1][0]);
                    webDAVRequest("PUT", $url, $params[1][1][0]);
                    webDAVRequest("PROPPATCH", $url, $params[1][1][0]);
                }
                break;
            case 'restore':
                if (str_contains($params[1][1][1], $GLOBALS['nextcloudPath'])) {
                    $filePath = vfs('getVirtualName',array($params[1][1][1]));
                    $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextclo
$filePath);
                    $path = $filePath.$value;
                    webDAVRequest("PUT", $url, $path);
                    webDAVRequest("PROPPATCH", $url, $path);
                }
                break;
            case 'restoreDir':
                if (str_contains($params[1][1][1], $GLOBALS['nextcloudPath'])) {
                    $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextclo
$params[1][1][1]);
```

```
207                     webDAVRequest("MKCOL", $url);
208                     $content = dirToArray($params[1][1][1]);
209                     $flatArray = array();
210                     flattenArray($content, $flatArray);
211                     foreach ($flatArray as $key => $value) {
212                         $path = $params[1][1][1].$value;
213                         $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nex
$path);
214                         if(vfs('isdir',array($path))) {
215                             webDAVRequest("MKCOL", $url);
216                         } else {
217                             webDAVRequest("PUT", $url, $path);
218                             webDAVRequest("PROPPATCH", $url, $path);
219                         }
220                     }
221                 }
222                 break;
223             case 'deleteTrash':
224                 if (str_contains($params[1][1][0], $GLOBALS['nextcloudPath'])) {
225                     $filename = vfs('getVirtualName',array($params[1][1][0]));
226                     $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextcl
$filename);
227                     webDAVRequest("DELETE", $url);
228                 }
229             case 'delete':
230                 # handelt by other event
231                 break;
232             case 'erase':
233                 # handelt by other event
234                 break;
235             case 'copy':
236                 if (str_contains($params[1][1][1], $GLOBALS['nextcloudPath'])) {
237                     $filenameFrom = vfs('getVirtualName',array($params[1][1][0]));
238                     $filenameTo = vfs('getVirtualName',array($params[1][1][1]));
239                     $urlFrom = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nex
$filenameFrom);
240                     $urlTo = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['next
$filenameTo);
241                     webDAVRequest("COPY", $urlFrom, "", $urlTo);
242                     webDAVRequest("PROPPATCH", $urlTo, $filenameTo);
243                 }
244                 break;
245             case 'copyDir':
246                 if (str_contains($params[1][1][1], $GLOBALS['nextcloudPath'])) {
247                     $urlFrom = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nex
$params[1][1][0]);
248                     $urlTo = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['next
$params[1][1][1]);
249                     webDAVRequest("COPY", $urlFrom, "", $urlTo);
250                 }
251                 break;
252             case 'move':
253                 //used as move for files
254                 if (str_contains($params[1][1][1], $GLOBALS['nextcloudPath']) &&
str_contains($params[1][1][0], $GLOBALS['nextcloudPath'])) {
255                     $filenameFrom = vfs('getVirtualName',array($params[1][1][0]));
256                     $filenameTo = vfs('getVirtualName',array($params[1][1][1]));
```

```
257                     $urlFrom = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nex
$filenameFrom);
258                     $urlTo = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['next
$filenameTo);
259                     webDAVRequest("MOVE", $urlFrom, "", $urlTo);
260                 } else if (str_contains($params[1][1][1], $GLOBALS['nextcloudPath'
!str_contains($params[1][1][0], $GLOBALS['nextcloudPath'])) {
261                     //moved in nextcloud folder
262                     $filename = vfs('getVirtualName',array($params[1][1][1]));
263                     $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextcl
$filename);
264                     webDAVRequest("PUT", $url, $filename);
265                     webDAVRequest("PROPPATCH", $url, $filename);
266                 } else if (!str_contains($params[1][1][1], $GLOBALS['nextcloudPath
str_contains($params[1][1][0], $GLOBALS['nextcloudPath'])) {
267                     //moved out of nextcloud folder
268                     $filename = vfs('getVirtualName',array($params[1][1][0]));
269                     $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextcl
$filename);
270                     webDAVRequest("DELETE", $url);
271                 }
272                 break;
273             case 'rename':
274                 //used as move for folder
275                 if (str_contains($params[1][1][1], $GLOBALS['nextcloudPath']) &&
str_contains($params[1][1][0], $GLOBALS['nextcloudPath'])) {
276                     $urlFrom = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nex
$params[1][1][0]);
277                     $urlTo = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['next
$params[1][1][1]);
278                     webDAVRequest("MOVE", $urlFrom, "", $urlTo);
279                 } else if ((str_contains($params[1][1][1], $GLOBALS['nextcloudPath
(!str_contains($params[1][1][0], $GLOBALS['nextcloudPath']))) {
280                     //moved in nextcloud folder
281                     $dir = $params[1][1][1]."/";
282                     $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextcl
$dir);
283                     webDAVRequest("MKCOL", $url);
284                     $content = dirToArray($dir);
285                     $flatArray = array();
286                     flattenArray($content, $flatArray);
287                     foreach ($flatArray as $key => $value) {
288                         $path = $dir.$value;
289                         $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nex
$path);
290                         if(vfs('isdir',array($path))) {
291                             webDAVRequest("MKCOL", $url);
292                         } else {
293                             webDAVRequest("PUT", $url, $path);
294                             webDAVRequest("PROPPATCH", $url, $path);
295                         }
296                     }
297                 } else if ((!str_contains($params[1][1][1], $GLOBALS['nextcloudPath
(str_contains($params[1][1][0], $GLOBALS['nextcloudPath']))) {
298                     //moved out of nextcloud folder
299                     $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextcl
$params[1][1][0]);
300                     webDAVRequest("DELETE", $url);
301                 }
```

```
302                 break;
303             case 'mkdir':
304                 if (str_contains($params[1][1][0], $GLOBALS['nextcloudPath'])) {
305                     $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextcl
$params[1][1][0]);
306                     webDAVRequest("MKCOL", $url);
307                 }
308                 break;
309             case 'rmdir':
310                 if (str_contains($params[1][1][0], $GLOBALS['nextcloudPath'])) {
311                     $url = str_replace($GLOBALS['nextcloudPath'], $GLOBALS['nextcl
$params[1][1][0]);
312                     webDAVRequest("DELETE", $url);
313                 }
314                 break;
315 
316             default:
317                 # nothing to do without known event
318                 break;
319         }
320     }
321     proc('end');
322 }
323 /*****************************************************************************
324 //      Helper Functions
325 /*****************************************************************************
326 
327 function updateVfs($filePath, $params) {
328     $modified = $params[0];
329     $synced = $params[1];
330     $creationapp = $params[2];
331 
332     //Getting the xml information
333     $fileInfo = vfs('readInfo', array($filePath));
334     $fileInfoChanged = false;
335     if($creationapp) {
336         $fileInfo['eyeFile'][0]['creationapp'][0] = $creationapp;
337         $fileInfoChanged = true;
338     }
339     if($synced) {
340         $fileInfo['eyeFile'][0]['synced'][0] = $synced;
341         $fileInfoChanged = true;
342     }
343     //Creating the xml information
344     if($fileInfoChanged) {
345         $file = getRealFileName($filePath);
346         file_put_contents($file . '.' . EYEOS_INFO_EXT, eyeXML('array2xml',array($
347     }
348     if($modified) {
349         touch($file . '.' . EYEOS_FILE_EXT, $modified);
350     }
351 }
352 
353 function dirToArray($dir) {
354     $result = array();
355     $cdir = vfs('getDirContent',array($dir));
356     foreach ($cdir as $value)
```

```
357     {
358         if (!in_array($value,array(".","..")))
359         {
360             $returnVal = substr($value, strrpos($value, '/') + 1);
361             if (vfs('isdir',array($value)))
362             {
363                 $result[$returnVal] = dirToArray($value);
364             } else {
365                 $result[$returnVal] = $returnVal;
366             }
367         }
368     }
369     return $result;
370 }
371
372 function flattenArray($array, &$returnVal, $parent = "") {
373     if($parent !== "") {
374         $parent = $parent."/";
375     }
376     foreach ($array as $key => $value) {
377         if(is_array($value)){
378             $returnVal[$parent.$key] = $parent.$key;
379             flattenArray($value, $returnVal, $parent.$key);
380         } else {
381             $returnVal[$parent.$value] = $parent.$value;
382         }
383     }
384 }
385
386 function deleteChanges($arrayToCompare, $arrayToIterate, $parent = "") {
387     foreach ($arrayToIterate as $key => $value) {
388         $returnVal = true;
389         if($parent !== "") {$searchPath = $parent."/".$key;}else{$searchPath=$key;}
390         checkArray($arrayToCompare, $searchPath, $returnVal);
391         if($returnVal) {
392             if(is_Array($value)) {
393                 deleteChanges($arrayToCompare, $value, $searchPath);
394             }
395         } else {
396                 //delete path
397                 $sURL = $GLOBALS['nextcloudURL'].$parent."/".(string)$key;
398                 webDAVRequest("DELETE", $sURL);
399         }
400     }
401 }
402
403
404 function checkArray($array, $searchPath, &$returnVal) {
405     $searchPath = rtrim($searchPath, "/");
406     if (strpos($searchPath, '/') !== false) {
407         $pathToSearch = explode("/", $searchPath);
408     } else {
409         $pathToSearch = array($searchPath);
410     }
411     if(array_key_exists($pathToSearch[0], $array)) {
412     //remove the first element of input
413         $array = $array[$pathToSearch[0]];
```

```
414         array_shift($pathToSearch);
415         if(empty($pathToSearch)) {
416             $returnVal = true;
417         } else {
418             checkArray($array, implode("/", $pathToSearch), $returnVal);
419         }
420     } else {
421         $returnVal = false;
422     }
423 }
424 
425 function webDAVRequest($method, $url, $file = "",  $destination = "") {
426     $nextcloudUserData = base64_encode($GLOBALS['config']['nextcloud'][0]['username
[0].":".$GLOBALS['config']['nextcloud'][0]['password'][0]);
427     if($destination == "") {
428         $header = "Authorization: Basic " . $nextcloudUserData . "\r\n";
429     } else {
430         $header = "Authorization: Basic " . $nextcloudUserData . "\r\n"."Destinatio
".$destination."\r\n"."Depth: infinity\r\n";
431     }
432     if($method == "PROPFIND") {
433         $content = '<?xml version="1.0" encoding="utf-8" ?>
434         <D:propfind xmlns:D="DAV:">
435             <D:prop>
436                 <D:oneyesync/>
437                 <D:getlastmodified/>
438             </D:prop>
439         </D:propfind>';
440         $header = "Authorization: Basic " . $nextcloudUserData . "\r\n"."Depth: in
441     }
442     if($method == "PROPPATCH") {
443         $fileModifiedTime = filemtime(getRealFileName($file) . '.' . EYEOS_FILE_EX
444         $content = '<?xml version="1.0" encoding="utf-8" ?>
445         <d:propertyupdate xmlns:d="DAV:">
446         <d:set>
447             <d:prop>
448                 <d:oneyesync>' . $fileModifiedTime . '</d:oneyesync>
449             </d:prop>
450         </d:set>
451         </d:propertyupdate>';
452         $header = "Authorization: Basic " . $nextcloudUserData . "\r\n"."Content-ty
application/xml\r\n";
453         updateVfs($file,array(null, $fileModifiedTime));
454     }
455     if($method == "PUT") {
456         $fp = vfs('open',array($file,'r'));
457         $size = vfs('filesize',array($file));
458         $content = fread($fp,$size);
459         fclose($fp);
460     }
461     $opts = array(
462             'http'=>array(
463                 'method'=>$method,
464                 'header'=>$header,
465                 'content'=>$content
466             )
467     );
```

```
    $context = stream_context_create($opts);
    $return = file_get_contents($url, false, $context);
    return $return;
}

function str_contains($haystack, $needle) {
    return $needle !== '' && mb_strpos($haystack, $needle) !== false;
}
?>
```

6.4 FUNKTIONEN ZUR ERWEITERUNG DES VFS

```
2448 /**
2449  * Trigger all apps which subscribed to file system events.Subscription handled
     by shared memory segment. Application need to register Name and Parameter to call
     on event.
2450  *
2451  *
2452  * @param $params array(method, affectedFile/affectedDir)
2453  * @return Does not return
2454  * @date 2021/10/27
2455  */
2456 function service_vfs_events($params) {
2457     if (is_array($params) || count($params) < 1) {
2458         $method = $params[0];
2459         $affectedFile = $params[1];
2460         $saveLocation = um('getCurrentUserDir') . '/' .
     'files/subscribeFileEvents.json';
2461         $shm = shm_attach(29654);
2462         if(!shm_has_var($shm, 1)) {
2463             $subscribedApps = json_decode(file_get_contents($saveLocation),
     true);
2464             shm_put_var($shm, 1, $subscribedApps);
2465         }
2466         $subscribedApps = shm_get_var($shm, 1);
2467         if(!is_null($subscribedApps)){
2468             foreach ($subscribedApps as $sub) {
2469                 proc('launch',array($sub[0],array($sub[1],array($method,
     $affectedFile))));
2470             }
2471         }
2472     }
2473 }
2474
2475 /**
2476  * Subscribe app to file events
2477  *
2478  * @param $params array(name, parameter)
2479  * @return Does not return
2480  * @date 2021/10/27
2481  */
2482 function service_vfs_suscribeEvents($params) {
2483     $shm = shm_attach(29654);
2484     if(!is_array($params) || count($params) < 2){
2485         shm_remove($shm);
2486         $shm = shm_attach(29654);
2487     }
2488     $saveLocation = um('getCurrentUserDir') . '/' .
     'files/subscribeFileEvents.json';
2489     if(shm_has_var($shm, 1)) {
2490         $oldVal = shm_get_var($shm, 1);
2491         if (!in_array(array($params[0], $params[1]), $oldVal)) {
2492             if(!(is_array($oldVal))) {
2493                 $oldVal = array();
2494             }
2495             array_push($oldVal, array($params[0], $params[1]));
2496             error_log("test:");
2497             error_log(print_r($oldVal,TRUE));
2498             error_log(print_r(array($params[0], $params[1]),TRUE));
```

```
2499            shm_put_var($shm, 1, $oldVal);
2500            vfs('real_putFileContent',array($saveLocation,json_encode($oldVal)));
2501        }
2502    } else {
2503        if(file_exists($saveLocation)) {
2504            $val = json_decode(file_get_contents($saveLocation), true);
2505            shm_put_var($shm, 1, $val);
2506        } else {
2507            $val[] = array($params[0], $params[1]);
2508            shm_put_var($shm, 1, $val);
2509            vfs('real_putFileContent',array($saveLocation,json_encode($val)));
2510        }
2511    }
2512    shm_detach($shm);
2513}
2514
2515/**
2516 * Remove app from file events
2517 *
2518 * @param $params array(name, parameter)
2519 * @return Does not return
2520 * @date 2021/10/27
2521 */
2522function service_vfs_removeEvents($params) {
2523    $shm = shm_attach(29654);
2524    if($params[2]){
2525        shm_remove($shm);
2526        $shm = shm_attach(29654);
2527    }
2528    $saveLocation = um('getCurrentUserDir') . '/' .
'files/subscribeFileEvents.json';
2529    if(shm_has_var($shm, 1)) {
2530        $oldVal = shm_get_var($shm, 1);
2531        $keysToRemove = array_keys($oldVal, array($params[0], $params[1]));
2532        foreach ($keysToRemove as $key => $value) {
2533            unset($oldVal[$value]);
2534        }
2535        shm_put_var($shm, 1, $oldVal);
2536        vfs('real_putFileContent',array($saveLocation,json_encode($oldVal)));
2537    } else {
2538        if(file_exists($saveLocation)) {
2539            $val = json_decode(file_get_contents($saveLocation), true);
2540            $keysToRemove = array_keys($val, array($params[0], $params[1]));
2541            foreach ($keysToRemove as $key => $value) {
2542                unset($val[$value]);
2543            }
2544            vfs('real_putFileContent',array($saveLocation,json_encode($val)));
2545            shm_put_var($shm, 1, $val);
2546        }
2547    }
2548    shm_detach($shm);
2549}
```

6.5 DOCKER-COMPOSE.YML NEXTCLOUD DOCKER

```yaml
version: "2"

volumes:
  nextcloud:
  db:

services:
  db:
    image: mariadb
    restart: always
    command: --transaction-isolation=READ-COMMITTED --binlog-format=ROW
    volumes:
      - db:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=somepassword
      - MYSQL_PASSWORD=somepassword
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud

  app:
    image: nextcloud
    restart: always
    ports:
      - 8080:80
    links:
      - db
    volumes:
      - nextcloud:/var/www/html
    environment:
      - MYSQL_PASSWORD=somepassword
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
      - MYSQL_HOST=db
```

LITERATURVERZEICHNIS

[1] statista, „statista,“ Februar 2021. [Online]. Available: https://de.statista.com/statistik/daten/studie/1204173/umfrage/befragung-zur-homeoffice-nutzung-in-der-corona-pandemie/. [Zugriff am 07 November 2021].

[2] S. Grüner, „Nextcloud ist der "Neustart" für Owncloud,“ 2 Juni 2016. [Online]. Available: https://www.golem.de/news/community-fork-und-neues-unternehmen-nextcloud-ist-der-neustart-fuer-owncloud-1606-121243.html. [Zugriff am 7 November 2021].

[3] Nextcloud GmbH, „How Nextcloud keeps your Data secure,“ [Online]. Available: https://nextcloud.com/secure/. [Zugriff am 7 November 2021].

[4] Nextcloud GmbH, „Comparison,“ [Online]. Available: https://nextcloud.com/compare/. [Zugriff am 7 November 2021].

[5] oneye Team, „oneye project,“ [Online]. Available: https://oneye-project.org/. [Zugriff am 9 November 2021].

[6] oneye Team, „oneye/oneye,“ 2021. [Online]. Available: https://github.com/oneye/oneye. [Zugriff am 9 November 2021].

[7] R. Vogel, T. Koçoğlu und T. Berger, in *Desktopvirtualisierung*, Wiesbaden, Vieweg +Teubner Verlag, 2010, pp. 7-9.

[8] RedHat, „What is a hypervisor?,“ 10 January 2020. [Online]. Available: https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor. [Zugriff am 9 November 2021].

[9] Institute for Advanced Study, „RemoteApp,“ [Online]. Available: https://www.ias.edu/itg/RemoteApp. [Zugriff am 11 November 2021].

[10] P. von Oven, in *Delivering Applications with VMware App Volumes 4*, Springer Science+Business Media , New York, 2021, pp. 2-4.

[11] Microsoft, „Visual Studio Code - Code editing. Redefined,“ [Online]. Available: https://code.visualstudio.com/. [Zugriff am 9 November 2021].

[12] The PHP Group, „History of PHP,“ [Online]. Available: https://www.php.net/manual/en/history.php.php. [Zugriff am 9 November 2021].

[13] R. Morschhauser, „Vorteile von PHP,“ 3 Februar 2002. [Online]. Available: https://www.mathematik.uni-ulm.de/sai/ws01/portalsem/rene/internetportale/node26.html. [Zugriff am 9 November 2021].

[14] J. Friesen, in *Java XML and JSON*, Berkeley, Apress, 2016, pp. 133-134.

[15] „Einführung in JSON,“ [Online]. Available: http://www.json.org/json-de.html. [Zugriff am 9 11 2021].

[16] M. Schulz, S. Henneberger und S. Dobratz, „Introduction to XML for ETDs,“ 20 Mai 2003. [Online]. Available: https://edoc.hu-berlin.de/bitstream/handle/18452/1823/index.pdf. [Zugriff am 9 November 2021].

[17] Refsnes Data, „XML DTD,“ [Online]. Available: https://www.w3schools.com/xml/xml_dtd.asp. [Zugriff am 9 November 2021].

[18] Docker Inc, „Docker CE for Linux installation script,“ [Online]. Available: https://get.docker.com/. [Zugriff am 9 11 2021].

[19] VMware, „XAMPP,“ [Online]. Available: https://www.apachefriends.org/index.html. [Zugriff am 9 November 2021].

[20] The IETF Trust, „HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV),“ Juni 2007. [Online]. Available: http://www.webdav.org/specs/rfc4918.html. [Zugriff am 9 November 2021].

[21] Nextcloud GmbH, „OCS APIs overview,“ 2021. [Online]. Available: https://docs.nextcloud.com/server/latest/developer_manual/client_apis/OCS/ocs-api-overview.html. [Zugriff am 9 November 2021].

[22] The PHP Group, „Variable scope,“ [Online]. Available: https://www.php.net/manual/en/language.variables.scope.php. [Zugriff am 9 November 2021].

[23] „Nextcloud - Offical Image,“ [Online]. Available: https://hub.docker.com/_/nextcloud. [Zugriff am 9 November 2021].