Berufsbildende Schulen Meppen

Berufliches Gymnasium Technik

Jahrgangsstufe 12

Abiturjahrgang 2015

Projektarbeit

im Fach Informationsverarbeitung

Thema: **Das virtuelle, browserbasierte Betriebssystem SpercOS**

Verfasser: Jonathan Lusky

Kursleiter: Herr Mahler

Bearbeitungszeit: Qualifikationsphase – 2. Kurshalbjahr

Abgabetermin: 16. Mai 2014

Benotung: _____ Punkte

_____

Unterschrift des Kursleiters

Inhaltsverzeichnis

1 Vorwort

Diese Projektarbeit beschäftigt sich mit der Beschreibung, Analyse und Bewertung des virtuellen, browserbasierten Betriebssystems „SpercOS“. Es ist vorweg wichtig zu betonen, dass diese Software bei Erstellung dieser Arbeit noch nicht fertiggestellt worden ist und daher die Beschreibungen sowie die Anhänge im folgenden Text möglicherweise von der aktuellsten Version abweichen.

2 Einleitung

Die erste Frage die sich aufdrängt, wenn man den Titel dieser Facharbeit liest ist wahrscheinlich, wie es überhaupt möglich ist, dass ein Betriebssystem im Browser ausgeführt werden kann. Diese Art Betriebssystem ist durchaus möglich, allerdings nehme ich vorweg, dass das hier beschriebene System nicht den Anforderungen eines Betriebssystems entspricht. Vielmehr ist es wie ein Allround – Content – Management – System zu sehen. Die Anforderungen, die vor und während der Entwicklungsphase festgelegt und ständig bestätigt wurden, waren, dass „SpercOS“ kommerziell nutzlos sein soll und es nicht einmal für den produktiven Einsatz geeignet sein muss. Außerdem durfte die Entwicklung keine zusätzlichen Kosten oder extraprivate Leistungen verursachen.

Es gibt mehrere Projekte, die „SpercOS“ auf bestimmte Art und Weise beeinflusst haben. Der Grundgedanke ein Betriebssystem über den Browser laufen zu lassen kam durch die Internetseite www.windoof.org, auf welcher Entwickler die Oberfläche von Microsofts Windows 7 umgesetzt haben. Das Onlineforensystem „Simple Machines Forums“, bei welchem ich seit 2012 ehrenamtlicher Mitentwickler bin, enthält seit Version 2 ein umfangreiches Modifikations- und Updatemodul (Anonym 1). Die theoretischen Grundlagen dieses Moduls wurden ebenfalls in „SpercOS“ implementiert.

Der Name „SpercOS“ ist zum größten Teil willkürlich und hat keine tiefere Bedeutung. Lediglich der letzte Teil „OS“ ist eine Abkürzung für „Operating System“ und steht für „Betriebssystem“.

3 Definition Betriebssystem

Es gibt viele verschiedene offiziellen Definitionen von Betriebssystemen, wobei manche davon schon aus den späten Anfängen der digitalen Informationsverarbeitung kommen. Die Differenzen bestehen hauptsächlich in der Frage, ob das Betriebssystem „zum Betrieb des Computers erforderlich“ (Grieser und Irlbeck) ist und in der Frage, ob das Betriebssystem eine Sammlung von Softwarekomponenten (S. Meiler; Uni Leipzig) oder eine einzige Software ist (H. Balzert, S. 30).

Ob das BS wirklich zum Betrieb eines Computers benötigt wird, lässt sich wohl nicht klären, wenn nicht einmal klar ist, was alles als Computer definitionell durchgeht und was nicht. Den Definitionen aus zuverlässigen Quellen (einschlägigen Wörterbüchern wie Duden und itwissen.info (K. Lipinski)) zufolge, ist ein Computer lediglich ein Gerät, welches Daten nach einem bestimmtem Muster verarbeitet. Dabei ist bewusst nicht erwähnt worden, dass dieses Gerät notwendigerweise elektronisch betrieben werden muss. Das kommt daher, dass auch schon vor der elektronischen Datenverarbeitung Theoretiker mit analogen, nicht elektrischen Gerätschaften[1] Daten verarbeitet haben. Eine Validierung der Aussage, dass das Betriebssystem wirklich zur Verwendung eines Computers notwendig ist, ist somit nicht möglich.

Die zweite Frage kann geklärt werden, wenn man sich aktuellere Betriebssysteme anguckt, in diesem Beispiel Microsofts Windows NT. Man stellt schnell fest, dass es sich dabei tatsächlich nicht um eine einzelne Softwarekomponente handelt, sondern das BS aus vielen Softwares besteht, die alle eine bestimmte Funktion erfüllen. Viele davon werden direkt mit Start des Computers gestartet, ob im Hintergrund (etwa Netzwerkdienste) oder im Vordergrund (wie beispielsweise der Explorer). Manche werden vom BS erst während des Betriebes, etwa nach Ablauf einiger Zeit oder bei Bedarf gestartet (Microsoft). Es ist aber auch durchaus möglich, dass das Betriebssystem nur aus einer Software besteht[2], die alle erforderlichen Aufgaben

1 Gerätschaften wie Rechenschieber oder Kurbelrechenmaschiene

2 Siehe Arduino

ausreichend erfüllt. Diese Betriebssysteme haben aber zumeist keine eigene grafische Oberfläche für Benutzer, sondern können lediglich durch Texteingaben, häufig sogar nur über analoge oder serielle Kommunikation bedient werden. Für viele Menschen mag diese Art sehr ungewöhnlich klingen und es fällt auf Anhieb möglicherweise kein reales Einsatzszenario ein, aber tatsächlich ist das wohl die am Häufigsten vorkommende Sorte. Fast alle Steuerplatinen und Prototyp Plattformen in der Industrie und im Privatwesen haben solche Betriebssysteme, die in manchen (aber eher selteneren Fällen) sogar keine gesonderten Treiber enthalten (Anonym 2).

Vereinfacht ist ein reales Betriebssystem fast immer so aufgebaut: Es gibt einen Kernel, der mittels Treiber direkt mit der Hardware kommuniziert und eindeutige Befehle an die Computerkomponenten erteilt und die Rückgabewerte dieser Komponenten erhält. Damit Programmierer ihre Programme einheitlich gestalten und kompatibel nutzbar machen können, ohne jedes mal aufs neue die Grundlagen der Computerinformationsverarbeitung zu bedenken, stellt der Kernel außerdem eine vereinfachte Programmierumgebung (API = Application programming interface) zur Verfügung. Dadurch kann der Programmierer direkt mit dem Kernel (Oder einer Vorstufe zum Kernel) kommunizieren, und ihr einen Befehl geben (M. Saifullah). Praktisch könnte das so aussehen, dass der Programmierer dem API sagt, dass er ein bisschen Arbeitsspeicher für einen Textbuchstaben (char) benötigt und dieser bitte unter der Variable „textbuchstabe“ abrufbar ist, und der Kernel sucht nach freiem Arbeitsspeicher (im Normalfall), fügt dort den Variablentitel ein und reserviert den benötigten Platz, adressiert dies und stellt diese Adresse dem Programmierer wieder zur Verfügung. Auf einer höheren Ebene kann es sein, dass der Programmierer dann nur noch sagt, dass für seine Anwendung ein Fensterrahmen braucht, wodurch sehr große Mengen an Informationen, bereitgestellt vom Betriebssystem, gespeichert und verarbeitet werden müssen (Anonym 3).

3.1 Differenzen der Definition und SpercOS

SpercOS ist, wie bereits vorher beschrieben, kein System, welches den Anforderungen eines Betriebssystems gerecht werden würde. Das zeigt sich am Einfachstem daran, dass es keine Hardwareebene gibt, auf die direkt zugegriffen werden kann. Statt Daten über einen Treiber in das Dateisystem zu schreiben, wird bei SpercOS das App API beansprucht, das ein Dateisystem simuliert. Dieses Vorgehen ist auch notwendig, da die einzelnen Applikationen selber keine Tabellen in der Datenbank anlegen dürfen[3]. Es wird also von der Software kontrolliert, was, wer, wann und wo schreibt. Außerdem kann sie diese Daten arrangieren, indem sie sie einheitlich filtern, sortieren und abändern kann. Dadurch kann der Benutzer über dieses Datenbankprinzip theoretisch eine Applikation auswählen, in welcher ein beliebiger Datensatz eingelesen und verarbeitet werden soll. Natürlich würde eine sinngemäße Anwendung dessen voraussetzen, dass der Datensatz mit der Applikation kompatibel ist (Die Möglichkeit einer Auswahl der App zu einem Datensatz ist in SpercOS nicht enthalten und aus praktischen Gründen nicht vorgesehen).

Das hier beschriebene System ist auch nicht in Ebenen aufeinander aufgebaut, wie es bei den großen Vorbildern der Fall ist. Vielmehr laufen alle Ebenen nebeneinander. Damit ist gemeint, dass sie sich nicht für die korrekte Eigenfunktion gegenseitig benötigen: Es lassen sich Funktionen aus der „functions_inc.php“[4] nutzlos ausführen und man würde ein korrektes Ergebnis bekommen, obwohl man nicht unbedingt die „functions_inc.php“ in eine Startdatei[5] eingebunden hat, oder es ist sogar möglich eine App zu starten, ohne das System im Browser zuvor aufzurufen. Anders wiederum ist es, wenn der Benutzer sich standardkonform verhält: Alle Dateien und Funktionen werden sequentiell eingebunden, ausgelöst von einer Datei, die selbst nur drei von den 14 für den Betrieb essentiellen Dateien einbindet.

---

3 Siehe Implementierung → Datenbank

4 Siehe Implementierung → Allgemein

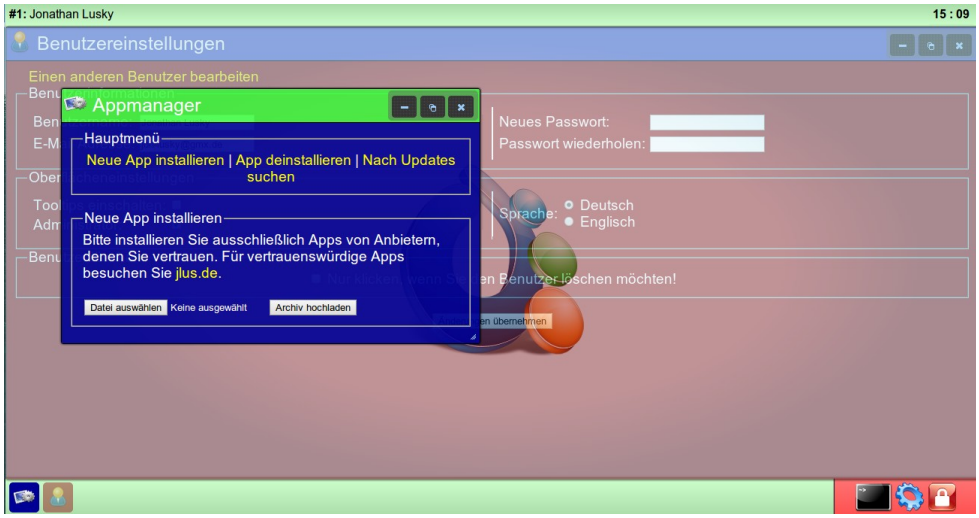
5 Siehe Implementierung → Allgemeine Sicherheitsvorkehrungen

Da SpercOS nicht direkt auf Hardware zugreifen kann, sind prinzipiell keine notwendig. Allerdings wurde bei der Entwicklung sehr viel Javascript verwendet, welches verschiedene Browser verschieden interpretieren. Um sich zusätzliche Arbeit bei der Anpassung an alle Browser zu sparen, wurde das weit verbreitete Javascript Framework „jQuery“ verwendet. „jQuery“ stellt einerseits grundlegende Funktionen[6] zur Verfügung, dessen Anwedungssimplizität und Funktionsumfang die üblichen Javascript-Routinen übertreffen. Andererseits bietet „jQuery“, was es bei „SpercOS“ vergleichbar mit einem Treiber (bzw. generell einer tieferen Ebene) macht, eine Programmierschnittstelle, die kompatibel zu allen gängigen, aktuellen Browser ist (the jQuery Foundation).

4 Funktionalitäten

4.1 Benutzer

4.1.1 Graphical User Interface



Das „Graphical User Interface“ (dt. „Grafische Benutzeroberfläche“) ist eine Bedienoberfläche, mit welcher selbst unerfahrene Benutzer die Möglichkeit haben das Betriebssystem mit seinen Funktionalitäten zu nutzen. Die bei „SpercOS“ verwendete Oberfläche ist außerhalb der vorgesehenen dynamischen Inhalten nur durch konkrete Codemodifikation auf der Ebene der Setupcall Funktionen möglich. Die Hauptelemente der GUI sind die

6 Wie beispielsweise die DOM-Selektions-Funktion oder die Verwendung von AJAX

Kopf- und die Fußzeile, in welchen sich Systeminformationen, Hinweise, Menüs und eine Taskleiste befinden, die Fensterrahmen und ihre Inhalte, sowie die Desktopfläche mit den (vorher eingestellten) Starticons.

Die GUI ist an die von Betriebssystemen wie Microsofts Windows angelehnt. Es gibt die Möglichkeit mehrere Fenster simultan zu öffnen, diese Fenster können durch direktes Auswählen oder durch die Taskleiste fokussiert werden, durch Mausinteraktion vergrößert werden und durch drei Schaltflächen in der rechten oberen Ecke minimiert, an die Bildschirmgröße angepasst und geschlossen werden. Zudem hat jede Applikation sein eigenes GUI, um dem Benutzer eine Interaktion mit der App zu ermöglichen. Das GUI von SpercOS ist nichts als eine HTML-Seite, dessen DOM dynamisch durch Javascript verändert wird.

4.1.2 Appauflistung mit Kurzbeschreibungen

- **Über SpercOS**: Kleine App mit Kurzbeschreibung des Projektes.
- **Logout**: Nur zum ausloggen.
- **Benutzereinstellungen**: Der Admin kann alle Profile bearbeiten, der Normalbenutzer nur sich selbst.
- **Appmanager**: Installation, Deinstallation von Apps, sowie Updatemanager für SpercOS
- **Snake**: Ein kleines Minigame
- **Note**: Ein kleiner Texteditor ohne tolle Funktionalitäten
- **Kalender**: Terminkalender für einfache Terminverwaltung.
- **Timewatcher**: Ein Countdown und eine Stoppuhr
- **Minen**: Ein ähnliches Spiel zu Microsofts Minesweeper
- **Maps**: Ein Kartendienst
- **Messenger**: Ein Mail Dienst unter den Benutzern, welcher ebenfalls zum Verschicken von E-Mails verwendet werden kann.

4.2 Programmierer

4.2.1 Unterscheidung Window API und App API

SpercOS ist bedienbar mit zwei API's, auf die der Appentwickler direkt zugreifen kann. Um den Unterschied zwischen diesen verstehen zu können, ist es notwendig, dass man die verwendete Technik versteht. Bei dem Start einer Applikation wird eine Javascript-Funktion des „Window API“ namens „open_window“ nur mit der eindeutigen App ID als Parameter gestartet. Diese Funktion ruft über AJAX (**A**synchronous **J**avascript **A**nd **X**ML) Informationen bezüglich des Fensterlayouts wie Höhe, Breite, Hintergrundfarben, Icon, Maximierbarkeit, Minimierbarkeit, Größenanpassbarkeit und Schließbarkeit, nachdem sie abgeglichen hat, ob der Abfragende entsprechende Berechtigungen zum Start dieser App hat. Über dieses API hat der Entwickler auch während der Laufzeit der App (also nicht nur für die Initialisierung) die Kontrolle über das dargestellte Fenster. Als Fensterinhalt wird ein Inhalt über das App-API erzeugt. Es stehen mehrere Funktionen in diesem API zur Verfügung, womit der Datenbankzugriff, die Dateiverwaltung, das Design, die Sprache und Zugriffsberechtigungen automatisch oder vereinfacht gesteuert werden kann.

4.2.2 Window API Funktionalitäten

- „**open_window**“: Das ist die (bisher) umfangreichste Funktion vom Window API. Der einzige Startparameter ist die ID von der App, die gestartet werden soll. Diese ID wird zuerst an den Server geschickt, welcher aus der Datenbank alle relevanten Daten zu dieser App nach einer Berechtigungskontrolle an den Client in einer Zeichenkette getrennt von einem Indikationsstring zurückschickt. Nach Kontrolle des korrekten Erhalts der Daten generiert der Client einen neuen „HTML-Div“, worin das neue Fenster enthalten ist. Individuell wird bei diesem in der Initialisierung nur das Icon, der Titel, die Buttons zum Maximieren, Minimieren und Schließen, die Breite, die Höhe und der iFrame-Pfad zur App API festgelegt. Außerdem wird eine neue ID für das neue Fenster erzeugt und mit dem diesem verknüpft. Nachdem das Fenster eingebunden, wird ein neuer Listeneintrag in

der Taskleiste erzeugt und eingebunden. Da jQuery nur auf Elemente zugreifen kann, die bereits in der HTML-DOM enthalten sind, bekommen die Elemente erst nach dem Einbinden in die Seite die Funktionen, dass die Größe verändert werden kann und sie verschoben werden können. Zudem bekommen die Schaltflächen und die Aktivierungsflächen ihre Funktionen zugewiesen. Aus ästhetischen Gründen wird der Titel beim Einbinden nicht direkt angezeigt, sondern erst anschließend langsam eingeblendet.

- „**window_maximize**“: Maximiert das Fenster, indem sie mithilfe der Fenster ID als Parameter das Fenster auf Höhe 100% minus der beiden Bedienleisten bringt, es repositioniert und die Möglichkeit des manuellen Verschiebens und Vergrößern/Verkleinern entfernt. Wenn sie erkennt, dass das Fenster bereits maximiert ist, führt sie das Gegenteil aus. Dabei wird das Fenster immer halb so hoch und breit wie der HTML-Körper.
- „**window_close**“: Blendet nur das Fenster langsam aus und entfernt es anschließend aus der DOM.
- „**window_minimize**“: Minimiert das Fenster, indem es die Höhe kontinuierlich verringert und damit den unteren Fensterrand nach oben zieht, bis die Höhe gleich Null ist und das Fenster unsichtbar ist.
- „**window_minimize_back**“: Diese Funktion kann logischerweise nur durch Klick auf den Taskbar-Eintrag des minimierten Fensters aufgerufen werden und agiert umgekehrt zu der „window_minimize“ – Funktion.
- „**window_make_active“**: Es kann immer nur ein Fenster simultan aktiv sein. Diesen Status erhält das Fenster, wenn der Benutzer mit diesem interagiert, also entweder auf das Fenster (mit dessen Inhalt), oder auf den Taskbar-Eintrag klickt.

4.2.3 App API Funktionalitäten

Da die App-API noch nicht fertiggestellt wurde, fehlen in dieser Auflistung die Funktionen, die das Verwalten der Datenbank übernehmen.

- **Aufrufroutine**: Beim Öffnen eines neues Fensters wird wie zuvor beschrieben[7] ein iFrame erzeugt mit einem Link zum App-API. An diesem Link ist mit der GET-Methode die App ID angehängt. Nach der Benutzerauthorisationskontrolle wird die Verfügbarkeit der einzubindenden Appdateien validiert. Die App API erzeugt eine HTML-Seite mit einem Standardstylesheet, bindet dort die für die Apps individuellen Javascript und CSS Dateien ein und schickt dies an den Client. Auf dem Server hat jede App einen eigenen Ordner. Dieser Ordner enthält alle für den Start erforderlichen Dateien, sowie Dateien, die der Programmierer für den Betrieb der App vorgesehen hat. Beim Aufruf muss in diesem Ordner das Icon unter „icon.png“ auffindbar sein und die Datei, die von der App zuerst aufgerufen werden soll, muss als „content.php“ gespeichert worden sein. Zudem müssen dort alle Dateien vorliegen, die als CSS oder Javascript – Dateien in der Datenbank stehen. Um innerhalb der App eine neue Inhaltsseite aufzurufen, muss man die App API mit dem GET-Schlüssel „file“ und dem relativen[8] Dateinamen als Inhalt aufrufen.
- „**set_lang**“: Gibt einen relativen Pfad zu einer Sprachdatei zurück. Dafür ist allerdings notwendig, dass ein numerischer Wert für die Anzahl der Ordnerebenen unter dem „SpercOS“ - Rootordner angegeben wird. Außerdem braucht die Funktion den Namen der Sprachdatei. Die Sprache ermittelt er nach drei Kriterien: Zuerst wird geguckt, ob der Benutzer sich in dieser Sitzung für eine Sprache bereits entschieden hat, die über den GET-Parameter „lang“ übertragen worden wäre. Wenn das nicht der Fall ist, wird in den aktuellen Sitzungsinformationen danach gesucht. Sollten auch diese leer sein, wird immer die deutsche Version gewählt.

---

7 Siehe: Funktionalitäten → Programmierer → Window API

8 Relativ zum App-Ordner

- „**auth_user_session**“: Überprüft, ob der Benutzer angemeldet ist. Da in den Sitzungsinformationen nur eine Benutzer ID und ein gehashtes Passwort eingetragen sind, muss er zuvor die Authentizität dieser Daten überprüfen.
- „**auth_user_admin**“: Ziemlich ähnlich der Funktion „auth_user_session“, nur mit dem Unterschied, dass hierbei überprüft wird, ob der angemeldete Benutzer ein Administrator ist.
- „**show_menue**“: Ein Menüaktuator (Der Menüöffner) ruft diese Funktion auf und öffnet ein Menü, welches sich schon in der HTML DOM befindet. Dieses Menü schließt sich bei einem beliebigem Klick im Browserfenster und ruft bei einem Klick auf ein Menüpunkt eine andere Funktion auf. Außerdem übernimmt sie das automatische positionieren des Menüs auf der Oberfläche.
- „**rgbToHex**“: Diese Funktion wird für die Umwandlung der Farbbezeichnung mit drei dezimalen 8-Bit Werten für die Farben Rot, Grün und Blau zur 6-stelligen hexadezimalen HTML Notation benötigt.

5 Implementierung

5.1 Allgemeine Sicherheitsvorkehrungen

Generell ist SpercOS durch moderne Codearchitektur, Eingabebereinigung und verpflichtende Nutzung aktueller PHP, Apache und MySQL Versionen[9] relativ gut geschützt. Dennoch gibt es Punkte, wo selbst Entwickler gefragter Betriebssysteme Probleme bekommen, was die Sicherheit angeht. Die BS-Entwickler müssen abwägen, wie viele Freiheiten die Softwareentwickler für die Anwendungen, die auf dem BS laufen sollen, sinnvoll sind und ab wann ein akutes Sicherheitsproblem vorherrscht. Apple hat die Lösung für Ihre mobilen Geräte darin gefunden, alle neuen Anwendungen (und Updates) erst eigenständig zu prüfen und dann zentral anzubieten (Apple). Diese Lösung erfordert selbst verständlich einen enormen Aufwand (folglich zusätzliche Kosten) und gewährleistet keine Anonymität bei Bezug einer neuen Applikation. Andere Anbieter sichern sich zumeist durch Anti-Viren-

9 Am 12.05.2014: PHP: 5.5.12; MySQL: 5.6.17; Apache: 2.4.9

Software ab. Diese Methode hat aber den Nachteil, dass häufig von solchen nur in einer Datenbank nach bekannten, schädlichen Muster in einer Software gesucht wird und bei einem Treffer (im Idealfall) vor einer Ausführung interveniert wird. Natürlich werden dabei keine neuartigen Schädlinge erkannt, sodass diese Methode eher unzureichend zum Beispiel sensible Daten vor einem individuellem Angriff schützt (C. Hoffman).

SpercOS hat kaum Abwehrmechanismen, die ein Infizieren des Systems verhindern würden. Der einzige, und bei der Zielgruppe von Webentwicklern dennoch wirksame, Schutz ist, dass eine Installation von neuen Apps nur durch Administratoren durchgeführt werden kann.

Außerdem sind alle Server-Interpretationsdateien in drei Kategorien untergeordnet und werden entsprechend dieser geschützt:

1. Startdatei: Diese Datei wird am Anfang direkt vom Client aufgerufen und definiert eine „SECURE“ - Konstante.
2. Include-Dateien: Dateien, die während der Laufzeit einer Startdatei temporär eingefügt werden und verschiedenste, globale Funktionalitäten erfüllen können. Zur Sicherheit wird die zuvor definierte „SECURE“ - Konstante abgefragt. Dadurch wird verhindert, dass der Client direkt diese Dateien anfragt und durch Parameterübergabe Informationen erhält, die sonst nicht für ihn bestimmt währen.
3. Sprachdateien: Müssen nicht gesichert, da dort keine direkten Ausgaben, Parameterverarbeitungen oder Parameterweitergaben erfolgen.

Weitere Standardmaßnahmen sind, dass in die Sitzungsinformationen keine konkreten Daten bis auf die User-ID und das gehashte Passwort geschrieben werden, sowie eine automatische Ordnerinhaltsauflistung mit Dummyindex-Dateien verhindert wird. Allerdings gibt es noch potentielle Sicherheitslücken:

1. Apps können auf servergelagerte Dateien zugreifen und verändern
2. Apps können auf die Datenbank zugreifen und diese manipulieren

3. Appentwickler bereinigen möglicherweise die Benutzereingaben nicht oder nicht korrekt, was eine Gefahr für Code-injections darstellt.

5.2 Allgemein[10]

Beim Start wird durch die Serverkonfiguration zuerst die index.php aus dem Root-Verzeichnis aufgerufen. Diese bindet zuerst die „mysql_inc.php“ und die „funcs_inc.php“ in die derzeitige Laufzeit ein. Nachdem klar ist, ob der Benutzer bereits angemeldet ist oder nicht, gibt er die entsprechenden Kopfinformationen aus und bindet danach die richtige HTML-Körper-Datei („login_inc.php“ oder „desktop_inc.php“) und Sprachdatei ein.

Mithilfe der „login_inc.php“ wird ein Formular ausgegeben. Beim Abschicken dieses Formulars werden die Daten per AJAX an den Server gesendet, der die Daten evaluiert und gegebenenfalls speichert. Anschließend schickt er einen Code zurück, der im Informationsfeld ausgegeben wird. Wenn diese Daten allerdings gleich „login_ok“ sind, dann wird der Ladebalken angezeigt, der am Ende eine Neuladung von „index.php“ zulässt. Dieser Datenstring wird dann gesendet, wenn der Benutzer sich erfolgreich registriert, oder angemeldet hat. In diesen Fällen wurde dem Benutzer ein temporärer Cookie zugeordnet, welcher nur für die Dauer der Sitzung aktiv ist. Da der Benutzer sich jetzt mit einer Sitzung bei der „index.php“ ausweisen kann, bekommt sie den Desktop ausgegeben.

5.2.1 Resultierende Server- und Clientkonfiguration

In der PHP-Konfiguration muss folgendes eingestellt sein:

- **„error-reporting“: „E_ALL & ~E_DEPRECATED & ~E_STRICT“** - Wichtig, damit nicht alle eher irrelevanten Fehler ausgegeben werden.
- **„magic_quotes_gpc“: „off“** - Dies kann zu Fehler bei der Datenweiterverarbeitung führen. Das Escapen der Eingaben wurde bereits manuell vorgenommen.

10 Für ein besseres Verständnis dieses Abschnittes wird eine Betrachtung des Anhangs „Dateiorganigramm“ empfohlen.

Das System wurde erfolgreich nur unter Firefox ab Version 28, Google Chrome ab Version 32 getestet. Dabei hat sich herausgestellt, dass der Firefox erhöhte Leistungsprobleme aufwies, wodurch eine Nutzung von Chromium empfohlen wird.

5.3 Datenbank

sos_apps:

Feld	Datentyp	Beschreibung
app_id	int(10)	Eindeutige Identifizierung
title	varchar(50)	Name der App
req_full_rights	tinyint(1)	Bekommt über iFrame Javascript-Rechte
scripts	text	Welche Skripte eingebunden werden sollen
css	text	Welche CSS-Dateien eingebunden werden
width	int(10)	Die Initialbreite
height	int(10)	Die Initialhöhe
resizable	tinyint(1)	Fensterfläche darf manuell verändert werden
fullscreenable	tinyint(1)	Darf auf Vollbild gestellt werden?
closeable	tinyint(1)	Fenster darf manuell geschlossen werden
on_close	varchar(50)	Die Js-Funktion vor dem Schließen ausf.
minimizable	tinyint(1)	Fenster darf minimiert werden
start_on_full	tinyint(1)	Fenster wird bei Initialisierung maximiert
bg_title	varchar(15)	Hintergrundfarbe Fensterkopf
bg_body	varchar(15)	Hintergrundfarbe Fensterkörper
admin_only	tinyint(1)	Nur Administrator darf ausführen

sos_desktop_icon[11]:

Feld	Datentyp	Beschreibung
desktop-icon-id	Int(10)	Icon Identifizierung
app_id	int(10)	Die verbundene App-Id
from_left	int(10)	Die Icon Position von links in px
from_top	int(10)	Die Icon Position von oben in px
user_id	int(10)	Welchem Benutzer gehört der Eintrag (ID)

11 Verbundene Funktion noch nicht implementiert

sos_files[12]:

Feld	Datentyp	Beschreibung
file_id	int(10)	Datensatz ID
app_id	int(10)	Welche App kann das starten?
user_id	int(10)	Wem gehört dieser Satz (Standard Ersteller)?
protected	varchar(50)	Passwortgeschützt, wenn Inhalt (md5)
title	varchar(50)	Dateiname
date_created	varchar(50)	Das Erstellungsdatum
date_last_edit	varchar(50)	Wann wurde es zuletzt bearbeitet?
created_by	int(10)	Welcher Benutzer hat den Satz erstellt?
last_edit_by	int(10)	Von wem wurde es zuletzt bearbeitet?
shared	tinyint(1)	Zugriff von Anderen erlaubt?
hidden	tinyint(1)	Wird in der Gesamtauflistung nicht gezeigt
content	text	Der eigentliche Inhalt

sos_users:

Feld	Datentyp	Beschreibung
user_id	int(10)	Benutzer-ID
username	varchar(50)	Der ebenfalls einzigartige Benutzername
pwd	varchar(50)	Passworthash (md5)
mail	varchar(75)	Die E-Mail Adresse
lang	varchar(5)	Die bevorzugte Sprache
admin	tinyint(1)	Ist Benutzer ein Administrator
recent	varchar(50)	Enthalt die zuletzt verwendete Funktion
lastlogin	varchar(50)	Wann wurde zuletzt eingelogged?
sent_command	varchar(50)	Ermöglicht eine Fernsteuerung des Benutzers
deleted	tinyint(1)	Wenn Wert = 1, wurde Benutzer gelöscht
tooltips	tinyint(1)	Sollen die Tooltips angezeigt werden?
style	varchar(50)	Reserviert für zukünftige Funktion

12 Nach dem EAV-DB-Prinzip (Anonym 4)

6 Literaturverzeichnis

Anonym 1: „SMF 2.0.7 Changelog“. Online im Internet: http://download.simplemachines.org/index.php?thanks;filename=smf_2-0-7_changelog.txt (Stand: 12.05.2014).

Anonym 2: „Download the Arduino Software“. Online im Internet: http://arduino.cc/en/main/software (Stand: 12.05.2014).

Anonym 3: „The Linux Kernel API“. Online im Internet: https://www.kernel.org/doc/htmldocs/kernel-api/ (Stand: 12.05.2014).

Anonym 4: „The EAV/CR Model of Data Representation“. Online im Internet: http://ycmi.med.yale.edu/nadkarni/EAV_CR_frame.htm (Stand: 15.05.2014).

Apple: „App Review“. Online im Internet: https://developer.apple.com/app-store/review/ (Stand: 12.05.2014).

Balzert, Helmut: „Lehrbuch Grundlagen der Informatik“ („Konzepte und Notationen in UML, Java, und C++; Algorythmik und Software-Technik; Anwendungen“). 2. Auflage. Spektrum Akademischer Verlag. 2005.

Grieser, Franz / Irlbeck, Thomas: „Computer - Lexikon: Das Nachschlagewerk zum Thema EDV“. Beck: Deutscher Taschenbuchverlag München. 1993.

Hoffman, Chris: „HTG explains how antivirus software works“. Online im Internet: http://www.howtogeek.com/125650/htg-explains-how-antivirus-software-works/ (Stand: 12.05.2014).

Lipinski, Klaus: „Computer“. Online im Internet: http://www.itwissen.info/definition/lexikon/Computer-computer.html (Stand: 12.05.2014).

Microsoft: „Gewusst wie: Starten von Diensten“. Online im Internet: http://msdn.microsoft.com/de-de/library/htkdfk18%28v=vs.110%29.aspx (Stand: 12.05.2014).

Minhas, Saifullah: „Architectural Design Theories“. Online im Internet: http://esenciel.blogspot.de/2013/12/design-theories.html (Stand: 12.05.2014).

Schmidt, Meiler: „Betriebssysteme“. Online im Internet: http://www.informatik.uni-leipzig.de/~meiler/Schuelerseiten.dir/MSchmidt/allgemein.html (Stand: 12.05.2014).

The jQuery Foundation: „jQuery“ („write less, do more“). Online im Internet: http://jquery.com/ (Stand: 15.05.2014).

Versicherung und Erklärung

Hiermit versichere ich, dass ich die Arbeit selbstständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und die Stellen der Projektarbeit, die im Wortlaut oder im wesentlichen Inhalt aus anderen Werken entnommen wurden, mit genauer Quellenangabe kenntlich gemacht habe.
Verwendete Informationen aus dem Internet werden dem Kursleiter auf Verlangen vollständig im Ausdruck zur Verfügung gestellt.

_____

(Ort, Datum)

_____ _____

(Vor- und Nachname in Druckschrift) (Unterschrift)

Hiermit erkläre ich, dass ich damit einverstanden bin, wenn die von mir verfasste Projektarbeit der schulinternen Öffentlichkeit zugänglich gemacht wird.

_____

(Ort, Datum)

_____ _____

(Vor- und Nachname in Druckschrift) (Unterschrift)