**Dokumentation:**

**HTML5- Dateibrowser**

**Ausbildungsbereich: Technik**

**Studiengang: Informatik**

**Vorlesung: Web-Engineering I**

**Dozent: Martin Spörl**

**Student: Maximilian Meier**

**Kurs: TINF20C**

**Matr. Nr.: 6020212**

Inhalt

[1 Verwendeter Browser 3](#_Toc79257045)

[2 Sicherheit 3](#_Toc79257046)

[2.1 Authentifizierung 3](#_Toc79257047)

[2.2 Logout 3](#_Toc79257048)

[3 Darstellung 3](#_Toc79257049)

[3.1 Anpassung an das Browserfenster 3](#_Toc79257050)

[3.2 Usability 4](#_Toc79257051)

[3.3 Fehlererkennung 4](#_Toc79257052)

[3.4 Verzeichnisse & Datentypen 4](#_Toc79257053)

[4 Navigation 4](#_Toc79257054)

[4.1 Verzeichnis öffnen 4](#_Toc79257055)

[4.2 Ebenen Wechseln 4](#_Toc79257056)

[4.3 Wiedererkennung 5](#_Toc79257057)

[5 Datei-Handling 5](#_Toc79257058)

[5.1 MP4-Video-Dateien abspielen 5](#_Toc79257059)

[5.2 Audio-Dateien abspielen 5](#_Toc79257060)

[5.3 Bild-Dateien ansehen 5](#_Toc79257061)

[5.4 Text-Dateien bearbeiten 5](#_Toc79257062)

[5.5 Dateien herunterladen 6](#_Toc79257063)

[5.6 Dateien Löschen 6](#_Toc79257064)

[5.7 Text-Datei erstellen 6](#_Toc79257065)

[5.8 Dateien Hochladen 6](#_Toc79257066)

[6 Verzeichnis-Handling 7](#_Toc79257067)

[6.1 Verzeichnis Löschen 7](#_Toc79257068)

[6.2 Verzeichnis-Erstellen 7](#_Toc79257069)

# ****1 Verwendeter Browser****

Für den Dateibrowser wurden **Google Chrome** (Version 92.0.4515.131) und **Mozilla Firefox** (Version 90.0.2) zur Entwicklung verwendet. Der Dateibrowser sollte für beide Browser funktionieren.

# ****2 Sicherheit****

Um den Dateibrowser sehen zu können muss der Nutzer Benutzername und Password in die dafür vorgesehenen „Input“-Felder eingeben. Damit das Passwort nicht als Klartext sichtbar ist wurde dem zuständigen Input Tag das Attribut „type“ mit dem Wert „password“ gegeben.

# 2.1 Authentifizierung

Funktion: login() [login\_logout.js]

Um vom Fileserver einen Authentifizierungstoken zu erhalten, wird der Serveranfrage der Benutzername und das Passwort mitgegeben. Aus der Antwort des Servers erhält man den Token welcher im „Session Storage“ gespeichert wird.

# 2.2 Logout

Funktion: logout() [login\_logout.js]

Mithilfe das Logout-Button, in der oberen rechten Ecke der Seite, kann der Nutzer sich jederzeit ausloggen. Wenn sich der Nutzer ausloggt kann er den Dateibrowser nicht mehr sehen und wird zurück zum Login Bildschirm gebracht. Dabei wird der Token im „Session Storage“ gelöscht.

# 3 Darstellung

# 3.1 Anpassung an das Browserfenster

**Die Breite von Elementen ist durch das CSS-Attribut „width“ geregelt und ist für alle Elemente in Prozent angegeben. Dadurch passen sich alle Elemente entsprechend er Browser Breite an. Mit Hilfe von „Media Queries“ wurde außerdem erreicht das sich der Abstand zwischen Elementen in der „navBar“, ganz oben auf der Seite, verringert sollte das Fenster kleiner als 600px sein.**

# ****3.2 Usability****

Alle permanenten Buttons/Icons befinden sich zusammen am Anfang der Seite in der „navBar“. Hier kann jederzeit ein neues Verzeichnis oder ein neues Textfile erstellt werden, der Dateibrowser kann neu geladen werden, es kann zurück gegangen werden und man kann sich ausloggen. Alle Funktionen an einem Ort vereinigt. Die Farbpallette der Tabelle und aller echter Buttons mit Text ist speziell gewählt um miteinander gut zu funktionieren.

# 3.3 Fehlererkennung

Funktion: ErrorHandling() [errorHandling.js]

Sollte ein Fehler beim Login entstehen, wie z.B. ein falsches Passwort oder Benutzername, wird dieser Direkt über dem Login-Button dargestellt. Alle anderen Fehler werden durch die Funktion ErrorHandling() dem Nutzer mitgeteilt. Dafür entsteht ein Banner ganz unten an der Seite mit der jeweiligen Fehlermeldung für je 3 Sekunden.

# 3.4 Verzeichnisse & Datentypen

Alle Verzeichnisse und Dateien werden in einer Tabelle Dargestellt, diese Tabelle beinhaltet den Namen, ob die Datei herunterladbar ist und den Typen der Datei. Zusammen mit Buttons zum löschen und ggf. herunterladen. Der Typ zeigt an ob es sich um ein Verzeichniss handelt mit „dir“. Sollte es sich um eine Datei handeln wird der Dateityp in der Typ Spalte gezeigt. Z.B. „image/png“ für ein png-Bild oder „text/plain“ für ein normales Textfile.

# 4 Navigation

# 4.1 Verzeichnis öffnen

Funktion: clickedOnPath() [fileHandling.js]

Wenn der Nutzer auf den Namen eines Verzeichnisses klick wird dieses geöffnet. Dafür wird die Tabelle neu geladen mit dem Inhalt des geklickten Verzeichnisses.

# 4.2 Ebenen Wechseln

Funktionen: previousDir() [directories.js], safeState()/loadState() [history.js]

Durch das klick auf den zurück Knopf, oben links auf der Seite, ist es dem Nutzer möglich eine Verzeichnissebene nach oben zu springen. Mit den Vor- und Zurück-Buttons des Browsers ist es möglich zu allen vorherigen Verzeichnissaufrufen zurück zu gehen. Es wird die Browser-History zurück gegangen.

# 4.3 Wiedererkennung

Funktion: displayCurrentDir() [directories.js]

Der Path des aktuellen Verzeichnisses wird immer in der „navBar“ dargestellt.

# 5 Datei-Handling

Wenn man eine Datei abspielen möchte klickt man auf ihren Namen. Durch ein Switch-Case in der Funktion loadFile() [loadFiles.js] wird dann die jeweilige Funktion aufgerufen welche diese Datei darstellt.

# 5.1 MP4-Video-Dateien abspielen

Funktion: displayVideo() [loadFiles.js]

In der „contents“ div wird ein „content-wrapper“ erstellt, in welchem ein „video“ Element erstellt wird mit dem das angeklickte Video dargestellt wird und abgespielt werden kann. Mit dem Button „Close“ kann kann das Video wieder von der Seite entfernt werden. Dies geschieht durch das Entfernen des „content-wrapper“.

# 5.2 Audio-Dateien abspielen

Funktion: displayAudio() [loadFiles.js]

Dieser Prozess funktioniert analog zu 5.1 allerdings wird ein „audio“ Element erstellt satt einem „video“ Element.

# 5.3 Bild-Dateien ansehen

Funktion: displayImage() [loadFiles.js]

Dieser Prozess funktioniert analog zu 5.1 allerdings wird ein „img“ Element erstellt statt einem „video“ Element.

# 5.4 Text-Dateien bearbeiten

Funktion: displayText() [loadFiles.js], saveText() [fileHandling.js]

Das Laden und Darstellen von Textdateien erfolgt analog zu 5.1, allerdings wird ein „textarea“ Element benutzt statt einem „video“ Element. Das Textfile kann in der Textarea beliebig editiert werden. Diese Änderungen können durch das Klicken auf den „Save“-Button, welcher die saveText() [fileHandling.js] Funktion aufruft, gespeichert werden. Dieser Button ist wird speziell für Textfiles geschaffen und dargestellt.

# 5.5 Dateien herunterladen

Funktion: downloadFile() [fileHandling.js]

Durch das klicken des Download-Buttons in der ganz rechten Spalte der Tabelle kann jede Datei, jederzeit heruntergeladen werden.

# 5.6 Dateien Löschen

Funktion: deleteFile() [fileHandling.js]

Durch das klicken auf den Löschen-Button, die Mülltonne, in der ganz rechten Spalte der Tabelle kann jede Datei, jederzeit gelöscht werden.

# 5.7 Text-Datei erstellen

Funktionen: createTextFile(), createNewTextFile() [createFile&Dir.js]

Durch das klicken auf das „Create File“ Icon in der „navBar“ werden zwei Textareas in der „contents“ div geschaffen. Beide sind noch einmal in einer unter div Namens „wrapper\_create“. Hier kann der File Name und der File Text eingegben werden. Der File Name ist dabei auf maximal 256 Zeichen reduziert. Mit dem „Create“-Button, welcher die createNewTextFile() Funktion aufruft, lässt sich das erstellte Text File speichern im aktuellen Verzeichnis.

# 5.8 Dateien Hochladen

Funktion: uploadFile() [fileHandling.js]

Durch das klicken auf das „Upload File“ Icon in der „navBar“ lässt sich eine Datei in das aktuelle Verzeichnis hochladen. Das hochladen geschieht mit Hilfe eines „Input“ Elements mit dem „type“ „file“. Das Icon „upload File“ klickt lediglich auf diesen Input um so eine Einheitlich Struktur in der „navBar“ zu ermöglichen, damit jeder Button durch ein Icon dargestellt werden kann.

# 6 Verzeichnis-Handling

# 6.1 Verzeichnis Löschen

Funktion: deleteFile() [fileHandling.js]

Das Verzeichnislöschen funktioniert analog zum Dateien löschen deshalb kann dieselbe Funktion verwendet werden sowie dasselbe Icon/Button in der Tabelle.

# 6.2 Verzeichnis-Erstellen

Funktionen: createDir(), createNewDir() [createFile&Dir.js]

Der Prozess Funktioniert analog zum Text-Dateien erstellen. Es wird jedoch nur eine „textarea“ benötigt, da man nur den Namen des Verzeichnisses eingeben muss und keinen Content wie beim Text File. Die „textarea“ wird auch im „wrapper\_create“ erschaffen und dargestellt.