Albert-Einstein-Gymnasium

Dokumentation

zu unserem Projekt im Projektkurs

Mathe-Physik-Informatik

Autoren: Jakob Fleischer
Robin Schlaak
Lars Schmalbach

Prüfer: Herr Thomas Bachran

Abgabedatum: 21.06.2017

I Umschreibung des Projekts

Jakob

Die grundsätzliche Zielsetzung von unserem Projekt war, etwas zu programmieren, das wir auch selber danach noch sinnvoll nutzen können. Ein Projekt zu finden, dass dieses Ziel erfüllte und für uns innerhalb eines Schuljahres realisierbar gewesen wäre, gestaltete sich jedoch als schwierig, da wir zwar alle drei bereits Programmiererfahrung besaßen, diese sich jedoch auf das Programmieren einfacher Klassenstrukturen mit Rückgabewerten in der Eclipse-Konsole beschränkte. Also mussten wir, begründet auf diesen Fakt, unsere Fähigkeit ein selbständiges Programm zu schreiben, zu diesem Zeitpunkt, realistisch, als nicht vorhanden einschätzen.

Dieses Eingeständnis blies uns recht schnell die Traumwolke eines Programms, welches mithilfe einer künstlichen Intelligenz Bilder auswertet, unter den Füßen weg und ließ uns den festen Boden der Tatsachen unter dem Selbigen spüren. Aus diesem Grund kamen wir zu dem Schluss, dass eine Software, die Dateien sämtlicher Dateitypen automatisch und sortiert abspeichert, als ein sinnvolles und zugleich erreichbares erstes Ziel, welches wir dann gut ausbauen könnten. Jedoch haben wir uns auch in diesem Punkt selber überschätzt.

Wir haben zwar unser Ziel eines webbasierten Programms zur sortierten Abspeicherung von Dateien erreicht, jedoch hat alleine dieses „erste Ziel“ unsere gesamte zeitliche Kapazität verschlungen, ohne uns Zeit für Erweiterungen zu lassen. Doch dazu mehr in der folgenden detaillierten Dokumentation unseres Kampfes mit Computern, Quellcodes und vor allem mit uns selbst.

II Inhaltsverzeichnis

III Abbildungsverzeichnis

IV Tabellenverzeichnis

V Listing-Verzeichnis

1 Kapitel: Herangehensweise

Jakob

Dieses Kapitel soll sich primär mit uns beschäftigen. Damit, wie wir an die Projektarbeit herangetreten und auch wo wir auf Probleme mit derselben gestoßen sind.

1.1 Ursprüngliche Vorstellung und Konsequenzen

Jakob

Wie bereits bei unseren Vorstellungen erwähnt [siehe [**??**]], ist unsere Projektarbeit hauptsächlich negativ durch unser Kommunikationsverhalten beeinflusst worden. Unser größtes Problem lag in der Organisation unserer Zusammenarbeit. Dies ist der Tatsache verschuldet gewesen, dass wir in der Schule meist nur unter strengen Regulierungen und Vorgaben arbeiten, weshalb so ein freies und selbständiges Arbeiten sehr ungewohnt für uns war und vielerorts neben den anderen Arbeiten, bei denen bei „Nicht-Erfüllen“ direkte Sanktionen drohten, an Priorität zu verlieren schien. Aus diesem Grund war unsere Zusammenarbeit größtenteils von stark differenzierenden Arbeitshaltungen geprägt, welche sich dann auch auf die Aktivität und Harmonie unserer Kommunikation untereinander ausgeprägt haben. Dies begünstigte unseren fehlgeleiteten Ansatz [siehe [**??**]] und führte auch in der späteren Projektarbeit immer wieder zu Einbrüchen unserer Produktivität, welche im Nachhinein als vermeidbar und gar unnötig einzustufen sind.

1.2 Verspätete Planung

Lars

Zu spät haben wir gemerkt, dass unsere bisherige Arbeit weder zielführend, noch in irgendeiner Weise sinnvoll war. Deshalb wir uns erneut zusammengesetzt und intensiv beraten haben. Aus dieser Beratung entstand die Erkenntnis, dass es sinnvoll ist, einen Arbeitsplan zu entwickeln. Dieser Plan musste sich realistisch an unseren Fähigkeiten und der noch zur Verfügung stehenden Zeit orientieren. Auch die Aufteilung der verschiedenen Arbeitsschritte zwischen den drei Gruppenmitgliedern fand hier genauer statt.

1.2.1 Aufteilung

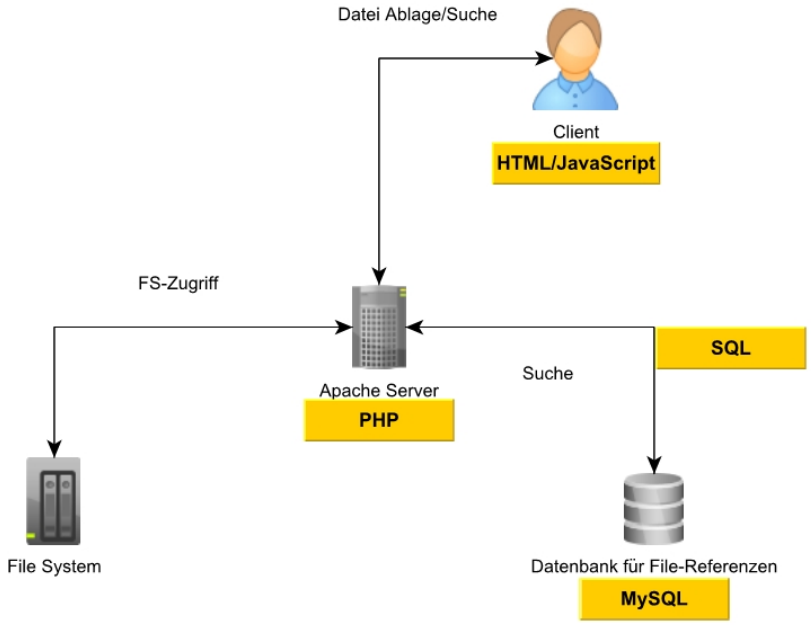

Abbildung 1: Erster realistischer Plan

In Abbildung 1 wird die Aufteilung der verschiedenen Aufgaben dargestellt. Der Client [1.2.2], der von Lars programmiert wird, umfasst - vereinfacht gesagt - das GUI, also die Benutzeroberfläche, und dessen Kommunikation mit dem Server. Der Server wird von Robin eingerichtet. Anfragen sollen derart kategorisiert werden, dass Datei-System und Datenbank entweder getrennt oder gemeinsam angesprochen werden. Die Datenbank wird von Jakob erstellt und soll zunächst nur die Referenzen zu den Dateien im Datei-System abspeichern und verwalten.

1.2.2 Client

Tabelle 1 stellt das Ergebnis unserer Beratung bezüglich des Clients dar. Folgendes ist geplant: *Typ* und *Größe* (grün) sollen automatisch erkannt werden, *Datei*, *Name* und *Datum* (rot) sollen erfragt werden können. *Zusätzliche Informationen* (blau) sollen eine weitere Möglichkeit der Eingabe darstellen. Für den Fall, dass bei der Abfrage keine Kriterieneingabe erfolgt, werden alle vorhandenen Daten ausgegeben.

Ablage	Suchen	Löschen	Ändern
- Typ - Größe - Datei - Name - Datum - zusätzliche Info	Kriterien: - Name - Inhalt - Typ - Datum - Ort	- verbunden mit Suche - senden eines Löschbefehls	Beinhaltet: - Suche - Ändern - Ablage

Tabelle 1: Planung des Clients

1.2.3 Server

Die Aufgaben des Servers in Bezug auf Client, Datei-System und Datenbank haben wir wie folgt definiert. (siehe Tabelle 2)

Aufgaben des Servers in Verbindung mit...

Client	Datei-System	Datenbank
- Auswerten der Metadaten bei der Ablage - Auswerten von Suchkriterien - Senden von Dateien bei der Suche	- Ablage - Abruf - Löschen	- Erstellen der Datenbank, falls keine vorhanden ist - Referenzen senden und erhalten - Einträge löschen

Tabelle 2: Planung der Aufgaben des Servers

1.2.4 Datenbank

Die Datenbank soll die Pfade zu den Dateien aufnehmen und den Abfragen antworten. Beinhalten sollte sie zunächst nur den Dateinamen, sowie weitere Attribute, welche in Tabelle [1] aufgeführt sind. Zusätzlich soll zu jeder Datei noch der jeweilige Pfad gespeichert werden, sodass ein Zugriff auf die Datei gewährleistet werden kann.

1.3 Zusammenarbeit und Kommunikation

Jakob

Wie bereits bei unseren Vorstellungen erwähnt [siehe [**??**]], ist unsere Projektarbeit haupt-

sächlich negativ durch unser Kommunikationsverhalten beeinflusst worden. Unser größtes Problem lag in der Organisation unserer Zusammenarbeit. Dies ist der Tatsache verschuldet gewesen, dass wir in der Schule meist nur unter strengen Regulierungen und Vorgaben arbeiten, weshalb so ein freies und selbständiges Arbeiten sehr ungewohnt für uns war und vielerorts neben den anderen Arbeiten, bei denen bei „Nicht-Erfüllen“ direkte Sanktionen drohten, an Priorität zu verlieren schien. Aus diesem Grund war unsere Zusammenarbeit größtenteils von stark differenzierenden Arbeitshaltungen geprägt, welche sich dann auch auf die Aktivität und Harmonie unserer Kommunikation untereinander ausgeprägt haben. Dies begünstigte unseren fehlgeleiteten Ansatz [siehe [**??**]] und führte auch in der späteren Projektarbeit immer wieder zu Einbrüchen unserer Produktivität, welche im Nachhinein als vermeidbar und gar unnötig einzustufen sind.

2 Kapitel: Grundlagen

Jakob & Lars

Im zweiten Kapitel dieser Dokumentation wird genauer auf die Themen eingegangen, die wir uns im Laufe des Projektkurses erarbeitet haben. Es wird sowohl auf Inhalte eingegangen, die in das Projekt eingegangen sind, als auch auf solche, die letztlich keine Verwendung gefunden haben.

2.1 LaTeX

Robin

LaTeX ist ein softwarepaket, welches die Benutzung des Textsatzsystems TeX vereinfacht. LaTeX wurde Anfang der 1980er von dem Programierer, Mathematiker und Informatiker Leslie Lamport entwickelt.[**?**][**?**] ...

2.2 XAMPP

Robin

Xampp ist ein Softwarepaket bestehend aus Freeware. Xampp vereinfacht das Instelliernen und das Konfigurieren von einem Apache Webservers mit z.B. SQLite und PHP. Außerdem sind in Xampp noch Werkzeuge wie z.B. FileZilla und phpMyAdmin vorhanden. Xampp ist nnicht für den Einsatz bei öffenliche Server gedacht, sondern dient lediglich für Entwickler, die ein schnelles und kompaktes Testsystem haben wollen, da eine starke Einschränkung der Sicherheit gibt.[**?**]

...

2.3 HTML

Lars

HTML ist die Abkürzung für „Hypertext Markup Language“, was zu deutsch „Hypertext-Auszeichnungssprache“ heißt. Es ist eine Programmiersprache, mit der man den Aufbau von Internetseiten bestimmt. Solche HTML-Dokumente stellen die Grundlage für das World Wide Web dar und werden von Browsern dargestellt.[**?**][**?**] Sie bestehen in der Regel aus drei Teilen.[**?**]

```
1 <html>
2   <head>
3     <meta charset="UTF-8">
4     <title> Titel</title>
5   </head>
```

```
6    <body>
7      Sichtbarer Text auf der Webseite.
8    </body>
9  </html>
```

Listing 1: Beispiel für ein einfaches HTML-Dokument

Der erste Teil eines üblichen HTML-Dokuments ist die Dokumenttyp-Deklaration. In ihr werden Angaben zur verwendeten HTML-Version gegeben. Im „head“, der den zweiten Teil bildet, werden Kopfdaten angegeben. Diese können sichtbar, wie zum Beispiel der Titel der Seite, oder unsichtbar sein. Unsichtbare Kopfdaten dienen beispielsweise der korrekten Darstellung der Webseite.[**?**] Anzuzeigende Inhalte werden in den dritten Teil, den „body“, geschrieben. Hier werden sämtliche Texte, Verweise, Grafiken und/oder ähnliches, die auf der Webseite sichtbar sein sollen, eingefügt.[**?**]
In unserem Fall stellt HTML die Grundlage für die optische Gestaltung des GUIs dar.

2.3.1 Frames

Eine hilfreiche Technik, die auch bei uns ihren Einsatz gefunden hat, heißt Frames. Diese Technik wurde 1996 von Netscape eingeführt. Mit ihr kann man mehrere Dateien gleichzeitig auf dem Bildschirm anzeigen lassen.[**?**] Sie wurde jedoch im Oktober 2014 mit HTML5[**?**] aus dem Standard entfernt, da sie entscheidende Nachteile aufweist. Aufgrund des Verwendungszweckes unserer Webseite benutzen wir Frames, obwohl empfohlen wird, Server-seitig andere Techniken zum Auslagern von Teilen der Seite zu benutzen.[**?**] Die stärksten Argumente waren die simple Handhabung und die guten Gestaltungsmöglichkeiten dieser Technik.
Im Folgenden Beispiel wird ein sogenanntes Frameset dargestellt, bei dem der Bildschirm, mit dem Attribut „rows“ in zwei Zeilen aufgeteilt wird, wobei die obere 20% der Pixel einnimmt und die untere den Rest, also 80%. Alternativ kann man den Bildschirm auch in Spalten aufteilen, dies geschieht mit dem Attribut „cols“. Des Weiteren wird mit dem Attribut „border“ die Breite des Randes zwischen den jeweiligen Frames angegeben.

```
1  <html>
2    <head>
3      <title>Titel</title>
4    </head>
5    <frameset rows="20%,*" border="1">
6      <frame src="Quelle1.html">
7      <frame src="Quelle2.html">
8    </frameset>
9  </html>
```

Listing 2: Beispiel für Frames in HTML

2.3.2 Tabellen

Tabellen in HTML [2.3] bieten gute, einfache und vielseitige Möglichkeiten, Internetseiten zu strukturieren. Sie wurden im Januar 1997 mit HTML 3.2 ins Standardrepertoire von HTML aufgenommen.[**?**]
Das Folgende Beispiel beinhaltet eine Tabelle mit zwei Zeilen und zwei Spalten. Tabellen in HTML werden Zeile für Zeile definiert. Eine Zeile beginnt mit <tr> und wird mit </tr> beendet. Mithilfe der Befehle <td> und </td> werden die Tabelleneinträge, also die Spalten in den jeweiligen Zeilen, definiert.

```
1  <html>
2    <head>
3      <meta charset="UTF-8">
4      <title> Titel</title>
5    </head>
6    <body>
7      <table>
8        <tr>
9          <td>
10           Oben links
11         </td>
12         <td>
13           Oben rechts
14         </td>
15       </tr>
16       <tr>
17         <td>
18           Unten links
19         </td>
20         <td>
21           Unten rechts
22         </td>
23       </tr>
24     </table>
25   </body>
26 </html>
```

Listing 3: Beispiel für Tabellen in HTML

2.3.3 Formulare

Formulare sind ein Element von HTML, [2.3] das es ermöglicht Daten zu erfassen und über das Hypertext Transfer Protocol per XMLHttpRequest, HTTP-GET oder HTTP-POST zur Verarbeitung an einen Server zu senden.[**?**] In unserem Programm kam letzteres zum Einsatz. Man kann in HTML zwar Formulare definieren und erstellen, für eine Verarbeitung und Auswertung der Eingaben ist jedoch eine andere Programmiersprache, wie zum Beispiel Javascript [2.3.4] oder PHP [2.4] nötig.[**?**]
In unserem Fall haben wir Formulare in Form von Suchfiltern und einer Anmeldemaske auch als Möglichkeit zum Hochladen von Dateien eingesetzt.
In dem folgenden Beispiel ist die Implementation eines Formulars in HTML dargestellt. Zu sehen ist ein Formular, welches ein Label, also den Text „Suchgbegriff“ enthält. Die Zugehörigkeit des darauf folgenden Eingabefelds zum Label wird durch das Attribut „name“ festgelegt. Das zweite „input“ Statement erstellt den Bestätigungsknopf, den man drücken muss, um das Formular abzusenden. Beim Absenden des Formulars wird die Datei oder die Internetseite aufgerufen, die im Kopf des Formulars unter dem Attribut „action“ steht.

```
1  <html>
2    <head>
3      <title>
4        Titel
5      </title>
6    </head>
7    <body>
8      <form action="action.php">
9        <label for="begriff">Suchbegriff</label>
10       <input type="text" name="begriff">
11
12       <input type="submit" name="Submit" value="Suchen">
13     </form>
14   </body>
15 </html>
```

Listing 4: Beispiel für Formulare in HTML

2.3.4 JavaScript

Bei JavaScript handelt es sich um eine interpretierende Programmier- beziehungsweise Skriptsprache, die 1995 von Netscape entwickelt wurde.[**?**][**?**] JavaScript ist sehr verbreitet, da sich in allen modernen Browsern Interpreter für die Sprache befinden. Es wird

hauptsächlich Client-seitig verwendet und ermöglicht es, dynamischen Einfluss auf Webseiten zu nehmen.[?] Für unser Programm kam die Sprache besonders häufig aufgrund des einfachen Umgangs mit Variablen und Funktionen zum Einsatz.

2.4 PHP

Robin

PHP (Hypertext Preprocessor, ursprünglich: Personal Home Page) ist eine Skriptsprache welche huptsöchlich zur erstellung dynamischer Webseiten und Webanwendungen genutzte wird. Und damit eine Art Erweiterung von HTML. Der Syntax von PHP ist dem von C und Perl angelent. PHP zeichnet sich durch Internet-Protokolleinbindung und Datenbankunterstützung. [?]

2.4.1 GET-/POST-Methoden

Robin

Die POST-Mehtode wir häufig eingesetzt, um eine Anfrage des Clients mit weiteren Daten, an den Server weiter zu geben. POST hat die Funktionen einen Datenblock mit dazugehörigen Informationen und die Funktion Nnchrichtin zu übertragen. Dies Mit hilfe der URI.

Die GET-MEthode hat die Möglichkeit Informationen jeglicher Art zu identifizieren. Dies auch mit hilfe der Ergebnis-URI. [?]

2.5 Apache HTTP Server

Lars

Bei dem Apache HTTP Server handelt es sich um den seit 1996 verbreitetsten Webserver der Welt. Er war das Gründungsprojekt der Apache Software Foundation und wurde im April 1995 veröffentlicht. Ursprünglich ist er eine gepatchte Erweiterung des bereits etablierten NCSA HTTP Servers. Im März 2000, also knapp 5 Jahre nach der Veröffentlichung der ersten Version wurde Apache 2.0 veröffentlicht. Es wurden sowohl die Stabilität, als auch die Geschwindigkeit verbessert. Die aktuellste Version, welche auch von den Entwicklern zur Benutzung empfohlen wird, ist die Version 2.4.25, welche am 20.06.2016 veröffentlicht wurde.[?][?]

Allgemein haben Webserver, sowie auch der Apache HTTP Server, die Hauptaufgabe, statische Dateien an Clients, wie zum Beispiel Webbrowser, zu übertragen. [?]

Auch der Apache HTTP Server, der in unserem Programm zum Einsatz kommt muss solche Dateien, hauptsächlich HTML- und PHP-Dokumente, [2.3][2.4] verwalten und entsprechend zur Verfügung stellen. Wir benutzen den Apache HTTP Server, in Verbindung

mit XAMPP, [2.2] da dieses die Installation, Konfiguration und Verwaltung des Servers erheblich vereinfacht.

2.6 Editoren und integrierte Entwicklungsumgebungen

Lars

Editoren werden zum Schreiben von Texten, wie zum Beispiel Quellcodes, benutzt. Gute Editoren helfen das Programmieren zu vereinfachen, indem sie gewisse Schlüsselwörter, sowie Befehle, farblich hervorheben, eine Autovervollständigung, eine Such- und Ersetzfunktion anbieten und den Quellcode automatisch einrücken. Zusätzlich stellen sie eine Schnittstelle für Plugins dar.[**?**]

Neben Editoren gibt es auch integrierte Entwicklungsumgebungen. Diese bestehen aus einer Sammlung von Computerprogrammen, mit denen es möglich ist, Software ohne die Verwendung vieler einzelner Programme zu entwickeln. Durch sie werden nicht nur Tippfehler verhindert, sondern auch Arbeitsschritte und somit Zeit bei der Softwareentwicklung gespart.[**?**][**?**] Bei der Entwicklung unseres Programms kamen die Editoren und integrierten Entwicklungsumgebungen Eclipse [2.6.1] (genau genommen Eclipse Neon IDE), Notepad++ [2.6.2] und Texmaker [2.6.3] zum Einsatz.

2.6.1 Eclipse

Robin

Eclipse ist eine Freeware und dient als Programierwerkzeug zur Entwicklung diverser Software. Früher wurde Eclipse als integrierte Entwicklungsumgebung (IDE) für Java genutzt. Mittlerweile wird eclipse wegen seiner Erweiterbarkeit auch für viele andere diverse Entwicklungsaufgaben genutzt. Eclipse ist der Nachfolger von IBM Visual Age for Java 4.0. Seit dem 7. Novermber 2001 ist der Quellcode für Eclipse freigegeben. [**?**] ...

2.6.2 Notepad ++

Robin

Notepad ++ ist eine Freeware und ein Texteditor für Windows. In Notepad++ kann man mit vielen verschiedenen Programiersprachen schreiben, es werden auch deren Syntax und Struktur hervorgehoben. Notepad++ selbst ist in C++ geschrieben. [**?**] ...

2.6.3 Texmaker

Robin

TeXmaker ist ein Unicode-Texteditor für das Erstellen von LaTeX-Dokumenten. Dieser Editor richtet sich insbesondere an LaTeX-Anfänger, da durch Assistenten die Erstellung von Dokumenten vereinfacht wird. [**?**] ...

2.7 GitHub

Lars

Bei GitHub handelt es sich um eine kollaborative Versionsverwaltung, die im Februar 2008 erschien. GitHub funktioniert im Browser, wobei es auch eine Desktop Version gibt.[**?**]
Durch die Arbeit mit GitHub, wird das gemeinsame, nicht zwingend parallele, Arbeiten möglich. Der Vorteil gegenüber bekannteren Konkurrenten, wie zum Beispiel Dropbox, Google Drive oder Microsoft OneDrive ist, dass man in GitHub verschiedene Branches erstellen kann. Diese Branches sind so zu sagen asynchron laufende Entwicklungsstände. Wir haben für die Arbeiten an unserem Programm zwei Branches benutzt. Zunächst den Branch „develope“, in der wir entwickelt haben. Hier haben wir unsere noch nicht komplett stabilen Versionen hochgeladen, damit die anderen Gruppenmitglieder weiter Arbeiten konnten. In unserem „master“ Branch wurde immer nur ein komplettes Programm hochgeladen, sodass man dort ein funktionstüchtiges Zwischenergebnis hatte. Ein weiterer Vorteil von GitHub gegenüber der Konkurrenz ist, dass man Quellcodes online öffnen und bearbeiten kann. Zusätzlich kann man auch sperren, dass eine Person alleine den „master“ Branch überschreibt.

2.8 Dropzone

Lars

Ursprünglich beschreibt der Begriff „Dropzone“ einen geheimen Speicherort für maschinell gestohlene Daten, wie zum Beispiel Passwörter und Konto7daten.[**?**]
In unserem Fall ist die Dropzone jedoch ein Script, welches in JavaScript [2.3.4] geschrieben wurde. Dieses dient zur einfachen Gestaltung des Datei-Uploads mittels HTML. Es soll die optische Anpassung des Eingabefelds vereinfachen.
Aufgrund ihrer Komplexität, die sich während unserer Arbeit heraus stellte, haben wir uns schlussendlich gegen die Dropzone und für einen herkömmlichen Datei-Upload mittels HTML-Formular [2.3.3] entschieden.

2.9 Java-Servlet

Jakob

Java-Servlets sind eine Weiterentwicklung der klassischen CGI Schnittstelle und sind somit für die Erzeugung dynamischer Web-Inhalte in Java-Webanwendungen zuständig. Das heißt, dass die Servlet-Klasse, entsprechend der HTTP-Methode, GET oder POST [2.3.3], mit ihrer doGet- bzw doPost-Methode die Anfrage bearbeitet, wobei sie die Anfragedaten und ein Objekt zur späteren Ausgabe des Ergebnisses der Bearbeitung entgegennimmt.[**?**]
Wie dies mit den restlichen Vorgängen bei der Bearbeitung eines HTML-Formulars [2.3.3]

im Zusammenhang steht, wird in folgender Grafik anschaulich dargestellt:

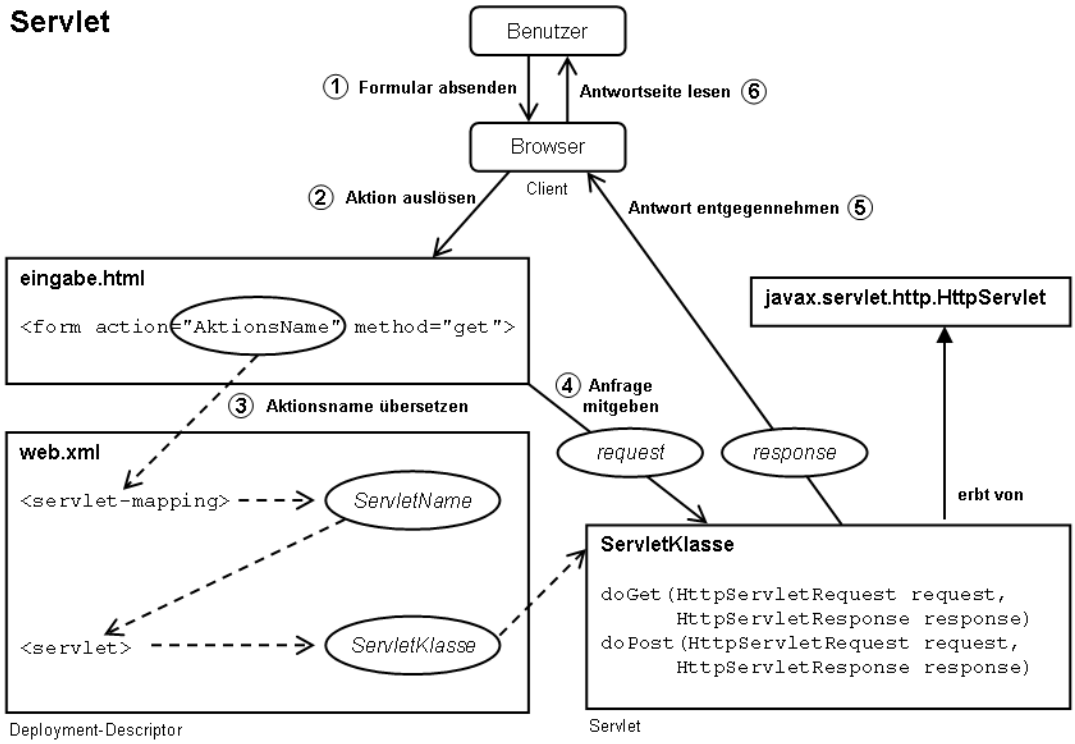

Abbildung 2: Java-Servlet[1]

Sowohl der Anfangs-, als auch der Endpunkt einer Datenverarbeitung mittels HTML-Formular besteht in dem Client, der mit dem Browser interagiert. Durch diese Interaktion sendet er ein Formular ab, welches daraufhin die gewünschte Aktion auslöst. Dazu muss der Aktionsname jedoch zunächst von einer XML-Datei dahingehend übersetzt werden, dass die entsprechende Initialisierung der ServletKlasse zur Bearbeitung dieser Aktion angesprochen wird. Sobald dies geschehen ist, erhält die ServletKlasse die Anfrage vom Client, welche sie, entsprechend dem HTML-Formular, mit ihrer doPost- oder doGet-Methode bearbeitet, und schickt die Antwort wiederum dem Browser, welcher diese dem Client anzeigt.

2.9.1 Einrichtung in Eclipse

Um eine erfolgreiche Einrichtung eines Java-Servlet mit Eclipse Neo durchführen zu können muss dieses, in der Java EE Version, und Apache Tomcat v 9.0 installiert sein. Wenn dies der Fall ist, ist es möglich in Eclipse einen neuen Server einzurichten,

[1] Quelle: `https://de.wikipedia.org/wiki/Servlet#/media/File:Servlet.png`

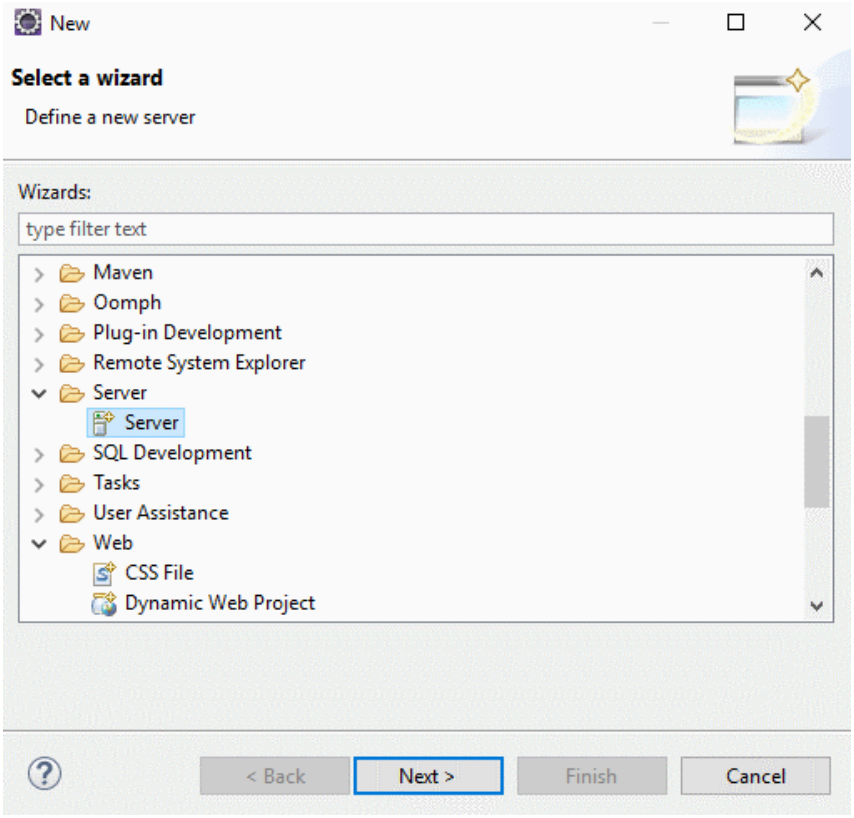

Abbildung 3: Server erstellen

wobei unter dem Ordner Apache die installierte Tomcat Version vorzufinden ist, welche hier als Servertyp verwendet wird.

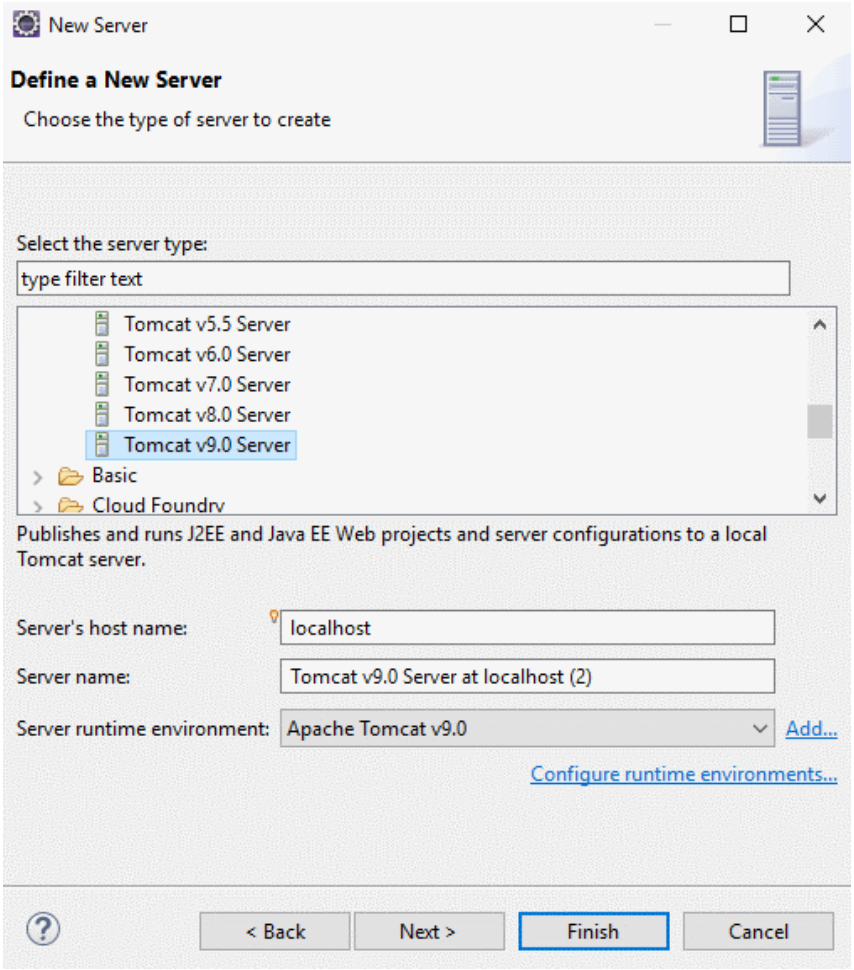

Abbildung 4: Servertyp festlegen

Nachdem dies geschehen ist, muss noch ein Dynamic Web Project eingerichtet und innerhalb desselben, im „Java Resources/src“- Ordner ein neues Servlet erstellt werden, indem sie nun programmieren können.

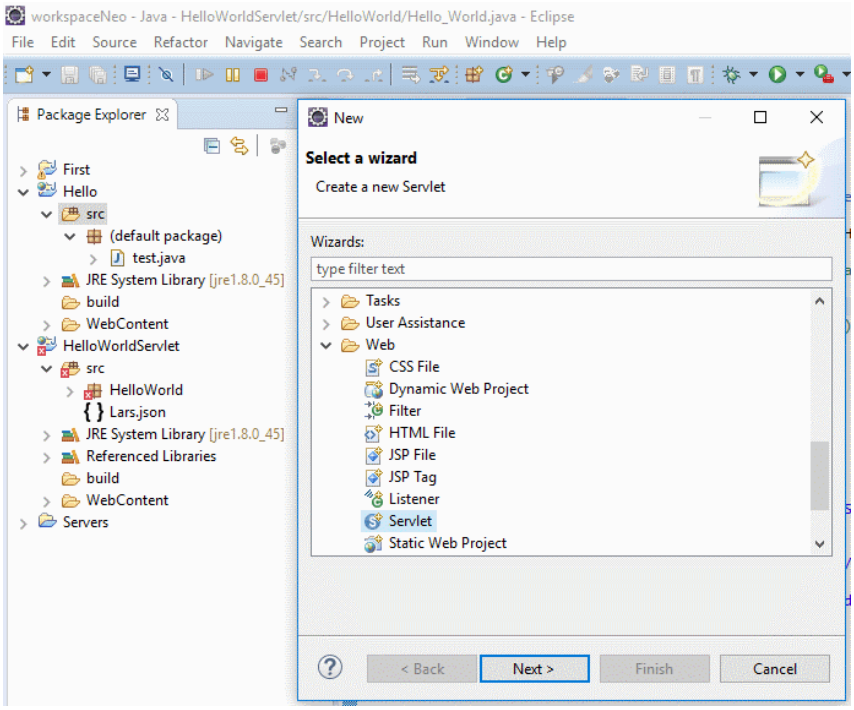

Abbildung 5: Servlet erstellen

Wenn nun aber die Fehlermeldung „cannot be resolved“ im Bezug auf die vorgegebenen import-Zeilen auftaucht

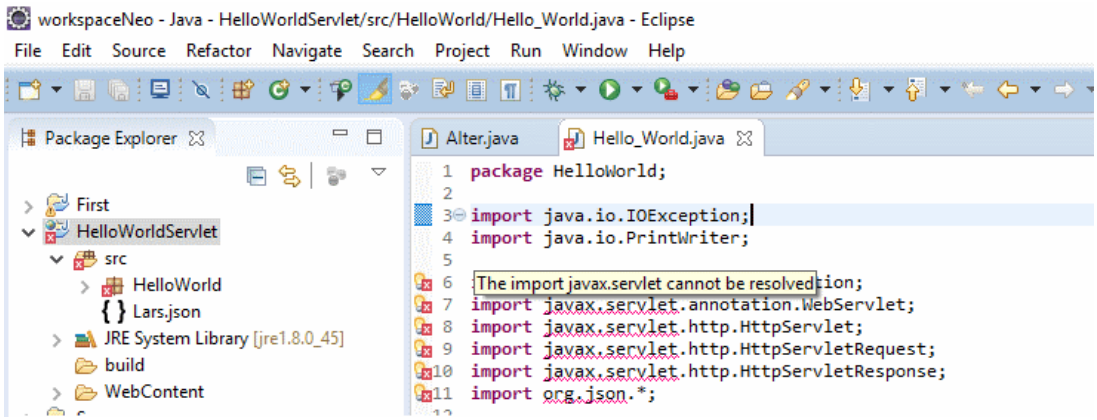

Abbildung 6: Fehlermeldung

muss außerdem noch die servlet-api.jar gedownloadet und über die Eigenschaften des „Dynamic Web Projects“ unter „Java Build Path“ in der Kategorie „Libraries“ mit der Funktion „Add External JARs“ hinzufgefügt werden.

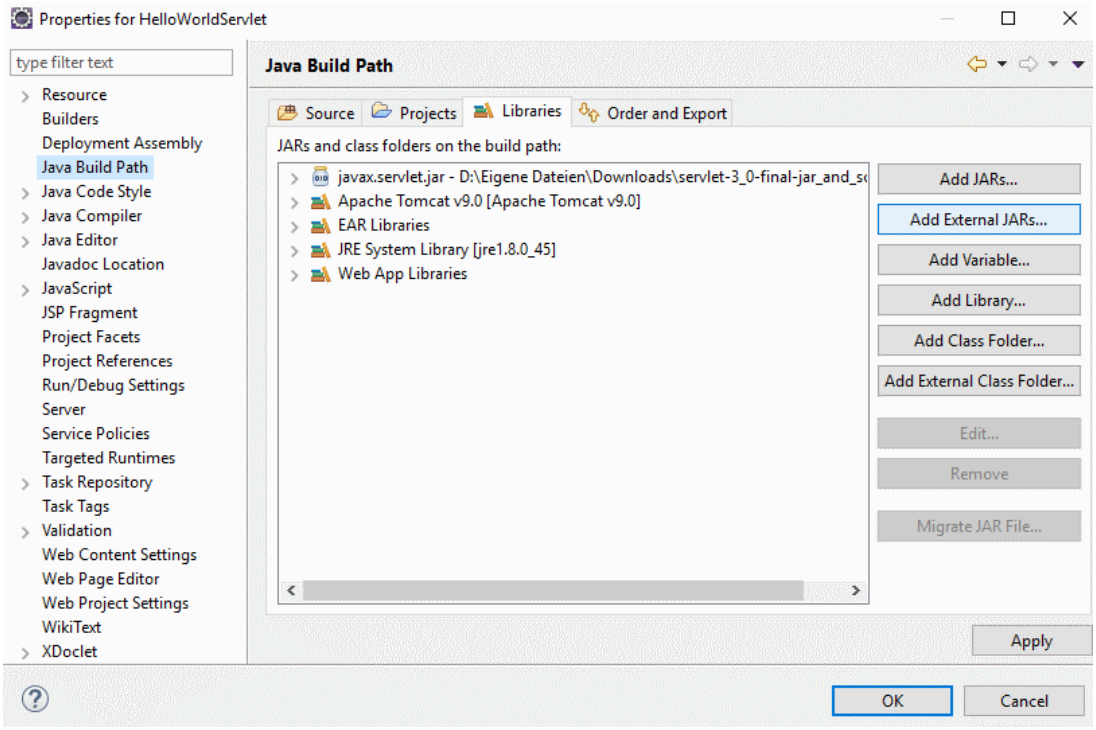

Abbildung 7: servlet-api.jar einbinden

2.9.2 Implementation

Die Implementation eines Java-Servlets gestaltet sich, im Vergleich zur Einrichtung eines solchen in Eclipse, eher einfach, da lediglich die doGet-Methode zu befüllen ist, während der restliche Quellcode bereits automatisch erstellt wurde.

So lässt sich beispielsweise mit der Java-Klasse PrintWriter ein HTML-Script in die doGet-Methode schreiben, welches dann beim Aufrufen des Servlets ausgeführt wird:

```
1  protected void doGet(HttpServletRequest request, HttpServletResponse
       response) throws ServletException, IOException {
2
3    PrintWriter writer = response.getWriter();
4
5      writer.println("<html>");
6      writer.println("<head>");
7      writer.println("<title>");
8      writer.println("Hello World");
9      writer.println("</title>");
10     writer.println("</head>");
11     writer.println("<body>");
12     writer.println("<h>");
13     writer.println("<b>Hello World Servlet</b>");
```

```
14        writer.println("</h>");
15        writer.println("<p>Das ist ein Text</p>");
16        writer.println("<a href = Hello_Space> hier </a>");
17        writer.println("<br>");
18        writer.println("<a href = https://www.google.de target = _blank>
              Google</a>");
19        writer.println("<br>");
20        writer.println("Und das auch");
21        writer.println();
22        writer.println("</body>");
23        writer.println("</html>");
24
25        writer.close();
26    }
```

Listing 5: Servlet Beispielcode

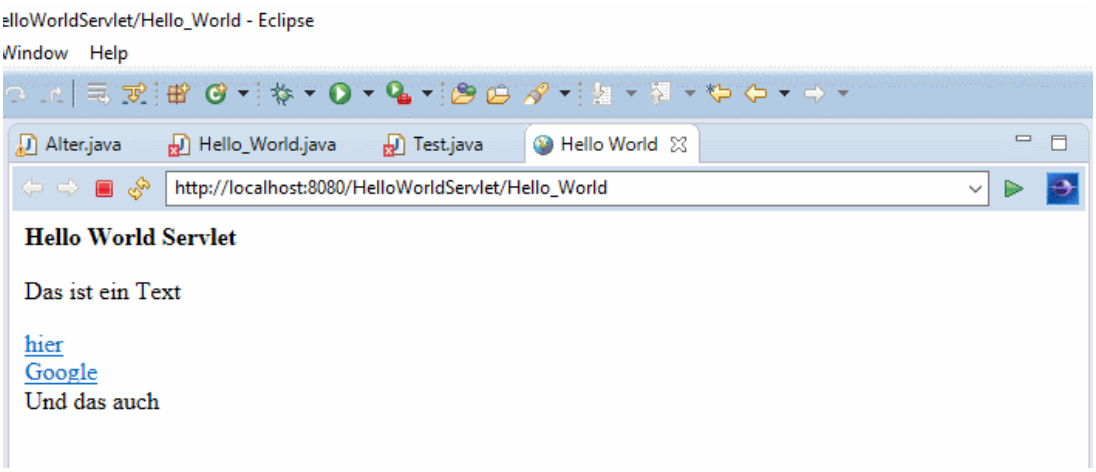

Abbildung 8: Servlet Beispiel

Wie man an diesem Beispiel sehen kann, ist das Servlet nun bereit und muss nur noch mit der gewünschten doGet-Methode befüllt werden.

2.9.3 Nutzung in unserem Projekt

In unserem Projekt selber haben wir nicht mit Java-Servlets gearbeitet, sondern haben uns stattdessen für die klassische CGI Schnittstelle, in Form von PHP, entschieden, da Robin bereits ein wenig Erfahrung mit PHP gesammelt hatte, während Java-Servlets absolutes Neuland für uns waren. Außerdem hat sich PHP für uns alle ein wenig leichter erschlossen, als die Arbeit mit den Servlets, weshalb uns PHP als der sinnvollere Weg erschien.

2.10 JSON

Jakob

JSON (= JavaScript Object Notation) ist ein Dateiformat, welches von Programmiersprachen unabhängig ist, weshalb es häufig zum Datenaustausch zwischen verschiedenen Programmiersprachen verwendet wird [Vgl. [**?**]].

Die Grundlage dieses Formats besteht in 2 Strukturen:

2.10.1 JSON-Objekt

Die erste dieser beiden Strukturen ist das JSON-Objekt, welches dazu dient Zeichenketten Werten zuzuordnen. Dies kann man sich wie folgt vorstellen:

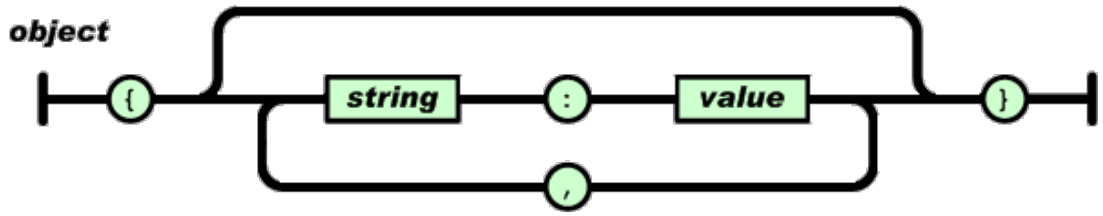

Abbildung 9: JSON-Objekt[2]

Bei der Deklaration eines JSON-Objektes muss lediglich ein Name/Wert-Paar, durch einen „:“ (Doppelpunkt) getrennt, in die geschweiften Klammern geschrieben werden, um ein Objekt zu deklarieren. Dabei ist die maximale Anzahl an Name/Wert-Paaren jedoch unbegrenzt. Einem JSON-Objekt können beliebig viele solcher Paare, durch Kommata getrennt, beigegeben werden [Vgl. [**?**]].

So sähe der Quellcode eines JSON-Objektes beispielsweise wie folgt aus:

```
1 {
2    "Name" : "Lars",
3    "Alter" : 17
4 }
```

Listing 6: Lars.json

Hier wird in der Datei „Lars.json“ ein JSON-Objekt erstellt, welches folgende Name/Wert-Paare beschreibt: Name = Lars und Alter = 17.

[2]Quelle: `http://www.json.org/object.gif`

Durch diese einfache Deklaration lässt sich ein JSON-Objekt auch problemlos in anderen Programmiersprachen erstellen, wie hier beispielsweise in JavaScript:

```
<script>
  var obj = { "Name" : "Lars, "Alter" : 17 };
  var jsonObj = JSON.stringify(obj);
</script>
```

Listing 7: JSON-Objekt in JavaScript

oder hier in Java:

```
import org.json.*;

public void jsonErstellen(){
  JSONObject obj = new JSONObject();
  obj.put("Name", "Lars");
  obj.put("Alter", 17);
}
```

Listing 8: JSON-Objekt in Java

Und hier wird die Stärke von JSON deutlich, welche in der Kompatibilität mit sämtlichen Programmiersprachen liegt, wodurch der Datentransfer zwischen diesen deutlich erleichtert wird.

2.10.2 JSON-Array

Die zweite Struktur, die JSON unterstützt ist das JSON-Array. Dieses dient, wie für ein Array üblich, der Abspeicherung von Werten, wobei auch eine Verschachtelung mit JSON-Objekten und anderen JSON-Arrays möglich ist [Vgl. [?]]. Dies lässt sich ähnlich, wie das JSON-Objekt, visualisieren:

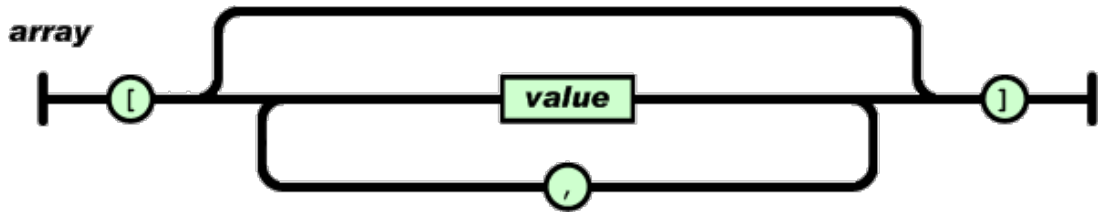

Abbildung 10: JSON-Array[3]

[3]Quelle: `http://www.json.org/array.gif`

Bei der Deklaration eines JSON-Arrays in JSON sind nur zwei Unterschiede zu der eines JSON-Objekts zu beachten, welche zum einen in den äußeren Klammern, die bei einem JSON-Array nicht geschweift sondern eckig sind, und zum andern in dem Inhalt des Arrays bestehen, da in einem Array keine Name/Wert-Paare, sondern nur Werte gespeichert werden.

Die Deklaration eines JSON-Arrays in JSON sähe also beispielsweise so aus:

```
1  [
2    {
3      "Name" : "Lars",
4      "Alter" : 17
5    }
6  ,
7    {
8      "Name" : "Robin",
9      "Alter" : 17
10   }
11 ]
```

Listing 9: Klasse.json

Und natürlich ist es auch problemlos möglich ein JSON-Array in anderen Programmiersprachen zu deklarieren, jedoch möchte ich an dieser Stelle auf Beispiele verzichten.

2.10.3 Nutzung in unserem Projekt

Trotz des einfachen Umgangs mit JSON haben wir uns gegen die Arbeit mit diesem Dateiformat entschieden, was der einfachen Tatsache geschuldet ist, dass wir ein webbasiertes Programm mit einer klassischen CGI-Schnittstelle programmieren wollten, weshalb unsere forcierten Programmiersprachen HTML und PHP waren. Nun hätten wir zwar den Datentransfer zwischen diesen Sprachen auch mit JSON bewerkstelligen können, jedoch ist dieser zwischen HTML und PHP sowieso durch die Post- und Get-Methoden beider Sprachen [Vgl. [2.3.3] und [**??**]] sehr einfach, weshalb wir an der Stelle diesen Weg eingeschlagen haben. Jedoch hätten wir JSON benutzt, wenn es unsere Absicht gewesen wäre, die Dateien in der MYSQL-Datenbank abzulegen und nicht nur die Referenz zu der Datei, aber auf dieses Thema wollen wir erst in Kapitel 3 ausführlich eingehen [siehe [3.4.1]].

2.11 SQL

Jakob

SQL (= Structured Query Language, d.h. Strukturierte Abfrage-Sprache) ist eine sogenannte „Datenbanksprache“ und somit für die Definition von Datenstrukturen in relationalen Datenbanken zuständig [Vgl. [**?**]]. Das heißt, das über SQL die Einrichtung und die Verwaltung einer Datenbank, wie zum Beispiel einer MySQL-Datenbank, möglich ist.

2.11.1 MySQL

MySQL ist ein Datenbanksystem, das ohne jede grafische Oberfläche, ausschließlich durch die Sprache SQL definiert und verändert werden kann [Vgl. [**?**]].

Eine solche MySQL-Datenbank besteht hauptsächlich aus sogenannten „Schemas“, welche die eigentliche Datenbank, im umgangssprachlichen Sinne, in Form von, durch sie gruppierte, „Tabellen“, darstellt und ebenjene „Tabellen“, in welchen die gewünschten Daten abgespeichert werden können.

2.11.2 Datenbankentwurf

Da eine solche Datenbank[4] dazu da ist effektiv Daten abzuspeichern, sodass diese auch leicht abrufbar sind, ist eine genaue Planung einer solchen Datenbank ein wichtiger Bestandteil der Erstellung einer solchen.

Dazu dienlich sind die sogenannten „ER-Diagramme“, welche eine visuelle Planung einer MySQL-Datenbank ermöglichen, da diese selber keine grafische Oberfläche besitzt. [siehe [2.11.1]].

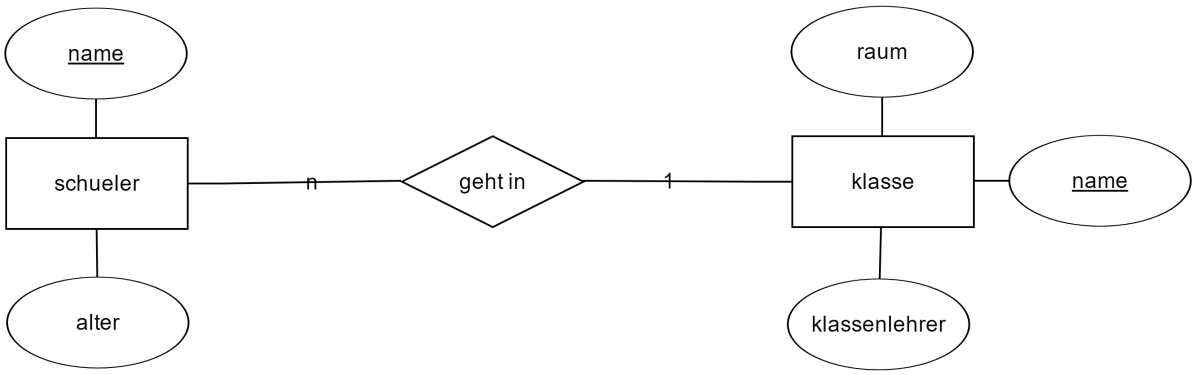

Abbildung 11: ER Diagramm

In Abbildung 11 kann man gut erkennen, dass ER-Diagramme genauso simpel zu handhaben sind wie die Sprache SQL. Sie stellen lediglich die Struktur eines Schemas in Form

[4] Da MySQL das am weitesten verbreitete Datenbanksystem ist, werde ich im Folgenden auch ausschließlich darauf eingehen, wenn ich von Datenbanken schreibe

der Tabellen zueinander dar, indem die Einträge der Tabellen, sogenannte Entitäten, als Rechtecke, [Vgl. Abbildung 12],

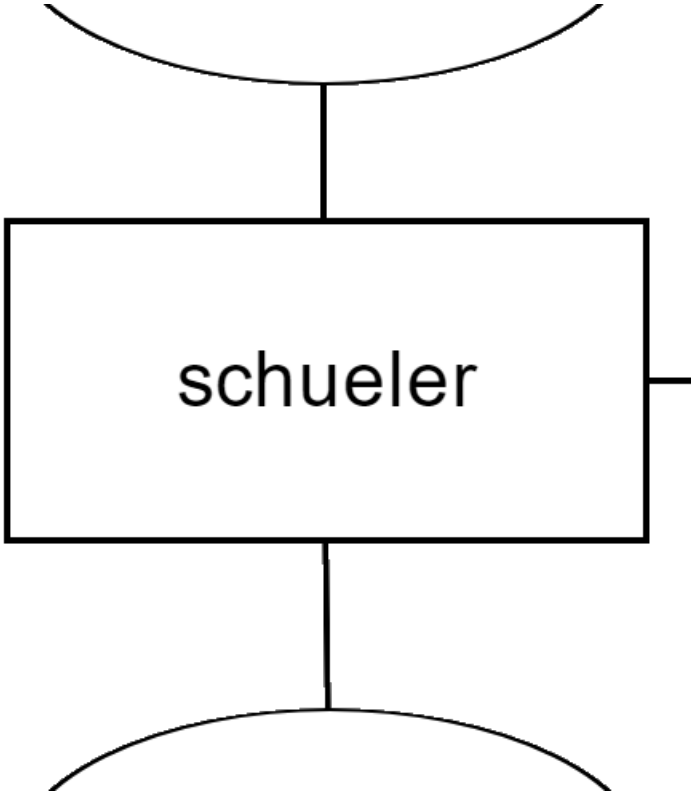

Abbildung 12: ER Diagramm Entitaet

und ihre Beziehungen zueinander, als Rauten, [Vgl. Abbildung 13],

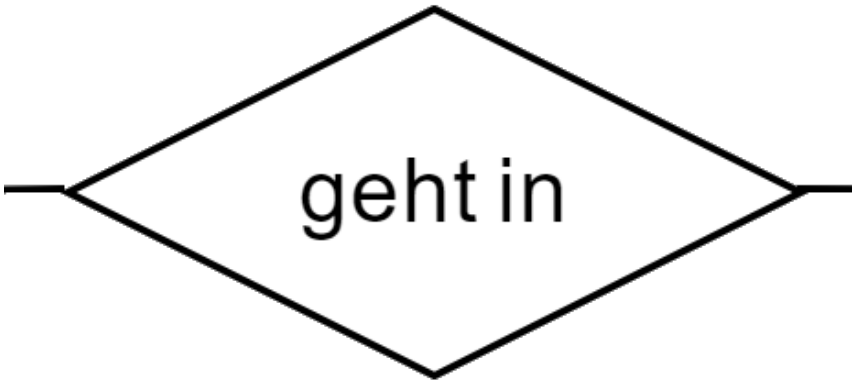

Abbildung 13: ER Diagramm Beziehung

visualisiert werden.

Zusätzlich zu dieser Struktur können noch Angaben über die Attribute jeder Entität (=

Spalten der jeweiligen Tabelle)

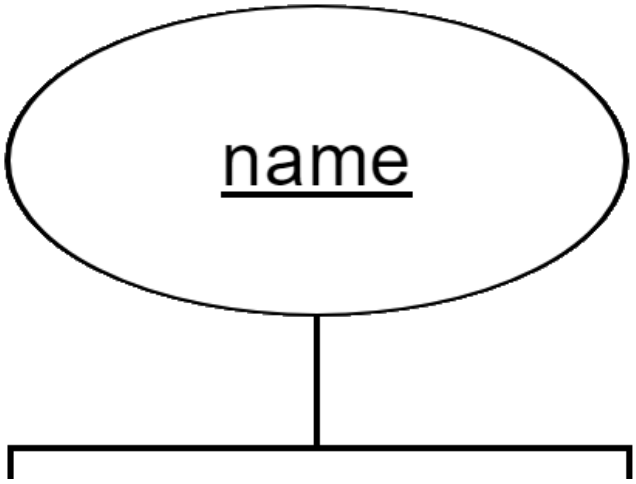

Abbildung 14: ER Diagramm Attribut

und über die Kardinalität der Beziehung zweier Entitäten zueinander gemacht werden.

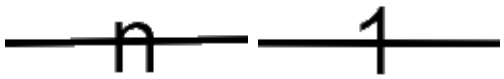

Abbildung 15: ER Diagramm Kardinalitaeten

Dabei findet sich die Kardinalität in Form von den Ziffern „1“ und „n“ beziehungsweise „m“ [Vgl. Abbildung 15] , wobei „1“ für genau eine und „n“/„m“ für eine unbestimmte Anzahl an möglichen Entitäten in der betrachteten Beziehung steht. So findet sich in dem gewählten Beispiel Schule [siehe Abbildung 11] die Beziehung „geht in“ zwischen Schüler und Klasse. Wobei eine Klasse natürlich mehrere Schüler umfasst, während ein Schüler immer nur eine Klasse besuchen kann, wodurch diese „1:n-Kardinalität“ zustande kommt.

Und diese Kardinalitäten spielen in der späteren Implementierung eine große Rolle, wozu ich aber im nächsten Abschnitt kommen möchte.

2.11.3 SQL-Statements

Der Schlüssel für die Verwaltung einer solchen Datenbank sind die „Statements“ von SQL, welche einzeilige Befehle und Abfragen darstellen.

Diese Statements lassen sich in drei Kategorien einteilen:

[4]Da MySQL das am weitesten verbreitete Datenbanksystem ist, werde ich im Folgenden auch ausschließlich darauf eingehen, wenn ich von Datenbanken schreibe

DDL Unter der Abkürzung DDL versteht man Statements, die dazu dienen das Datenbankschema zu definieren [Vgl. [**?**]]. Dazu gehören die „Create-“ und „Drop-Befehle“, die für das Erstellen und Löschen von Schemas und Tabellen in diesen zuständig sind.

Die Syntax dieser Befehle ist, wie die der gesamten Sprache SQL sehr simpel und einzeilig gehalten. So sähe ein „Create-Befehl“ zur Erstellung des Schemas „schule“, welches wir vorher in dem ER-Diagramm geplant hatten [siehe [**??**]], wie folgt aus:

```
1 CREATE DATABASE schule;
```

Listing 10: SQL Create Schema

und der „Drop-Befehl“ zum Löschen derselben folgt der gleichen Syntax:

```
1 DROP DATABASE schule;
```

Listing 11: SQL Drop Schema

Nachdem wir das Schema erfolgreich erstellt und nicht gelöscht haben können wir es nun mit den passenden Tabellen füllen.
Vorausgesetzt für das erfolgreiche Erstellen einer Tabelle ist zunächst, dass man sich mit einem, in der angesprochenen Datenbank vorhandenen Schema, verbunden hat, wie dies zum Beispiel mit PHP funktioniert lege ich in dem dazugehörigen Abschnitt dar [siehe [2.11.4]]. Wenn dies geschehen ist, dann ist das Erstellen einer solchen Tabelle ebenfalls über ein „Create-Statement“ realisierbar jedoch sieht dieses bei einer Tabelle etwas komplexer aus, da in dem Statement die Namen und Datentypen der sogenannten „columns“ (zu Deutsch: Spalten), wie auch der „primary key“, das ist das einzigartige Erkennungsmerkmal einer jeden Zeile der entsprechenden Tabelle, definiert werden müssen.

So könnte der Befehl zur Erstellung der Tabelle „schueler“, aus unserem ER-Diagramm [siehe [**??**]], in der SQL-Syntax zum Beispiel wie folgt aussehen:

```
1 CREATE TABLE schueler (
2  name VARCHAR(40) NOT NULL,
3  klassenname VARCHAR(2) NOT NULL,
4  alter INT(2) NOT NULL,
5 PRIMARY KEY (name)
```

```
6 );
```

Listing 12: SQL Create Table

Mit diesem Befehl würde nun folgende Tabelle erzeugt werden:

name	**klassenname**	**alter**
...	...	...

Tabelle 3: SQL Schueler

Wobei die Inhalte der Spalte „Name“ den Primary Key, also das Erkennungsmerkmal, in Form einer Zeichenkette mit der maximalen Zeichenanzahl von 40 darstellt, die Spalte „klasse“ ebenfalls Inhalte des Datentyps VarChar (= Zeichenkette) speichert, jedoch dürfen diese nur maximal eine Länge von zwei Zeichen erreichen. Lediglich die Spalte „Alter“ grenzt sich im Datentyp von den Anderen ab, da das Alter logischerweise in Form eines Integers gespeichert wird, der hier auf zwei Zeichen beschränkt ist, da ein Schüler selten ein dreistelliges Alter besitzt. Eine zu erwähnende Besonderheit, die sich alle drei Spalten hier teilen ist jedoch, dass sie nicht leer sein dürfen, was durch den Befehl „NOT NULL“ erreicht wird.

An dieser Stelle sollten wir uns jedoch an unser Er-Diagramm erinnern [siehe [**??**]] und uns die Beziehungen der Entität Schüler nochmal anschauen. Und da wird deutlich, dass das Attribut „klassenname“ in der Entität „schueler“ nicht nur eine Zeichenkette, sondern ein Verweis auf eine Entität „klasse“ ist. Und, um das umzusetzen, müssen wir uns eines sogenannten „Foreign Key“(= Fremdschlüssel) bedienen. Dies können wir ganz einfach, vorausgesetzt wir haben bereits eine solche Tabelle „klasse“ erstellt:

bezeichnung	**klassenlehrer**	**raum**
...	...	...

Tabelle 4: SQL Klasse

Wobei das Attribut „bezeichnung“ der Primärschlüssel ist. Denn, wenn dies der Fall ist können wir unser „Create-Statement“ der Tabelle „schueler“ einfach dahingehend ändern, dass das Attribut „klasse“ als Fremdschlüssel erkannt wird, sodass er auch aktualisiert wird, wenn Änderungen an der Entität vorgenommen werden, auf die er verweist. Diese Prävention erreicht man mit einem „Constraint-Befehl“ erreicht:

```
1  CREATE TABLE schueler (
2  name VARCHAR(40) NOT NULL,
3  klassenname VARCHAR(2) NOT NULL,
4  alter INT(2) NOT NULL,
5  PRIMARY KEY (name),
6  KEY klasse2 (klassenname),
7  CONSTRAINT klasse2 FOREIGN KEY (klassenname)
8    REFERENCES klasse (bezeichnung)
9      ON DELETE CASCADE ON UPDATE NO ACTION
10 );
```

Listing 13: SQL Create Table + Constrait

Mit diesem Befehl wird zwar im Prinzip die gleiche Tabelle erzeugt, wie im ersten Beispiel, jedoch beugt diese Methode mit dem „Constraint-Befehl“ Anomalien vor, indem er dafür sorgt, dass der Fremdschlüssel gelöscht wird, wenn die Entität, die das Ziel dieser Referenz war, entfernt wird (siehe „ON DELETE CASCADE“).

Das Löschen einer Tabelle zeigt sich nun, im Gegensatz zum Erstellen einer solchen, ohne jede Besonderheit, syntaktisch genauso, wie auch das Löschen eines „schemas“:

```
1 DROP TABLE schueler;
```

Listing 14: SQL Drop Table

So würde das „Drop-Statement“, welches in Listing 14 dargestellt wird, die Tabelle „schueler“ löschen.

DML Die Abkürzung DML beschreibt, im Gegensatz zu der, eben erläuterten, Abkürzung DDL, die Arbeit mit bereits erstellten Tabellen. Womit sie den zentralen Teil der Datenspeicherung in einer solchen Datenbank darstellen, da ihnen die Aufgabe zufällt Daten abzulegen, zu suchen und zu löschen [Vgl. [**?**]].

Dies realisiert SQL mit 3 zentralen Befehlen: „Insert Into“, „Select From“ und „Delete From“.

Diese Befehle sind ebenjenen Aufgaben zugeordnet, welche ihr Name umschreibt, woraus zu schließen ist, dass der „Insert-Befehl“ zum Einfügen, der „Select-Befehl“ zur Abfrage und der „Delete-Befehl“ zur Löschung von Daten ist.

Zunächst müssen also Datensätze, sogenannte „Entitäten“ in die Tabellen „schueler“ und „klasse“ gelangen, was mit dem „Insert-Befehl“ wie folgt möglich ist:

```
1 INSERT INTO klasse
2   (bezeichnung, klassenlehrer, raum)
3     VALUES
4       ("5b", "Herr Bachran", "209");
```

Listing 15: SQL Insert Into Table

```
1 INSERT INTO schueler
2   (name, klassenname, alter)
3     VALUES
4       ("Lars", "5b", 17);
```

Listing 16: SQL Insert Into Table 2

Nun haben wir jeweils einen Datensatz in die vorher erstellten Tabellen eingefügt, welche nun wie folgt aussehen:

bezeichnung	**klassenlehrer**	**raum**
5b	Herr Bachran	209

Tabelle 5: SQL Klasse mit Datensatz

name	**klassenname**	**alter**
Lars	5b	17

Tabelle 6: SQL Schueler mit Datensatz

Um auf diese Datensätze zugreifen zu können, kann man sich nun des „Select-Statements“ bedienen, welches sich, wie die anderen Statements auch, vom syntaktischen Aufbau an der englischen Grammatik orientiert. Um also den Namen von dem Schüler „Lars“, welcher in der Tabelle „schueler“ abgelegt wurde, ausgegeben zu bekommen, kann man sich folgendem „Select-Statements“ bedienen:

```
1 SELECT name FROM schueler;
```

Listing 17: SQL Select Name

Wenn jedoch die Tabelle „schueler“ wie folgt aussehen würde:

name	**klassenname**	**alter**
Lars	5b	17
Robin	7c	17

Tabelle 7: SQL Schueler mit Datensätzen

Sprich, mehr, als nur einen Datensatz besitzen würde, müssten wir unsere Abfrage weiter eingrenzen, um das selbe Ergebnis zu erhalten, was durch die Erweiterung des „Select-Statements“ durch den Befehl „Where“ umsetzbar ist.

```
1 SELECT name FROM schueler WHERE klassenname LIKE '5b' AND alter Like 17;
```

Listing 18: SQL Select Name + Where

Wie man im Listing 19 erkennen kann leitet der „Where-Befehl“ lediglich eine Reihe von Bedingungen ein, welche von dem Abfrage-Ergebnis erfüllt werden müssen. Auf diese Weise lässt sich der „Select-Befehl“ mit den verschiedensten Befehlen erweitern, mit denen sich die Anfrage-Ergebnisse unter anderem auch sortieren, beziehungsweise gruppieren lassen, jedoch möchte ich an dieser Stelle nicht zu sehr ins Detail gehen.

Wenn wir nun aber einen Datensatz aus der Tabelle entfernen wollen, können wir uns des „Delete-Befehls“ bedienen. Da jedoch vor jedem Löschvorgang zunächst der zu löschende Datensatz gefunden werden muss, ist es nur logisch, dass der „Delete-Befehl“, genauso, wie eben bei dem „Select-Befehl“ dargestellt, durch sämtliche Suchkriterien erweiterbar ist, wodurch er im Prinzip eine Kopplung aus einer Abfrage und einem Löschvorgang ist.

Um dies nun an einem Beispiel zu demonstrieren werde ich einfach mal die „5b“ aus folgender Tabelle „klasse“ entfernen:

bezeichnung	**klassenlehrer**	**raum**
5b	Herr Bachran	209
7c	Herr Scholl	231

Tabelle 8: SQL Klasse mit Datensätzen

Der dazu dienliche SQL-Befehl, sähe also wie folgt aus:

```
1 DELETE FROM klasse WHERE bezeichnung LIKE '5b';
```

Listing 19: SQL Select Name + Where

Nachdem dieser Befehl ausgeführt wurde sehen unsere beiden Tabellen dann so aus:

bezeichnung	**klassenlehrer**	**raum**
7c	Herr Scholl	231

Tabelle 9: SQL Klasse mit Datensatz nach Delete

name	**klassenname**	**alter**
Robin	7c	17

Tabelle 10: SQL Schueler mit Datensatz nach Delete

An der Stelle sollte uns nun auffallen, dass neben der Klasse 5b auch der Schüler Lars gelöscht worden ist, obwohl wir dies gar nicht veranlasst haben. Dies ist auf den „Constraint-Befehl“ zurückzuführen, den wir bei der Definition der Tabelle „schueler“ verfasst haben. Dieser beinhaltete nämlich die Bedingung, dass der Fremdschlüssel ebenfalls gelöscht wird, wenn das Ziel dessen REferenz entfernt wird und da er „Not Null“ sein darf, wurde der gesamte Datensatz Lars gelöscht.

DCL Die „Data Control Language“ ist für die Rechteverwaltung innerhalb einer Datenbank zuständig [Vgl. [**?**]]. Da sie aber in unserem Projekt nicht verwendet worden ist und auch nicht direkt mit der Organisation von Datensätzen in einer Datenbank zusammenhängt, möchte ich es an dieser Stelle bei dieser kurzen Erwähnung belassen.

2.11.4 Verwaltung einer Datenbank mit PHP

Wie bereits oben erwähnt, lässt sich eine Datenbank mithilfe der Sprache PHP [2.4] ansprechen. Dies lässt sich über das Objekt „mysqli“ umsetzen, welches die Verbindung zur Datenbank herstellt und wie folgt deklariert wird:

```
1 <?php
2 $db = new mysqli(<MYSQL_HOST>, <MYSQL_BENUTZER>, <MYSQL_KENNWORT>, <
      MYSQL_DATENBANK>);
3 ?>
```

Listing 20: SQL PHP Mysqli

Dabei sind die in Großbuchstaben verfassten Variablen Platzhalter für die Verbindungsinformationen zu der Datenbank, mit der man sich verbinden möchten, mit der optionalen Angabe eines Schemas, welche man hinzufügt, wenn man an Tabellen arbeiten will.

Um nun einen SQL-Befehl anzuwenden muss man diesen erstmal in SQL übersetzen und an die Datenbank schicken, was mit PHP in einer Zeile zu implementieren ist:

```
1 <?php
2 $erg = $db->query("SELECT * FROM schueler WHERE name LIKE 'Robin'");
3 ?>
```

Listing 21: SQL PHP Mysqli

Als Ergebnis wird nun zunächst ein Array in der Variablen „erg“ gespeichert, welches lediglich Daten darüber enthält, was alles von der Abfrage angesprochen wurde.

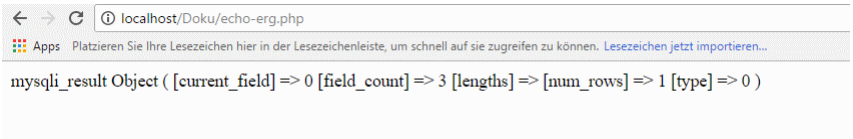

Abbildung 16: Ausgabe des Arrays

Dies kann man sich zu Nutze machen, indem man über eine if-Verzweigung abfragt, ob es überhaupt einen betroffenen Datensatz, sprich ein Ergebnis, gibt:

```
1 <?php
2 if($erg->num_rows){
3 echo 'Ergebnis vorhanden';
4 }
5 ?>
```

Listing 22: SQL PHP NumRows

Wenn also ein Ergebnis vorhanden ist, lässt sich das Array mit dem Befehl „fetch all“ dahingehend übersetzen, dass die gewünschten Ergebnisse angezeigt werden. Natürlich ist dies nur bei einer „Select-Abfrage“ sinnvoll, da ansonsten ja keine Datensätze zurückgegeben werden.

```
1 <?php
2 $array = $erg->fetch_all(MYSQLI_ASSOC);
3 ?>
```

Listing 23: SQL PHP Mysqli

Nun sähe die Ausgabe dieses Arrays wie folgt aus:

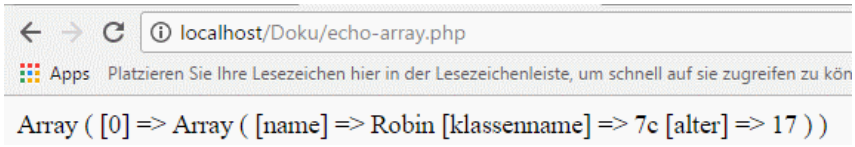

Abbildung 17: Ausgabe des Datensatzes

Diese Vorgehensweise habe ich mir mithilfe einer Webseite zum Thema PHP angeeignet, da ich zwar mit SQL umgehen konnte, jedoch noch nie zuvor mit PHP programmiert habe [Vgl. [?]]

2.11.5 Nutzung in unserem Projekt

Die Datenbanksprache SQL und die, mit ihr verwaltete MySQL Datenbank, spielen eine sehr bedeutende Rolle in unserem Projekt, da mit ihnen die Daten der Benutzer und der Files verwaltet werden, wodurch eine sortierte Ablage, mit möglicher Abfrage erst möglich wird. Mehr zu diesem Teil von unserem Projekt finden Sie in Kapitel 3 [siehe [3.4]].

3 Projekt

Lars

In diesem Kapitel wird nun unser Projekt dargestellt. Zunächst wird das Programm im Gesamten beschrieben. Anschließend werden die einzelnen Teile getrennt dargestellt.

3.1 Datei-Verwaltungs-Programm

Lars

Seit Beginn des Projekts im September 2016 sind neun Monate vergangen und das Programm umfasst nun ein Anmeldungsverfahren inklusive Registrationsmöglichkeit, die Möglichkeit Dateien hoch zu laden und nach diesen zu suchen. Im Hintergrund arbeitet ein Server, der sämtliche Anfragen des GUIs annimmt und entsprechend weiter leitet. Zusätzlich verfügt unser Programm über eine Datenbank, welche die Pfade zu den Dateien, sowie ihre Details, also zum Beispiel die Größe und den Dateityp, speichert.

3.2 GUI

Lars

GUI ist die Abkürzung für „Graphical User Interface“, was zu Deutsch „Graphische Benutzeroberfläche“ bedeutet. Als GUI wird also das bezeichnet, was der Benutzer von einem Programm sieht.

3.2.1 Benutzererfahrung

Bei der Verwendung unseres Programms muss sich der Benutzer zunächst anmelden. Das sich öffnende Anmeldefenster kann man in Abbildung 18 sehen. Falls der Benutzer noch kein Benutzerkonto besitzt, kann er sich ein solches erstellen. Nach einer erfolgreichen Anmeldung wird der Benutzer auf die Hauptseite (siehe Abbildung 19), den Kern des Programms, weiter geleitet. Hier kann er auf der linke Seite Dateien hochladen, indem er sie entweder per Auswahlfenster öffnet oder per Drag and Drop dem Eingabefeld übergibt. Optional können den hochzuladenden Dateien Beschreibungen hinzufügt werden. Auf der rechten Seite kann der Benutzer seine Dateien, die er bereits hochgeladen hat verwalten. Dazu ist zum Zeitpunkt der Verfassung dieses Texts nur eine Suchfunktion vorhanden. Es werden noch eine Such-, Lösch- und Änderungsfunktion folgen.

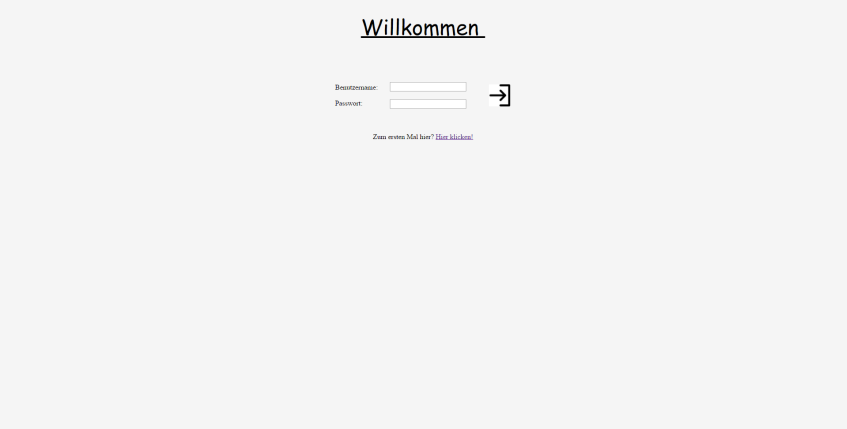

Abbildung 18: Aufbau der Anmeldung

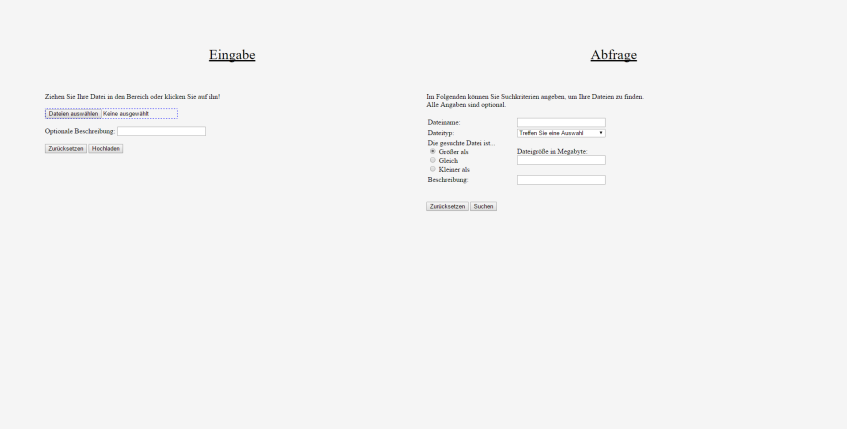

Abbildung 19: Aufbau der Hauptseite

3.2.2 Realisierung der Hauptseite

Angefangen hat alles mit einem simplen Quellcode, zum Hochladen einer Datei per HTML. Dieser ist im Folgenden Dargestellt. (Listing 24) Er bestand lediglich aus einer Aufforderung, die Datei, welche der Benutzer hochladen möchte einzufügen, einem Eingabefeld und einem Knopf mit dem Namen „Senden“. Aus diesem Code lernten wir, besonders ich, wie das Hochladen von Dateien per HTML funktioniert.

```
1 <html>
2   <head>
3     <title>
4       Dateiupload
5     </title>
6   </head>
7   <body>
8     Bitte f&uumlgen Sie die Datei ein, die Sie hochladen m&oumlchten!
```

```
9       <form method="post" action="upload.php" enctype="multipart/form-data
            ">
10      Datei:
11        <input type="hidden" name="MAX_FILE_SIZE" value="100000">
12        <input type="file" name="datei" size="40" maxlength="100000">
13        <input type="submit" name="Submit" value="Senden">
14      </form>
15    </body>
16  </html>
```

Listing 24: Erste Version des Datei-Uploads

Nach Sammlung von Inspiration auf der Internetseite `https://jqueryui.com` wurde die Hauptseite [19] entworfen und programmiert. Zunächst war die Idee, unterhalb der Eingabe und Abfrage das Suchergebnis zu platzieren. Dies stellte sich aber als eher unpraktisch heraus, weshalb wir beschlossen, das Frame [2.3.1], was zuvor nur die Abfrage enthielt, nun auch das Suchergebnis ausgeben zu lassen. Der Plan war nun also kurz gesagt, mithilfe des Formulars [2.3.3], das die Suchabfrage aufnimmt, auf ein PHP-Dokument [2.4] zu verweisen, welches schlussendlich das HTML-Dokument[2.3] mit dem Ergebnis aufruft.

3.2.3 Realisierung der Anmeldung

Mit der Zeit entstand die Idee der Gestaltung eines Anmeldeverfahrens. Dieses sollte aus einem Benutzernamen und einem Passwort bestehen. Als die Anmeldung funktionierte fiel uns auf, dass natürlich auch eine Registrierung notwendig sein wird. Auch sie besteht aus einem Feld zur Eingabe eines Benutzernamens. Jedoch beinhaltet die Registrierung zwei Felder für die Eingabe des Passworts, um zu gewährleisten, dass sich der Benutzer dass richtige Passwort auch tatsächlich merkt oder zumindest notiert. Der Nachteil beim einmaligen Eintippen des Passworts bei der Registrierung ist, dass der Benutzer sich eventuell vertippen kann, sich also ein falsches Passwort merkt beziehungsweise notiert und somit später keinen Zugriff mehr auf seine Dateien erhält.

3.3 Engine I: Interface und Datei-System

Robin

Dieser Teil der Engine ist für den Upload zuständig. Die üpload.phpDatei bekommt die hochzuladene Datei durch eine POST-Methode von der ëingabe.htmlDatei. Die Uplaod datei ließt nun die benötigten Daten aus der Datei heraus wie z.B. Name, Größe, Datei-Typ und Erstellung-/Änderungsdatum. Diese Informationen werden dann zusammen mit dem neuen Pfand in der Datenbank abgespeichert. Danach wird die ëingabe.htmlDatei

wieder aufgerufen um wieder auf den vorherigen Frame zu kommen. Zunächt war geplant, die üpload.phpDatei in die ëingabe.htmlßu integrieren, doch später habe ich mich dazu entschieden, den Upload extern zu machen, da es so einfacher zu schreiben ist und übersichtlicher in der Datei selbst.

3.4 Engine II: Interface und Datenbank

Jakob

Der zweite Teil unserer Engine besteht in der MySQL Datenbank, welche die Aufgabe besitzt die Metadaten und die Dateipfade der abgelegten Files, sowie auch die Anmeldedaten der Benutzer zu verwalten.

3.4.1 Planung

Zunächst bestand die Datenbank lediglich aus einem Schema „filesystemreferences“ beziehungsweise „fs“, in welchem eine gleichnamige Tabelle vorlag, in der alle, über das Programm hochgeladene, Files in Form von deren Dateipfad und Metadaten mit einer optionalen Beschreibung dokumentiert wurden. Als jedoch die Möglichkeit der Anmeldung dazukam, war dies keine Option mehr und das Datenbank Konzept musste erweitert werden. Aus diesem Grund habe ich dazu entschieden ein Schema „user“ in die Datenbank zu integrieren, in dem die Benutzer mit Namen, Passwort und dem Namen ihres „fs-Schemas“ verwaltet werden. Und dieser Verweis auf ein anderes Schema führt mich zu der größten Änderung an meinem Konzept: Ich habe mich nämlich dazu entschieden nicht nur ein Schema für die Files zu verwenden, sondern für jeden Benutzer ein eigenes anzulegen, damit jeder nur auf seine Files zugreifen kann.

Von Anfang an zurückgewiesen haben wir jedoch die Idee, alle Files, als JSON-Datei selber in der Datenbank abzuspeichern, da wir dem Benutzer die Möglichkeit offen lassen wollten auch über andere Anwendungen, wie zum Beispiel den Windows Explorer, die abgelegten Dateien zu finden.

3.4.2 Umsetzung

Um die Datenbank automatisch einrichten und verwalten lassen zu können habe ich mich dafür entschieden über PHP mit der Datenbank zu interagieren, da der Datentransfer zwischen HTML und PHP sehr leicht umsetzbar ist. Wie dies im Detail funktioniert, habe ich bereits in Absatz 2.11.4 erläutert. An dieser Stelle möchte ich lediglich nochmal die Struktur darstellen, in der ich die Verwaltung der Datenbank angeordnet habe. Hinter jeder PHP-Datei steckt bei mir zunächst meine „ind.php“, welche die Verbindungsinformationen zu meiner Datenbank speichert:

```
1  <?php
2  error_reporting(E_ALL);
3
4  define ( 'MYSQL_HOST',       'localhost' );
5
6  define ( 'MYSQL_BENUTZER',   'root' );
7
8  define ( 'MYSQL_KENNWORT',   '' );
9
10 ?>
```

Listing 25: Ind.php

Auf die Variablen, die in dieser Datei zur Vermeidung von Redundanzen, ausgelagert wurden, greifen die Dateien „konfiguration.php“, „registration.php“ und „anmeldung.php“, die alle Teil des Anmeldeverfahrens sind, zu. Das Anmeldeverfahren ist kompliziert zu erklären, weshalb ich hier zunächst eine Grafik dazu zeigen möchte.

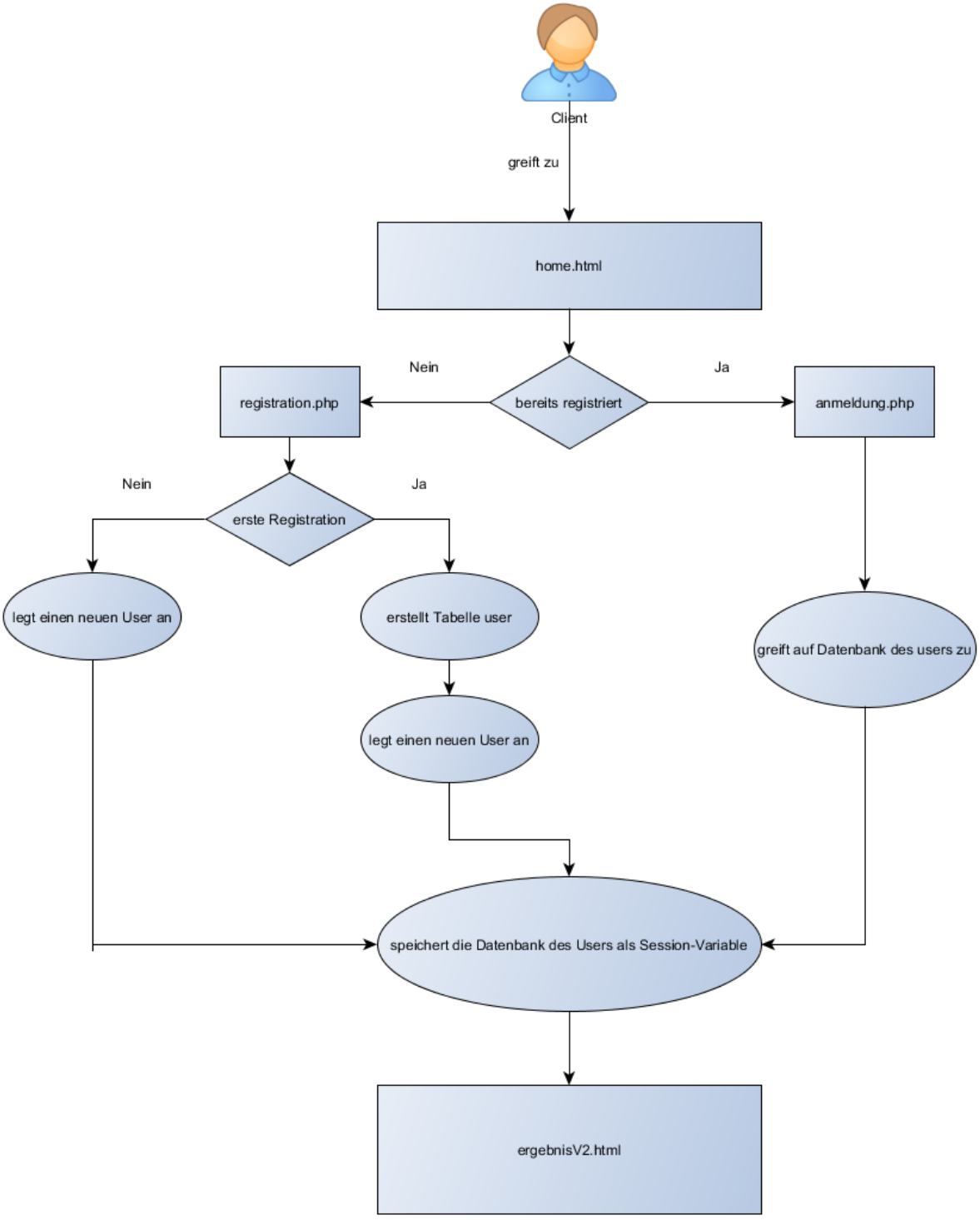

Abbildung 20: Anmeldeverfahren

In dieser Grafik wird dargestellt, wie das Programm mit der Anmeldung eines Benutzers umgeht, wobei ich darauf verzichtet habe die Ausgaben der Fehlermeldungen, bei falschen Angaben des Clients, mit in das Diagramm einzubinden.

Der entscheidende Punkt der Anmeldung ist die Speicherung des Namens der Datenbank des eingeloggten Benutzers als Session-Variable, da die Datei „konfiguration.php“ auf ebenjene zugreift:

```
1  <?php
2  session_start();
3
4  require('ind.php');
5
6  error_reporting(E_ALL);
7
8  $database = $_SESSION[ 'database' ];
9  define ( 'MYSQL_DATENBANK', ''.$database);
10 ?>
```

Listing 26: Konfiguration.php

Somit integriert diese PHP-Datei alle Verbindungsinformationen, die ein „mysqli-Objekt“ [2.11.4] benötigt, um auf die Tabelle eines bestimmten Schemas zuzugreifen. Und damit ist die Funktion dieser Datei recht offensichtlich: Nach der Anmeldung beziehen sich sämtliche SQL-Statements nur noch auf die Verbindungsinformationen, welche die „konfiguration.php“ mitgibt, damit jeder User nur auf die eigenen Files zugreifen kann. Die restlichen Zugriffe auf die Datenbank sind einfache Abfragen/Befehle, wie ich sie bereits dargestellt habe [siehe [2.11].

4 Fazit

Lars

Im Folgenden wird jeder von uns ein Fazit geben, indem er darstellt, was er aus dem Projekt gelernt hat, Dinge die ihm gefielen und so weiter.

4.1 Lars

Im Verlaufe des Projektes ist mir deutlich geworden, dass mir das Programmieren sehr viel Spaß macht und ich mich auch nach der Schule in Richtung IT orientieren werde.
Ferner habe ich festgestellt, dass Teamarbeit nicht leicht ist. Insbesondere, wenn jedes Gruppenmitglied den anderen im Team gleich gestellt ist, es also keine Gruppenleitung gibt. Mit der Zeit wurde zwar die Leitung übernommen. Jedoch entstand dann das Problem, dass diese Leitung nicht autorisiert war. Im Falle einer Autorisierung, zum Beispiel durch Wahl, hätte es die Möglichkeit gegeben, Leitungsaufgaben wie Delegieren, konsequent verfolgen zu können.
Insgesamt habe ich die Arbeitsstruktur, die durch den Lehrer vorgegeben war, als sehr angenehm empfunden. Ich habe es sehr geschätzt, dass uns sehr viele Freiheiten gelassen wurden und dass es keine festen Vorgaben gegeben hat, an die wir halten mussten. Diese Arbeitsstruktur hat zu einer sonst guten Arbeitsatmosphäre beigetragen.

4.2 Robin

...

4.3 Jakob

Wie bereits eingangs mehrfach erwähnt, bestand die Schwierigkeit von unserem Projekt, meiner Meinung nach, nicht in der Thematik, sondern in der Arbeit miteinander. Dadurch, dass wir zu dritt an einem solchen Projekt, welches viel größer ist, als alles Andere, was wir bis dahin programmiert haben, gearbeitet haben, sind wir auf 2 Ebenen auf ganz neuem Terrain gewandert. Zum einen war natürlich die Thematik und die Art des Programmierens für uns absolut ungewohnt. Viel schwerwiegender war jedoch die Tatsache, dass wir bis dato noch nie so frei an einem Projekt gearbeitet haben, welches es gemeinsam zu verfolgen galt. Und genau in den Problemen, die daraus entstanden, sehe ich den größten und essentiellsten Lerneffekt, den wir mitnehmen sollten. Denn in diesem Schuljahr konnten wir einen ersten Blick in das Leben nach der Schule werfen, durften feststellen, wie es ist nicht mehr an der Hand gehalten und behütet durch die Aufgaben geführt zu werden. Wir mussten uns zwar erst langsam an diese Situation gewöhnen und haben letztendlich auch kein großartiges Projekt auf die Beine stellen können, jedoch bin ich froh und

dankbar, dass ich diese Erfahrung des „Unbehütet-Seins“ in einer behüteten Umgebung machen durfte. Aus diesem Grund bin ich froh mich für diesen Projektkurs entschieden zu haben und überzeugt davon, dass er uns besser auf das „richtige Leben“ vorbereitet hat, als so manch anderes Fach, wenn man bereit ist sich seine Fehler einzugestehen und daraus Konsequenzen zu ziehen.

5 Literaturverzeichnis

[apa] apache.org. https://httpd.apache.org/. Zugriff: 18.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rJqu0IFq.

[itw] itwissen.info. http/www.itwissen.info/JavaScript-JavaScript-JS.html. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrxwvIa.

[Jso] Json.org. http://www.json.org/json-de.html. Zugriff: 18.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rJM0jEv9.

[Omk] Omkt.de. http://www.omkt.de/mysql/. Zugriff: 19.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rMWky2Wf.

[Pra] Axel Pratzer. https://www.php-kurs.com/. Zugriff: 20.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rMgss0mY.

[sea] searchenterprisesoftware.de. http://www.searchenterprisesoftware.de/definition/JavaScript. Zugriff: 15.06.2017.

[sela] selfhtml.org. https://wiki.selfhtml.org/wiki/HTML. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrS8Nzd.

[selb] selfhtml.org. https://wiki.selfhtml.org/wiki/HTML/Dokumentstruktur_und_Aufbau. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrkA67U.

[selc] selfhtml.org. https://wiki.selfhtml.org/wiki/HTML/Kopfdaten. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrgojtp.

[seld] selfhtml.org. https://wiki.selfhtml.org/wiki/HTML/Frames. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrhMYl6.

[sele] selfhtml.org. https://wiki.selfhtml.org/wiki/HTML/Formulare/Form. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrygHFX.

[self] selfphp.de. http://www.selfphp.de/praxisbuch/praxisbuchseite.php?site=183. Zugriff: 19.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rJt3l5hw.

[Wika] Wikipedia.org. https://de.wikipedia.org/wiki/LaTeX. Zugriff: 17.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHyFh5A9.

[Wikb] Wikipedia.org. https://de.wikipedia.org/wiki/Leslie_Lamport. Zugriff: 17.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHyFh5A9.

[Wikc] Wikipedia.org. https://de.wikipedia.org/wiki/XAMPP. Zugriff: 17.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHyFh5A9.

[Wikd] Wikipedia.org. https://de.wikipedia.org/wiki/Hypertext_Markup_Language. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrfyK5o.

[Wike] Wikipedia.org. https://de.wikipedia.org/wiki/HTML5. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrhmADe.

[Wikf] Wikipedia.org. https://de.wikipedia.org/wiki/Webformular. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHri8fVO.

[Wikg] Wikipedia.org. https://de.wikipedia.org/wiki/JavaScript. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrxrkq9.

[Wikh] Wikipedia.org. https://de.wikipedia.org/wiki/PHP. Zugriff: 17.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHyFh5A9.

[wiki] wikipedia.org. https://de.wikipedia.org/wiki/Apache_HTTP_Server. Zugriff: 18.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rJp3X1B2.

[wikj] wikipedia.org. https://de.wikipedia.org/wiki/Webserver. Zugriff: 18.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rJt3l5hw.

[Wikk] Wikipedia.org. https://de.wikipedia.org/wiki/Texteditor. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrzluyW.

[Wikl] Wikipedia.org. https://de.wikipedia.org/wiki/Integrierte_Entwicklungsumgebung. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHrzLWhS.

[Wikm] Wikipedia.org. https://de.wikipedia.org/wiki/Medienbruch. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHs0BqSF.

[Wikn] Wikipedia.org. https://de.wikipedia.org/wiki/Eclipse_(IDE). Zugriff: 17.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHyFh5A9.

[Wiko] Wikipedia.org. https://de.wikipedia.org/wiki/Notepad%2B%2B. Zugriff: 17.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHyFh5A9.

[Wikp] Wikipedia.org. https://de.wikipedia.org/wiki/Texmaker. Zugriff: 17.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHyFh5A9.

[wikq] wikipedia.org. https://de.wikipedia.org/wiki/GitHub. Zugriff: 19.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rLOwbSll.

[Wikr] Wikipedia.org. https://de.wikipedia.org/wiki/Dropzone. Zugriff: 15.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHs0WyMC.

[Wiks] Wikipedia.org. https://de.wikipedia.org/wiki/Servlet. Zugriff: 17.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rHyFh5A9.

[Wikt] Wikipedia.org. https://de.wikipedia.org/wiki/SQL. Zugriff: 19.06.2017, Archiviert mit WebCite®: http://www.webcitation.org/6rMWig5xY.

Anhang

Erklärung

Hiermit versichere ich, Jakob Fleischer, dass ich meinen Anteil an unserer Dokumentation selbständig verfasst und keine anderen, als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: .....

(Unterschrift)

Erklärung

Hiermit versichere ich, Robin Schlaak, dass ich meinen Anteil an unserer Dokumentation selbständig verfasst und keine anderen, als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: .....

(Unterschrift)

Erklärung

Hiermit versichere ich, Lars Schmalbach, dass ich meinen Anteil an unserer Dokumentation selbständig verfasst und keine anderen, als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: .....

(Unterschrift)