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 denselbigen 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 verspätete Planung

Lars

Unsere Unerfahrenheit in der Projektarbeit hat uns zu einem überstürzten und unüberlegten Projekteinstieg getrieben, bei dem wir uns in Themen eingearbeitet haben, ohne uns einen genauen Plan zu erstellen. Zu spät haben wir gemerkt, dass unsere bisherige Arbeit weder zielführend, noch in irgendeiner Weise sinnvoll war, weshalb wir uns zusammengesetzt und intensiv beraten haben. Aus dieser Beratung entstand zum ersten Mal ein und realistischer Plan bezüglich des Aufbaus unseres Programms. Auch die Aufteilung der verschiedenen Arbeitsschritte zwischen den drei Gruppenmitgliedern fand hier spezifisch statt.

1.1.1 Aufteilung

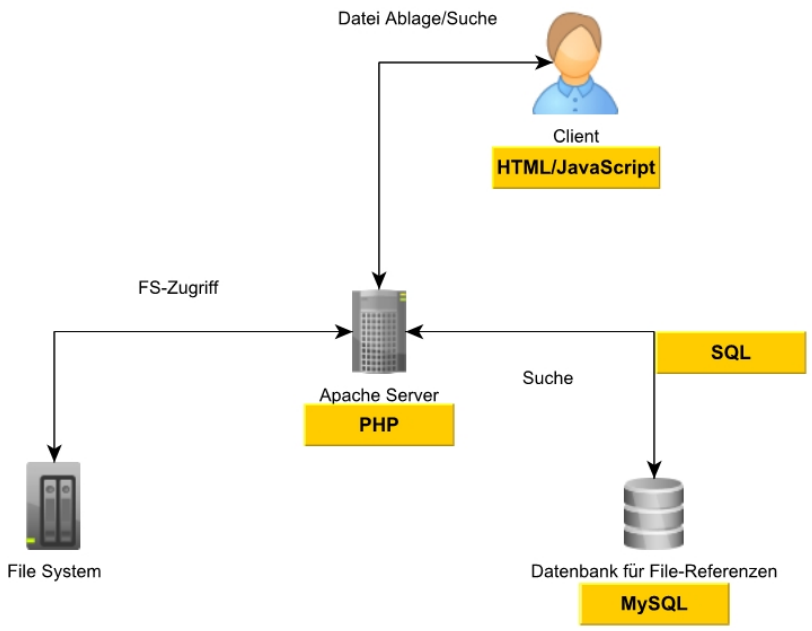

Abbildung 1: Erster realistischer Plan

In Abbildung 1 wird die Aufteilung der verschiedenen Aufgaben dargestellt. Der Client [1.1.2], welcher von Lars programmiert werden sollte, umfasst allgemein gesagt dass GUI, also die Benutzeroberfläche, und dessen Kommunikation mit dem Server. Dieser wiederum

soll von Robin eingerichtet und so gestaltet werden, dass er Anfragen kategorisiert und je nach Art der Anfrage das Datei-System beziehungsweise die Datenbank anspricht. Die soeben erwähnte Datenbank wird von Jakob erstellt und soll zunächst nur die Referenzen zu den Dateien im Datei-System abspeichern und verwalten.

1.1.2 Client / GUI

Die Tabelle 1 ist die digitalisierte Version des Tafelbilds, welches bei der oben beschriebenen, intensiven Beratung entstanden ist. Grün steht dabei dafür, dass diese Kriterien automatisch erkannt werden sollen, rot markierte sollen erfragt werden und das blau markierte Kriterium ist eine optionale Angabe. Hierbei ist zu beachten, dass bei der Suche alle Angaben optional sind, sodass man, wenn man eine Suche ohne Kriterien startet, alle vorhandenen Dateien als Ergebnis ausgegeben bekommt.

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 GUIs und dessen Funktionen

1.1.3 Server

Auch die Aufgaben des Servers wurden in unserem Tafelbild aufgeführt. Diese werden in Tabelle 2 dargestellt.

1.1.4 Datenbank

Die Datenbank, so nahmen wir es uns vor, soll Dateien aufnehmen und den Abfragen entsprechend 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.

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 des Servers

1.2 Zusammenarbeit und Kommunikation

Jakob

....

2 Kapitel: Grundlagen

Jakob & Lars

Im zweiten Kapitel dieser Dokumentation wird genauer auf die Themen eingegangen, in welche wir uns im Laufe des Projektkurses hineingearbeitet haben. Es wird nicht nur auf für das Projekt relevante Techniken, sondern auch auf solche, in die wir uns zwar eingearbeitet haben, sie jedoch nicht verwendet haben, eingegangen.

2.1 LaTeX

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.[**?**][**?**] *Robin*

...

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“, welcher den zweiten Teil darstellt, werden Kopfdaten, wie zum Beispiel der Titel der Seite oder andere, für den menschlichen Betrachter der Webseite zunächst nicht sichtbare, Informationen zur korrekten Darstellung des sichtbaren Teils der Webseite, angegeben.[**?**] Anzuzeigende Inhalte werden in den „body“, den dritten Teil, geschrieben. Hier werden also sämtliche Texte, Verweise, Grafiken und so weiter eingefügt, die auf der Webseite sichtbar sein sollen.[**?**]
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 mit 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 Anmeldeformularen, Suchfiltern und 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, durch sie ermöglichten, 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. [**?**] ...

2.5 Apache

Lars

...

2.6 Editoren und integrierte Entwicklungsumgebungen

Lars

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

Neben Editoren gibt es auch integrierte Entwicklungsumgebungen. Diese bestehen aus einer Sammlung an 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

Jakob

Durch die Arbeit mit GitHub, wird das gemeinsame, nicht zwingend parallele, Arbeiten möglich. GitHub verwaltet die Quellcodes, welche die Benutzer hochladen so, dass jeder, der über Zugriff auf das Projekt verfügt, dieses weiter führen kann. -Lars

2.8 Dropzone

Lars

Ursprünglich beschreibt der Begriff „Dropzone“ einen geheimen Speicherort für maschinell gestohlene Daten, wie zum Beispiel Passwörter und Kontodaten.[**?**]
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, welche sich leider erst nach und nach herauskristallisierte, 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:

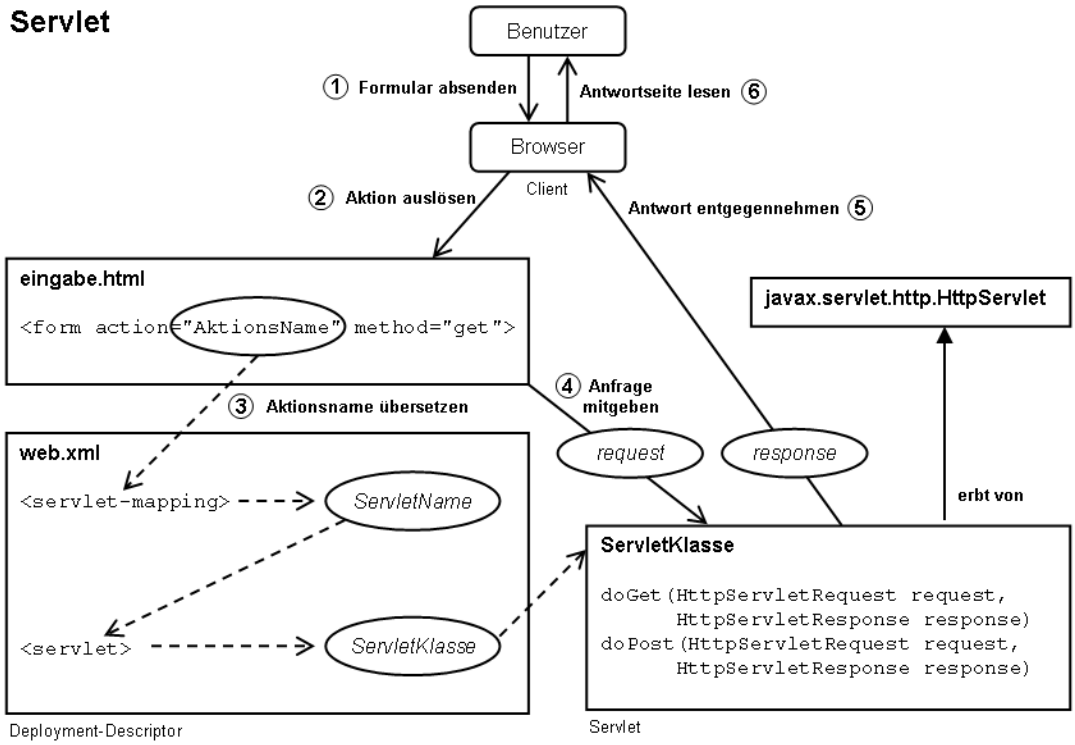

Abbildung 2: Java-Servlet[1]

Sowohl der Anfangs-, als auch der Endpunkt einer Datenverarbeitung mittels HTML-Formular [2.3.3] 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.

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

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,

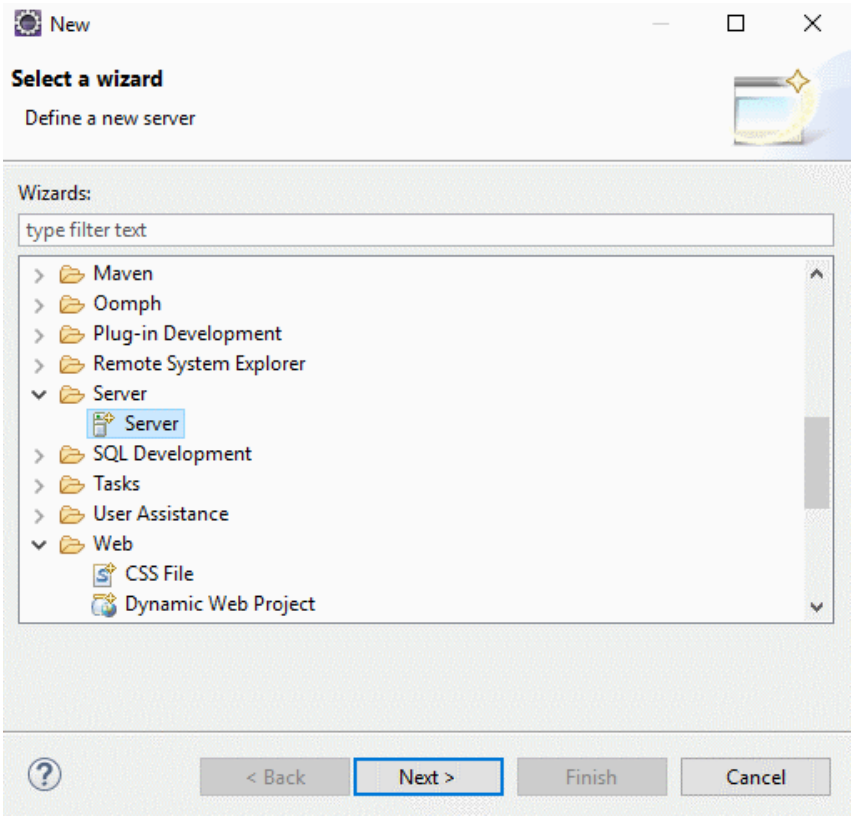

Abbildung 3: Server einrichten

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

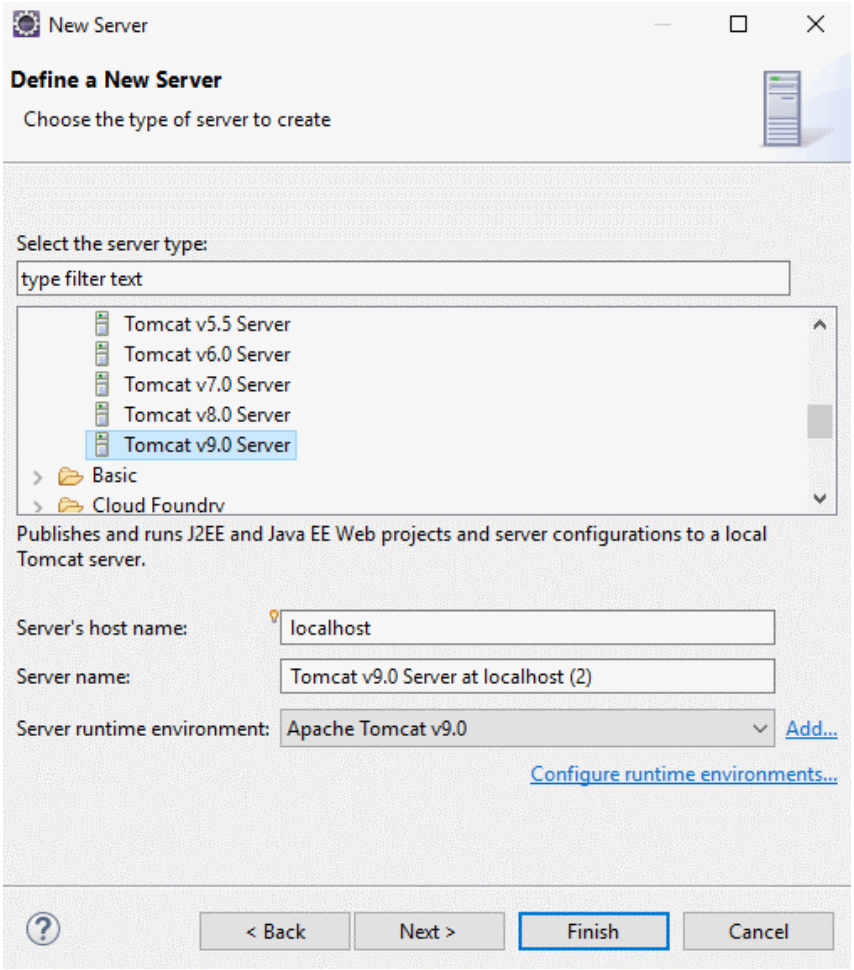

Abbildung 4: Server einrichten 2

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.

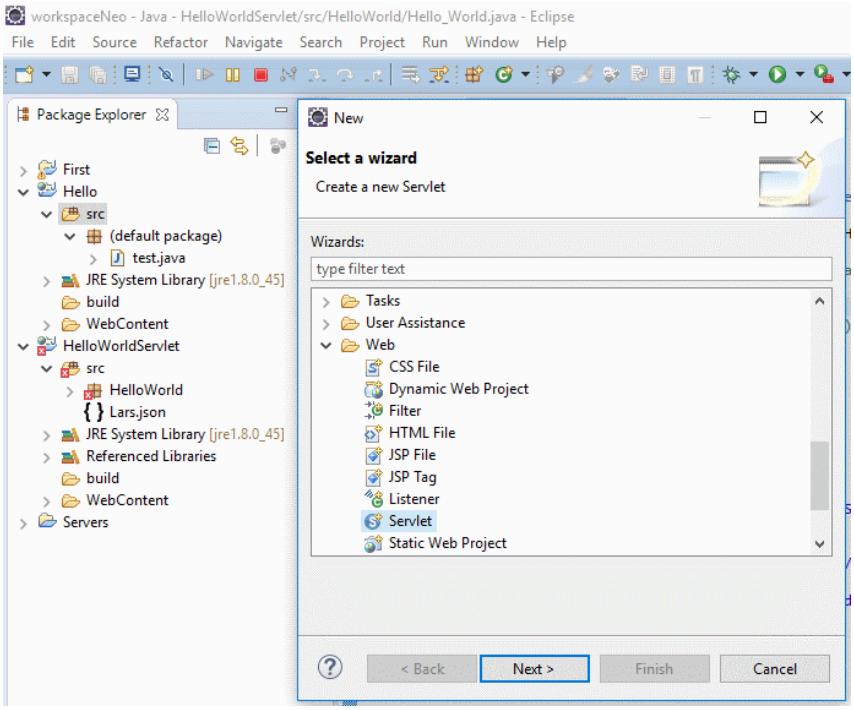

Abbildung 5: Servlet erstellen

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

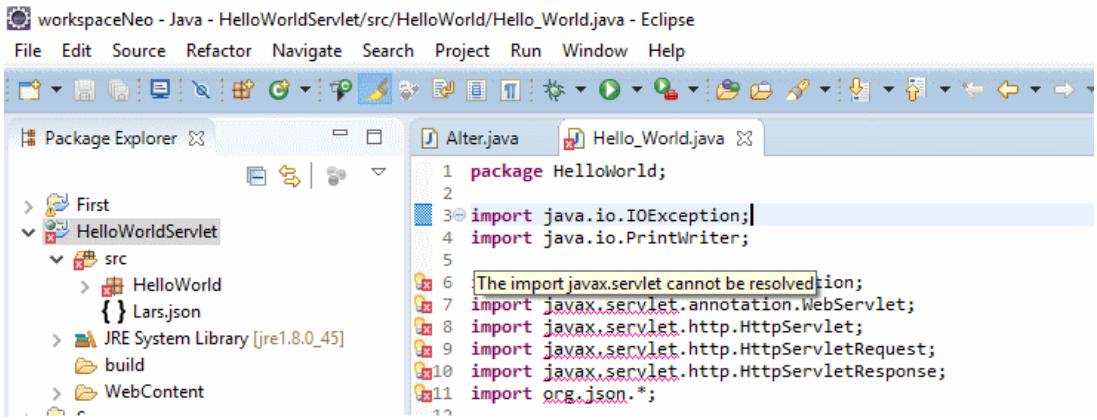

Abbildung 6: Fehlermeldung

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.

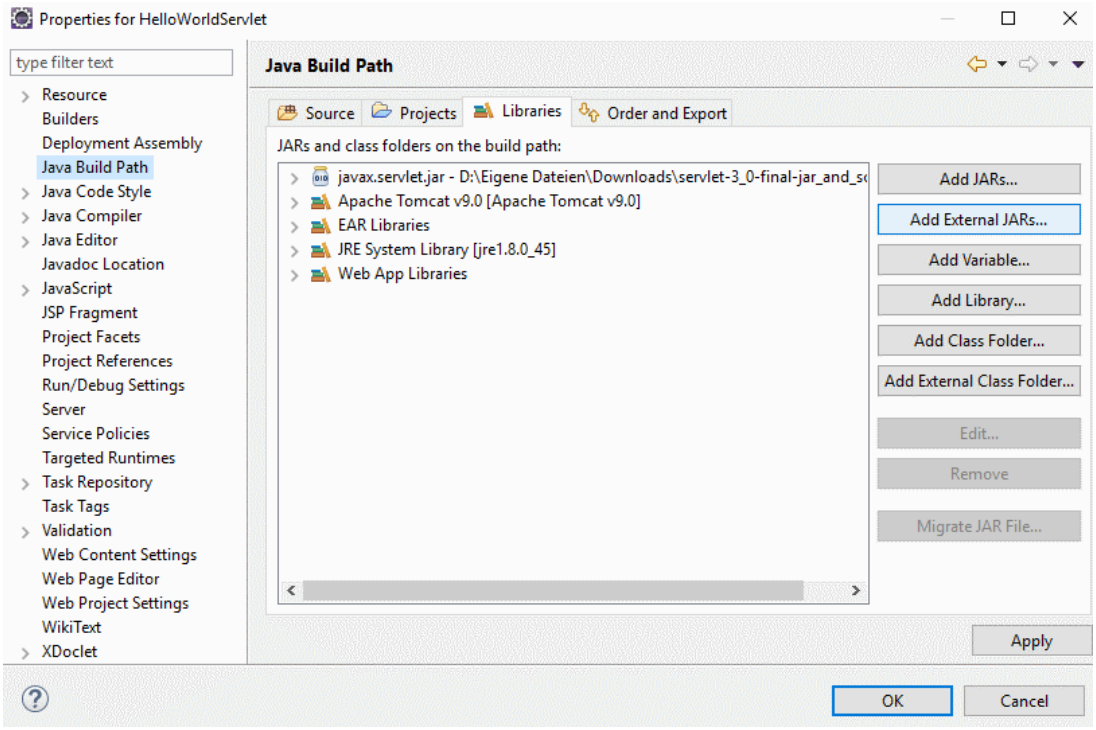

Abbildung 7: servlet-api.jar einbinden

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

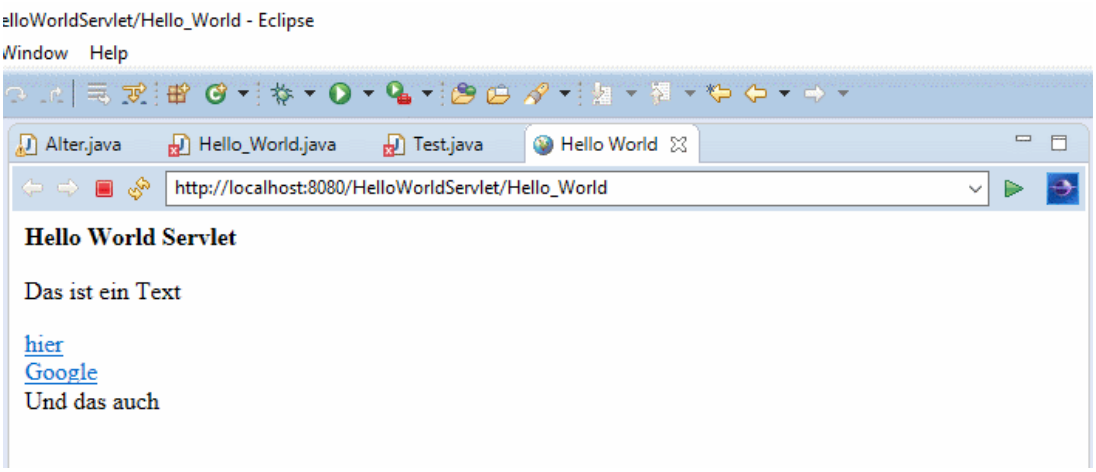

Abbildung 8: Servlet Beispiel

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

Ein weiteres Thema, in welches ich mich eingearbeitet habe war die Erstellung und das Auslesen von Dateien des Formates JSON mit Eclipse. ...

2.10.1 Erstellen

...

2.10.2 Auslesen

...

2.10.3 Vorteile

...

2.10.4 Nachteile

...

2.11 MYSQL

Jakob

Nachdem obige Thematiken angesichts unseres erneuerten Projektplans irrelevant geworden waren, habe ich mich mit MYSQL beschäftigt. ...

2.11.1 Vorteile

...

2.11.2 Datenbank-Entwurf

Im Folgenden möchte ich auf das Entwerfen einer MYSQL-Datenbank eingehen. ...

2.11.3 Verwaltung einer Datenbank mit PHP

Um eine Datenbank in unser Projekt zweckmäßig einzurichten, musste ich mich in eine MYSQL-Einbindung mittels PHP einlesen, welche ich nun darstellen möchte. ...

3 Projekt

Lars

In diesem Kapitel wird nun unser Projekt dargestellt, indem zunächst das Programm im Gesamten beschrieben wird, woraufhin die einzelnen Teile getrennt dargestellt werden.

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

Darstellung GUI...

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

Darstellung Interface und Datenbank...

4 Fazit

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

4.1

Lars

...

4.2

Robin

...

4.3

Jakob

...

Anhang

Erklärung

Hiermit versichere ich, dass ich meine Dokumentation selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: .....

(Unterschrift)