Ausarbeitung eines praktischen Lernkonzeptes zur Heranführung an die Programmierung mit sukzessive ansteigender Lernkurve

Projektarbeit Nr.:

vorgelegt am:

von:

Matrikelnummer:

Duale Hochschule:

Studienbereich:	Technik
Studiengang:	Praktische Informatik
Kurs:	
Ausbildungsstätte:	dotSource GmbH Goethestraße 1 07743 Jena

Betreuer Praxisbetrieb:

Gutachter Duale Hochschule:

Head Office Jena
Goethestraße 1
07743 Jena
FON +49 (0) 3641 797 9000
FAX +49 (0) 3641 797 9099
E-MAIL info@dotSource.de

Office Berlin
Pappelallee 78/79
10437 Berlin
FON +49 (0) 30 220 122 360

Office Leipzig
Hainstraße 1–3
04109 Leipzig
FON +49 (0) 341 9919 1000

Bankverbindung
Deutsche Bank Jena
IBAN DE63 8207 0000 0633 7778 00
BIC DEUTDE8EXXX

Commerzbank Jena
IBAN DE31 8204 0000 0259 9934 00
BIC COBADEFF821

Sparkasse Jena
IBAN DE35 8305 3030 0018 0037 61
BIC HELADEF1JEN

Geschäftsführer
Christian Otto Grötsch
Christian Malik
Frank Ertel

Gerichtsstand
Amtsgericht Jena
HRB 210634
USt-IdNr.: DE246243309
Steuer-Nr.: 162/107/03164

Sperrvermerk

Die vorliegende Arbeit beinhaltet interne und vertrauliche Informationen der Firma dotSource GmbH. Die Weitergabe des Inhaltes der Arbeit und eventuell beiliegender Zeichnungen und Daten im Gesamten oder in Teilen ist grundsätzlich untersagt. Es dürfen keinerlei Kopien oder Abschriften - auch in digitaler Form - gefertigt werden. Ausnahmen bedürfen der schriftlichen Genehmigung der Firma dotSource GmbH

I Inhaltsverzeichnis

II Abbildungsverzeichnis

III Abkürzungsverzeichnis

HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
HTTP	Hyper Text Transfer Protocol
REST	Representational State Transfer

IV Anlagenverzeichnis

1 Einleitung

„Der Mangel an IT-Fachkräften hat einen neuen Höchststand erreicht. In Deutschland gibt es derzeit 82.000 offene Stellen für IT-Spezialisten. Das entspricht einem deutlichen Anstieg um 49 Prozent im Vergleich zum Vorjahr. 2017 waren 55.000 Stellen vakant.“[1] Um dem Fachkräftemangel entgegen zu wirken muss dafür gesorgt werden, dass möglichst viele Menschen Begeisterung an der Informatik finden.

Die dotSource GmbH bietet beispielsweise Praktika für Schüler an. Den Praktikanten wird dabei die Möglichkeit gegeben die Programmierung und das Berufsleben eines Programmierers näher kennenzulernen. Es werden zentrale Lernaufgaben bereitgestellt, mit denen die Schüler die Grundlagen der Programmierung näher betrachten können. Eine Beschäftigung in Entwicklerteams würde einen zu hohen Aufwand bedeuten, außerdem können die Schüler hier nur bedingt produktiv eingesetzt werden. Das derzeitige Aufgabenkonzept für Schüler umfasst mehrere Themenbereiche und ist so ausgelegt, dass nur eine geringe Betreuung notwendig ist. Um Schüler effizient an die Informatik und Programmierung heranzuführen wird aber eine bessere Betreuung benötigt. Es soll daher ein Aufgabenkonzept erstellt werden, mit welchem Mitarbeiter einen sehr geringen Einarbeitungsaufwand haben. Gleichzeitig sollen diese aber auch für die gesamte Zeit eines Praktikums für die Betreuung der Schüler zuständig sein.

In dieser Arbeit werden die Planung, Entwicklung und Umsetzung eines Lernkonzeptes für die Programmierung betrachtet. Dazu wird die Konzipierung von Aufgaben, sowie die Entwicklung einer dazugehörigen Applikation und deren Einsatz in einem Praktikum beschrieben. Zum Schluss wird auf Basis der Praktikumsauswertung ein Fazit gezogen, ob sich das Aufgabenkonzept für die dotSource nutzen lässt.

[1] [Bit18]

2 Ausgangslage

Die bisherigen Aufgaben für Praktika sind nicht einheitlich organisiert. Es gibt kein genaues Schema nach welchem die Praktika ablaufen. Häufig wird erst vor Beginn des Praktikums ein genauerer Plan für den Ablauf erstellt. Trotz verschiedener Aufgaben sind diese nicht für jeden Kenntnisstand ausgelegt. Es handelt sich um teilweise zu schwere und nicht sehr interessante Aufgaben für Praktikanten. Des Weiteren ist die Betreuung der Schüler auf ein Minimum reduziert. Dadurch sind die Kenntnisvermittlung und die Überprüfung des Kenntnisstandes so gut wie nicht gegeben.

In der dotSource GmbH sind derzeit sowohl Aufgaben für einzelne Tage als auch für Praktikumswochen vorhanden. Die Aufgaben für die einzelnen Tage sind dafür da einen Einblick in Hyper Text Markup Language (HTML), Cascading Style Sheets (CSS) und JavaScript zu erhalten. Die einzelnen Sprachen werden an einer Beispielseite für Urlaubsbuchungen erklärt und betrachtet.

Es handelt sich bei der Beispielseite um eine rudimentäre HTML Datei, welche für die Gestaltung das Bootstrap Framework und für die Funktionalitäten JavaScript nutzt. Die Aufgaben, mit denen an die Programmierung herangeführt werden soll, sind in drei Bereiche unterteilt. Die einzelnen Bereiche unterscheiden sich jeweils in der Schwierigkeit voneinander. Das Ziel dieser Aufgaben ist es mit steigender Anforderung für jeden Schüler die passenden Aufgaben anzubieten. Leistungsstarke bzw. erfahrenere Schüler sollen die einfachen Aufgaben möglichst schnell bewältigen und mit den anspruchsvolleren Aufgaben beschäftigt werden. Im Gegensatz dazu bekommen die anderen Schüler genügend leichte Aufgaben, um ebenfalls Erfolgserlebnisse davonzutragen.

Die Aufgaben für mehrwöchige Praktika bestehen aus verschiedenen Aufgabenbereichen. Ein Aufgabenbereich ist auf die Einführung in HTML und CSS ausgerichtet. Hier wird ein Tutorial und mehrere detaillierte Anleitungen bereitgestellt,

wodurch die Schüler selbstständig die Bearbeitung der dazugehörigen Aufgaben übernehmen können. Ziel der Aufgaben ist es eine statische Seite nachzubauen und so HTML und CSS kennenzulernen. Der nächste Aufgabenbereich dient zur Heranführung an JavaScript. Die Schüler sollen hier selbstständig einen HTML Taschenrechner um seine Funktionalitäten erweitern und so die grundlegenden Elemente der Programmierung kennenlernen. Ein weiterer Aufgabenbereich befasst sich mit der visuellen Programmierung eines Lego Roboters. Hier müssen mittels visueller Programmieroberfläche verschiedene Aktionen für den Roboter entworfen werden. Der letzte Aufgabenbereich befasst sich mit der Java Programmierung.

3 Aufgabenkonzeption

Grundlegende Probleme bei der Aufgabenkonzeption sind zum einen, dass der Kenntnisstand der Praktikanten im Vorfeld nicht eindeutig klar ist und zum anderen ist ein stark differenzierter Kenntnisstand der Schüler zu erwarten. Es muss daher darauf geachtet werden, dass für jeden Kenntnisstand die passenden Aufgaben vorhanden sind.

Das neue Aufgabenkonzept wird ähnlich dem derzeitigen sein. Auch in dem neuen Konzept sollen die Grundlagen der Programmierung und der Aufbau von Webseiten anhand einer Beispielseite erklärt werden. Um die unterschiedlichen Kenntnisstände abzudecken werden die Aufgaben wieder in verschiedene Bereiche eingeteilt, welche sich durch ihren Anspruch voneinander unterscheiden.

Zunächst wird die grundlegende Struktur der Applikation konzipiert. Es soll eine Webanwendung werden, welche einen rudimentären Shop abbildet. Der Shop soll nur beispielhaft die Funktionalitäten, wie das anzeigen von Produkten und das hinzufügen zu einem Warenkorb, realisieren. Auf die Seite soll mittels eines Webbrowsers lokal zugegriffen werden können.

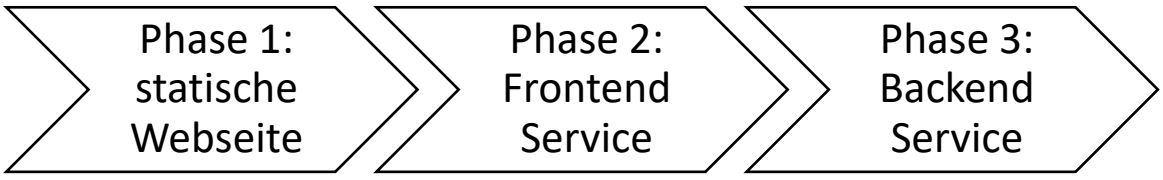

Abbildung 1: Phasenmodell

Um eine ansteigende Lernkurve zu gewährleisten werden die Aufgaben und die Applikation in drei Phasen eingeteilt (siehe Abbildung 1). In diesen Phasen ändern sich sowohl die Aufgabenschwierigkeit als auch die Komplexität der Applikation. Die erste Phase wird aus einer ähnlichen Aufgabe bestehen, wie die bereits existierende Aufgabe für einzelne Praktikumstage. Es wird wieder eine statische HTML Seite mit

CSS und JavaScript Komponenten bereitgestellt. Die statische Seite wird verschiedene Produkte anzeigen und es möglich machen, diese zu einem Warenkorb hinzuzufügen. Der Warenkorb zeigt die in ihm befindlichen Produkte an und gibt dem Nutzer die Möglichkeit Produkte auch wieder zu entfernen. Außerdem soll der Warenkorb die Steuern und Gesamtpreise anzeigen. Da es sich um eine einfache statische HTML Seite handelt, werden Änderungen nicht gespeichert und ein aktualisieren des Browsers setzt die Seite wieder in ihren Ausgangspunkt zurück. An der Seite müssen Änderungen in Struktur, Gestaltung und Funktionalität vorgenommen werden. Des Weiteren müssen Fehler im Code erkannt und behoben und Neuerungen hinzugefügt werden.

In der nächsten Phase wird an einem Frontend Service gearbeitet. Der Frontend Service verarbeitet vom Webbrowser entgegenkommende Anfragen, erzeugt eine HTML Ausgabe und gibt sie an den Browser zurück. Auch hier werden wieder dieselben Funktionalitäten ermöglicht, wie in der ersten Phase. Der wesentliche Unterschied besteht darin, dass Änderungen am Warenkorb dieses Mal solange erhalten bleiben sollen, wie der Browser des Nutzers geöffnet ist. Zusätzlich werden der Warenkorb und die Übersichtsseite der Produkte auf verschiedene Dateien ausgelagert und über verschiedene Adressen erreichbar gemacht. Hinzu kommt noch eine Produktdetailseite, welche Informationen zu einem einzelnen Produkt anzeigen soll. Die Aufgaben in dieser Phase sind zunächst ebenfalls auf das Suchen und Auffinden von Fehlern ausgerichtet. Im weiteren Verlauf sollen auch hier Funktionalitäten geändert und hinzugefügt werden.

In der dritten Phase wird an einem Backend Service gearbeitet. Die Applikation ist strukturell ähnlich der des Frontend Services aufgebaut. Der Unterschied besteht darin, dass Daten, wie der Inhalt des Warenkorbs oder die Produkte an sich in einer Datenbank gespeichert werden. Um auf die Daten zuzugreifen werden zusätzliche Strukturen eingebaut, welche Anfragen entgegennehmen und Daten zurückliefern, bzw. die Datenbank manipulieren. Ziel ist es das Frontend vom Backend

weitestgehend zu trennen und jeweils einem Service zuzuordnen. Ein Service ist für die Ausgabe und ein weiterer für Hintergrundfunktionalitäten zuständig. In dieser Phase orientieren sich die Aufgaben an dem Hinzufügen von neuen Funktionalitäten.

Für die Phase zwei und drei wird als Programmiersprache Java festgelegt. Um das Arbeiten mit der Applikation zu erleichtern wird Spring Boot als Grundlage genutzt. Dadurch wird ein hoher Programmieraufwand erspart, denn Spring Boot liefert sowohl die Basisapplikation als auch den Applikationsserver mit sich. Somit ist es möglich die Entwicklung der eigentlichen Funktionalität der Webseite zu übernehmen, ohne sich dabei Gedanken über Konfigurationen oder das Hosting der Anwendung machen zu müssen. Für die dritte Phase wird als Datenbank MongoDB verwendet. Auch dadurch werden Aufwände, wie z.B. das organisieren und planen einer Datenbank erspart.

4 Verwendete Technologien

4.1 Bootstrap

Bootstrap ist ein Framework, welches die Entwicklung mit HTML, CSS und JavaScript erleichtert. Mit Bootstrap können ohne größere Aufwände Templates erstellt werden. Mit vorgefertigten CSS Klassen können HTML Elemente gestaltet werden und des Weiteren werden diverse JavaScript Funktionalitäten bereitgestellt, um die Oberfläche dynamisch zu gestalten. Bootstrap ist darauf ausgerichtet, sowohl „responsive“ als auch „mobile-first“ Templates zu erstellen.[2]

4.2 Gradle

Gradle ist ein auf Java basierendes Build-Management-Automatisierungs-Tool. Es nutzt eine auf Groovy (Programmiersprache) basierende domänenspezifische Sprache zur Beschreibung der zu erstellenden Projekte.[3] Der Software Build Prozess umfasst das Kompilieren von Quellcode, das Verpacken der kompilierten Dateien in kompensierte Formate (z.B. JAR oder ZIP), das Erstellen von Installern und das Erstellen oder Aktualisieren von Datenbankschemen. Eine Automatisierung liegt dabei vor, wenn diese Aktionen wiederholbar sind, kein Eingreifen eines Entwicklers erforderlich ist und keine weiteren Informationen benötigt werden, außer die bereits hinterlegten.[4]

4.3 Spring / Spring Boot

„Spring is described as a lightweight framework for building Java applications“[5]. Mit Spring können verschiedene Java Applikationen erstellt werden. Darunter sind alleinstehende Applikationen, Web oder auch Java Platform Enterprise Edition

[2] Vgl. [Boo]
[3] Vgl. [Gra18]
[4] Vgl. [Agi]
[5] [Cos17], S. 1

Anwendungen. Mit Spring ist es möglich durch nur wenige Änderungen, bereits die Funktionalitäten des Spring Core zu nutzen.[6]

Das Spring Boot Framework dient dazu Spring Applikationen schnell zu erstellen und ausführbar zu machen. Um Spring Boot zu nutzen müssen nur wenige Konfigurationen oder Code Änderungen vorgenommen werden. Der Fokus liegt darauf die Funktionalität zu entwickeln ohne sich dabei größere Gedanken um die Spring Konfiguration zu machen.[7]

4.4 Thymeleaf

Thymeleaf ist eine Java Template Engine, welche sowohl für Webanwendungen als auch für alleinstehende Applikationen genutzt werden kann.[8] Thymeleaf wird verwendet, um das Backend (Java) vom Frontend (HTML) zu trennen. Mittels Thymeleaf ist es möglich Informationen an ein Template zu übergeben. Die übergebenen Informationen werden vor der Ausgabe der Datei in HTML Strukturen umgewandelt.

4.5 MongoDB

MongoDB ist eine Dokumenten Datenbank, welche frei und open-source ist. Es ist die am weitesten verbreitetste NoSQL Datenbank. Bei einer NoSQL Datenbank werden die Daten in JavaScript Object Notation artigen Dokumenten abgespeichert. Die abgespeicherten Objekte referenzieren dabei auf die im Programmcode befindlichen Objekte. MongoDB zeichnet sich durch eine hohe Verfügbarkeit und Skalierbarkeit aus.[9]

---

[6] Vgl. [Cos17]
[7] Vgl. [Wal15]
[8] Vgl. [Thy]
[9] Vgl. [Mon]

5 Entwicklung des Prototyps

5.1 Statische Webseite

Die statische Seite besteht im Wesentlichen aus zwei Dateien. Eine HTML Datei, für die Struktur sowie Gestaltung und eine JavaScript Datei für die Funktionalität. Für die Gestaltung der Seite wird das Bootstrap Framework genutzt.

```
<div class="card">
    <div class="card-header bg-primary text-white">Produktname</div>
    <img class="card-image-top" src="media/bild.jpg" alt="Bild"/>
    <div class="card-body"></div>
</div>
```

Abbildung 2: Produkt Element

Zunächst werden die Produkte eingefügt. Jedes Produkt erhält einen Namen, eine Beschreibung, ein Bild und einen Preis. Sie werden statisch in die HTML Datei eingefügt (siehe Abbildung 1). Die Produkte werden kartenartig angezeigt und erhalten zudem einen Button. Mittels dieses Buttons werden die Produkte zum Warenkorb hinzugefügt.

```
<div class="modal fade" id="warenkorb" role="dialog">
    <div class="modal-dialog">
        <div class="modal-content">
            <!-- Modal Kopfzeile -->
            <div class="modal-header"></div>
            <!-- Modal Inhalt -->
            <div id="warenkorbInhalt" class="modal-body"></div>
            <!-- Modal Fußzeile -->
            <div class="modal-footer"></div>
        </div>
    </div>
</div>
```

Abbildung 3: Warenkorb Modal

Der Warenkorb wird als sogenanntes Modal erstellt. Ein Modal ist eine Pop-up Anzeige, welche erst durch eine bestimmte Interaktion mit der Seite angezeigt wird.

Für eine solche Anzeige stellt Bootstrap bereits alle nötigen CSS Klassen und JavaScript Funktionen bereit. Es wird ein HTML Container erstellt, welcher eine Kopfzeile, einen Inhaltsbereich und eine Fußzeile besitzt (siehe Abbildung 2). Der Container benötigt nur eine eindeutige ID und die CSS Klassen, welche für die Darstellung eines Modals zuständig sind. Die nötigen Struktur- und Gestaltungselemente sind damit erstellt. Die Funktionalität des Modals wird durch entsprechende Attribute an den Produktbuttons für den Warenkorb erreicht. Klickt der Nutzer nun auf einen der Buttons wird das Modal für den Warenkorb angezeigt.

```
function zeigeWarenkorbInhalt() {
    var text;
    var warenkorbInhalt = document.getElementById('warenkorbInhalt');
    warenkorbInhalt.innerHTML = "";
    warenkorb.forEach(function(produkt) {
        text = "<p>"
            + produkt.name
            + ": "
            + document.getElementById(produkt.name).dataset.preis
            + " €</p>";
        warenkorbInhalt.innerHTML+=text;
    });
    var gesamt = berechneGesamt();
    document.getElementById('gesamt').innerHTML=gesamt;
    document.getElementById('steuer').innerHTML=berechneSteuer(gesamt);
}
```

Abbildung 4: Funktion für die Anzeige des Warenkorbs

Um es zu ermöglichen, Produkte zum Warenkorb hinzuzufügen, wird in der JavaScript Datei die Datenhaltung realisiert. Dazu wird ein Array genutzt, welches Produktobjekte speichert. Die Produktobjekte werden statisch erstellt und verweisen jeweils auf das in der HTML Datei zugehörige Produkt. Die Produktobjekte speichern die ID und den Namen des Produktes. Im Anschluss wird eine Funktion definiert, welche es ermöglicht den Inhalt des Warenkorb Modals zu manipulieren (siehe Abbildung 3). Diese Funktion fügt die Produkte, welche sich im Warenkorb Array befinden, in das Warenkorb Modal ein. Dazu wird einfach ein definierter Text in das HTML Element eingefügt. Informationen, wie beispielsweise der Preis eines Produktes, werden aus dem HTML

genommen. Für die Preisberechnungen werden zwei weitere Funktionen definiert, welche jeweils die Steuer und den Gesamtpreis berechnen. Um den Warenkorb manipulieren zu können werden weitere Funktion zum Hinzufügen und Entfernen von Produkten erstellt. Diese fügen das angegebene Produkt zum Warenkorb Array hinzu oder entfernen es wieder. Um diese Funktionen nutzen zu können werden Event Handler Attribute auf den Buttons für das Hinzufügen zum Warenkorb platziert. Diese Event Handler führen zunächst die Funktion zum Hinzufügen eines Produktes aus und im Anschluss die Funktion zum Anzeigen des Warenkorbs. Der Nutzer bekommt somit die aktuell im Warenkorb befindlichen Produkte angezeigt. Um weitere Grundlagen für Aufgaben bereitzustellen, werden HTML Elemente hinzugefügt, wie z.B. ein Header, eine Navigationsleiste, ein Slider in dem Bilder durchlaufen und eine Suchfunktion.

```
function suche() {
    const value = $("#produktSuche").val().toLowerCase();
    $(".card-header").filter(function () {
        $(this).parents(".col-md-6")
            .toggle($(this).text().toLowerCase().indexOf(value) > -1);
    });
}
```

Abbildung 5: Suchfunktion

Für die Suchfunktion wird ein Formular erstellt, welches beim Klick auf einen Button eine JavaScript Funktion aufruft. Diese Funktion sucht in den HTML Elementen für die Produkte nach einem Namen, welcher die eingegebene Zeichenkette beinhaltet (siehe Abbildung 4). Groß- und Kleinschreibung wird dabei ignoriert. Um ein paar einfachere Aufgaben zu ermöglichen werden die Pfade zu Bildern leicht abgeändert, sodass diese nicht mehr angezeigt werden.

5.2 Frontend Service

Für diese Anwendung werden das *starter-web* und das *starter-thymeleaf* Paket über Gradle eingebunden. Nun können in der Applikation Thymeleaf Templates und diverse Web-Funktionalitäten verwendet werden.

```java
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {

    @GetMapping("")
    public String index() {
        return "home";
    }
}
```

Abbildung 6: HomeController

Als erstes wird eine Startseite für den Webshop erstellt. Dazu wird ein sogenannter Controller benötigt. Bei einem Controller handelt es sich um eine Java Klasse, welche Hyper Text Transfer Protocol (HTTP) Anfragen verarbeitet. Spring Boot stellt diverse Annotationsfunktionen bereit und mit der Annotation *@controller* kann eine Klasse als Controller gekennzeichnet werden (siehe Abbildung 3). Innerhalb dieser Klasse können nun verschiedene Mapping Annotationen genutzt werden. Eine Mapping Annotation sorgt dafür, dass eine URL einer bestimmten Funktion zugeordnet wird. Erhält die Applikation eine HTTP Anfrage an die definierte URL, dann führt die Applikation die Funktion mit der entsprechenden Annotation aus. Es wird eine Funktion erstellt, welche einen String zurückliefert und die *@GetMapping* Annotation besitzt. Außerdem wird eine rudimentäre HTML Datei erstellt, welche etwas Inhalt aufweist und den Namen des in der Funktion zurückgegebenen Strings besitzt. Wird die Anwendung gestartet, ist die Webseite über die Adresse *http://localhost:8080* erreichbar und zeigt den Inhalt der zuvor erstellten HTML Datei gerendert an. Durch den gerade erstellten Controller und die Funktion wird von nun an die Route *localhost:8080/* auf die Funktion gemappt. Damit sind die Grundlagen für die weiteren Funktionalitäten der Seite geschaffen.

Es wird wieder dafür gesorgt, dass verschiedene Produkte angezeigt werden. Dazu wird die Klasse *product* erstellt. Diese Klasse wird dazu genutzt verschiedene

Informationen zu Produkten zu speichern, wie z.B. ID, Name, Beschreibung, Foto und Preis.

```
private List<Product> productList;

public ProductModel() {
    productList = new ArrayList<>();
    productList.add(new Product( id: "1",  name: "Ananas",  description: "text",  photo: "bild.jpg",  price: 1));
    productList.add(new Product( id: "2",  name: "Birne",  description: "text",  photo: "bild.jpg",  price: 2));
    productList.add(new Product( id: "3",  name: "Apfel",  description: "text",  photo: "bild.jpg",  price: 3));
}

public List<Product> findAll() { return productList; }

public Product find(String id) {
    for (Product product : productList) {
        if (product.getId().equals(id)) {
            return product;
        }
    }
    return null;
}
```

Abbildung 7: ProductModel

Um spezifische Produkte abrufbar zu speichern wird die Klasse *ProductModel* erstellt (siehe Abbildung 6). Diese wird genutzt, um einige Produkte statisch bereitzustellen. Beim Erzeugen eines Objektes der Klasse werden verschiedene Instanzen von Produkten erzeugt und in einer Liste gespeichert. Diese Liste ist immer gleichbleibend und dient als Datenbank für die Produkte. Um die Produkte nun abzurufen, wird ein neuer Controller mit dem Namen *ProductController* erstellt. Dieser beinhaltet zwei Funktionen, welche beide für die Produktdarstellung genutzt werden. Die erste Funktion dient dazu da alle Produkte anzuzeigen und die zweite dient zur Anzeige eines einzelnen Produkts. Der Controller fragt dazu die Produktdaten vom *ProductModel* ab. Die Produktdaten werden von dem Controller an das Template weitergegeben. Das Template wird durch Thymeleaf erzeugt und sorgt dafür, dass übergebene Informationen in HTML umgewandelt werden. Für die Anzeige aller und einzelner Produkte werden zwei separate Templates erzeugt. Im Template für die Anzeige aller Produkte wird eine Schleife gebaut, welche für jedes in der Liste befindliche Element neue HTML Elemente erstellt. Diese Elemente erhalten dann die Informationen über die Produkte und Buttons, um das Produkt zu einem Warenkorb

hinzuzufügen. Es wird dadurch eine ähnliche Ausgabe, wie in der ersten Phase, erzeugt. Das zweite Template ist eine Produkt-Detailseite. Der Unterschied zum anderen Template ist, dass keine Schleife darin vorkommt und dass statt einer Liste ein einzelnes Produkt an das Template übergeben wird.

```
@GetMapping("id/{id}")
public ModelAndView showProduct(@PathVariable("id") String id) {
    Product product = new ProductModel().find(id);
    if (product != null) {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject( attributeName: "product", product);
        modelAndView.setViewName("productDetail");
        return modelAndView;
    }
    return new ModelAndView( viewName: "redirect:/products/all");
}
```

Abbildung 8: Funktion für die Detailseite

Um auf die Produktdetailseite zu gelangen werden Verlinkungen auf der Übersichtsseite in das Template eingebaut. Die Produkte sind über die Adresse http://localhost:8080/products/id/ erreichbar. Am Ende dieser Adresse wird die ID des Produktes, welches angezeigt werden soll, angehängt. Um dies zu realisieren werden Pfadvariablen genutzt (siehe Abbildung 7). Der Nutzer kann sich somit alle oder einzelne Produkte anzeigen lassen.

Im Anschluss wird dafür gesorgt, dass diese Produkte in einen Warenkorb getan werden können. Zunächst wird eine Klasse *Cart* erstellt, welche eine Liste von Items speichert. Dafür wird zusätzlich noch die Klasse *Item* erstellt. Diese speichert eine Stückzahl zu einem speziellen Product. Für den Cart werden noch Funktionen erstellt, welche die Manipulation des Warenkorbs ermöglichen. Zum einen wird eine Funktion für das Hinzufügen eines Produktes zum Warenkorb erstellt. Diese fügt ein neues Item mit dem Produkt und der Stückzahl eins zum Warenkorb hinzu. Zum anderen werden Funktionen implementiert, welche zum entfernen von Produkten genutzt werden.

Hierzu wird eine Funktion erstellt, welche ein bestimmtes Produkt mit einer speziellen ID aus dem Warenkorb entfernt. Die letzte Funktion leert die Item Liste des Warenkorbs. Um den Index eines Produktes im Warenkorb zu erhalten, wird eine Hilfsfunktion erstellt.

```
private static Map<String, Cart> carts = new HashMap<>();

private CartModel() {}

public static Cart getCartForSessionId(String sessionId) {
    if(carts.containsKey(sessionId)) {
        return carts.get(sessionId);
    }
    Cart cart = new Cart();
    carts.put(sessionId, cart);
    return cart;
}
```

Abbildung 9: CartModel

Damit der Warenkorb für einen Nutzer gespeichert werden kann, wird die Klasse *CartModel* erstellt (siehe Abbildung 8). Diese speichert ein Warenkorb-Objekt und die dazugehörige Nutzer Session ID. Die Nutzer Session ID ist eine Identifikationsmöglichkeit, welche der Nutzer für seine Browsersitzung erhält. Diese ID erhält der Nutzer beim Aufruf der Seite und behält sie solange bis der Webbrowser geschlossen wird. Dadurch ist es möglich, dass der Warenkorb gespeichert und dem Nutzer zugeordnet werden kann.

```
@RequestMapping(value = "removeAll", method = {RequestMethod.POST})
public ModelAndView removeAllProductsFromCart(HttpSession session) {
    Cart cart = CartModel.getCartForSessionId(session.getId());
    cart.removeAllItems();
    return createCartView(cart);
}
```

Abbildung 10: CartController

Um den Warenkorb anzuzeigen und zu manipulieren wird der CartController erstellt (siehe Abbildung 9). Dieser Controller erhält eine Funktion, um den Inhalt des

Warenkorbs anzuzeigen und weitere Funktionen, um den Warenkorb über den Aufruf verschiedener Adressen zu manipulieren. Das Manipulieren soll über die HTTP POST Methode erfolgen. Als erstes wird ein Template für den Warenkorb angelegt, welche den Inhalt anzeigt und Buttons, welche es ermöglichen die Produkte zu entfernen. Die Buttons besitzen dabei einen Link, auf die Adressen, welche für das Manipulieren des Warenkorbs zuständig sind und übergeben über die URL gleichzeitig jeweils noch die Product ID. Die Funktionen zum manipulieren des Warenkorbs greifen über die Session ID auf den Warenkorb des Nutzers zu und nehmen Änderungen an diesem vor. Der neue Warenkorb wird dann jeweils an das Warenkorb Template übergeben und dieses zeigt den neuen Stand des Warenkorbs an. Um Produkte zum Warenkorb hinzuzufügen, müssen noch die Links der Buttons im Produkt Template angepasst werden. Diese verweisen jeweils auf die http://localhost:8080/cart/add Adresse und übergeben per POST Methode die Produkt ID.

Die Applikation erfüllt nun alle Funktionalitäten, welche auch in der ersten Phase bereitgestellt werden. Auch in diesem Service werden wieder die Links zu Bildern oder Routen manipuliert, um die Grundlage für einfache Aufgaben zu schaffen.

5.3 Backend Service

Der Backend Service muss dieselbe Funktionalität bieten wie der Frontend Service. Allerdings sollen die Datenbestände in einer Datenbank gespeichert werden und von dort aus abgefragt werden. Diese Abfrage wird über eine Representational State Transfer (REST) Schnittstelle erfolgen. Um Aufwände zu sparen wird der Backend Service nicht komplett neu gebaut, sondern der Frontend Service wird als Vorlage genutzt. Es werden zunächst Änderungen an den Model Klassen vorgenommen. Diese Klassen werden so umgebaut, dass die Produkte und Warenkörbe in einer Datenbank liegen. Um MongoDB in Kombination mit Spring Boot verwenden zu können wird über Gradle das starter-data-mongodb Paket hinzugefügt.

```
public interface CartRepository extends MongoRepository<Cart, String> {

    Cart findCartBySessionId(String sessionId);

    List<Cart> findAll();

}
```

Abbildung 11: CartRepository

Um mit der Datenbank interagieren zu können werden zwei Repository Klassen erstellt. Mit der *Cart*- und *ProductRepository* Klasse können Datenbestände abgefragt und manipuliert werden. Die beiden Repository Klassen sind Interfaces und erweitern jeweils die *MongoRepository* Klasse. Dadurch sind über die Interfaces die Funktionen, welche mit der Datenbank zu kommunizieren verfügbar. Im *CartRepository* werden Funktionen zum finden von allen und einzelner Warenkörbe eingefügt. Mit diesen beiden Funktionen ist es möglich, *Cart* Objekte aus der Datenbank abzufragen. Auch im *ProductRepository* werden Funktionen zum Finden einzelner und aller Produkte eingefügt. Nun ist es möglich sowohl Produkte als auch Warenkörbe aus der Datenbank abzufragen. Um nun aber auch Datenbestände in die Datenbank einzufügen, werden nun REST Controller erstellt. Zunächst wird die *CartRestController* Klasse erstellt. Dieser Controller ist dafür zuständig sowohl Produkte zurückzuliefern,

als auch Produkte in die Datenbank einzufügen und zu löschen. Dafür werden vier Funktionen implementiert, von denen zwei ähnlich wie beim *ProductController* für die Rückgabe aller und einzelner Produkte zuständig sind. Die beiden Funktionen rufen die zuvor definierten Funktionen der Repository Klasse auf. Die anderen Funktionen nutzen ebenfalls die Repository Klasse, greifen aber auf die Funktionen zu, welche durch die *MongoRepository* Klasse zur Verfügung stehen. Es ist wenig aufwändig die Datenbestände zu manipulieren, da der mit Spring Boot bereitgestellte Objekt Relation Mapper (ORM) viel Aufwand abnimmt. Der ORM sorgt dafür, dass Daten aus der Datenbank auf Objekte in der Applikation gemappt werden.

```
@RequestMapping("current/{sessionId}")
public Cart getCurrentCart(@PathVariable String sessionId) {
    Cart cart = repository.findCartBySessionId(sessionId);
    if(cart == null) {
        cart = new Cart(sessionId);
    }
    repository.save(cart);
    return cart;
}
```

Abbildung 12: CartRestController

Die *CartRestController* Klasse ist ähnlich aufgebaut, nur dass hier noch weitere Funktionen zum manipulieren des Warenkorbs hinzugefügt werden. Neben einer Funktion, welche alle Warenkörbe zurückliefert, wird eine weitere eingebaut, welche den Warenkorb des aktuellen Nutzers zurückgibt. Dabei wird wieder die Session ID zur Identifizierung genutzt. Die Session ID muss dieses Mal allerdings über die URL an die REST Schnittstelle mit übergeben werden, da sich die Session ID innerhalb des REST Controllers von der des normalen Controllers unterscheidet. Denn der Rest Controller erstellt für jede Anfrage eine neue Session ID. Die anderen Funktionen sind für das Hinzufügen und Entfernen von Produkten zum Warenkorb zuständig. Auch diese Funktionen benötigen als URL Parameter die Session ID, um den betroffenen Warenkorb identifizieren zu können. Alle Klassen, um die Datenbank zu nutzen sind fertig, nun müssen noch die bestehenden Klassen angepasst werden. Alle Model

Klassen werden entfernt, da ihre Funktionalität durch die Repository Klassen und REST Controller übernommen wurden.

```
@GetMapping("id/{id}")
public ModelAndView showProduct(@PathVariable("id") String id) {
    RestTemplate restTemplate = new RestTemplate();
    Product product = restTemplate.getForObject( url: REST_PRODUCT_URL + "/id/" + id, Product.class);
    if (product != null) {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject( attributeName: "product", product);
        modelAndView.setViewName("productDetail");
        return modelAndView;
    }
    return new ModelAndView( viewName: "redirect:/products/all");
}
```

Abbildung 13: ProductController

Nur die Controller Klassen werden nun noch angepasst. In den Funktionen des *CartControllers* und denen des *ProductControllers* wird eine Instanz der sogenannten *RestTemplate* Klasse genutzt. Damit können Daten von den REST Controllern abgefragt werden. Im *CartController* wird jeweils eine Anfrage an den Rest Controller gestellt, welcher den aktuellen Warenkorb zurückliefert. Dieser Warenkorb wird dann an das Template übergeben. Ähnlich ist dieses Vorgehen auch im *ProductController*. Dort wird ebenfalls eine Anfrage über die *RestTemplate* Klasse an den REST Controller gestellt und die zurückgelieferten Ergebnisse werden an das Template übergeben.

6 Praxiseinsatz

6.1 Vorbereitung

Die Aufgaben und das dazu entwickelte Projekt wurden in einem Schülerpraktikum getestet. Innerhalb zwei Wochen mussten insgesamt neun Schüler die Aufgaben bewältigen. Nur einer von neun Schülern hatte bereits Erfahrungen in der Programmierung sammeln können und der Rest hatte bislang nur wenig bis gar keine Programmiererfahrung.

<table>
<tr><th>Zeit</th><th>Tag 1</th><th>Tag 2</th><th>Tag 3</th><th>Tag 4</th><th>Tag 5</th><th>Tag 6</th><th>Tag 7</th><th>Tag 8</th><th>Tag 9</th></tr>
<tr><td>0,5h</td><td>Begrüßung</td><td rowspan="15">Phase 1</td><td>Grundlagen</td><td rowspan="15">Phase 2</td><td rowspan="15">Gruppen-arbeit</td><td>Grundlagen</td><td rowspan="15">Phase 3</td><td rowspan="15">Gruppen-arbeit</td><td rowspan="15">Scratch</td></tr>
<tr><td>1h</td><td>Grundlagen</td><td>Projekt-einrichtung</td><td>Projekt-einrichtung</td></tr>
<tr><td>1,5h</td><td>Projekt-einrichtung</td><td rowspan="14">Phase 2</td><td rowspan="14">Phase 3</td></tr>
<tr><td>2h</td><td rowspan="13">Phase 1</td></tr>
<tr><td>2,5h</td></tr>
<tr><td>3h</td></tr>
<tr><td>3,5h</td></tr>
<tr><td>4h</td></tr>
<tr><td>4,5h</td></tr>
<tr><td>5h</td></tr>
<tr><td>5,5h</td></tr>
<tr><td>6h</td></tr>
<tr><td>6,5h</td></tr>
<tr><td>7h</td></tr>
</table>

Abbildung 14: Ablaufplan

Für das Praktikum wurde im Vorfeld eine zeitliche Planung entworfen. Diese sieht vor, dass der erste Tag mit einer Einführungs- und Vorstellungsrunde beginnt. Darauf folgt eine kurze Grundlagen Erklärung von HTML, CSS und JavaScript, gefolgt von der Einrichtung der Projekte. Den Rest dieses Tages und am nächsten Tag werden die Aufgaben der ersten Phase durchgeführt. Den Rest der ersten Woche wird an der Phase 2 gearbeitet. Dazu wird wieder erst ein kleiner Grundlagen Vortrag gehalten und die Projekte werden eingerichtet. Zum Schluss der zweiten Phase werden nicht die normalen Aufgaben gelöst, sondern es wird eine Gruppenarbeit stattfinden. Die Gruppenarbeit wird abgeschlossen, durch einen Vergleich der Ergebnisse der

Gruppen. In der zweiten Woche wird die Phase drei behandelt. Auch hier wird wieder damit begonnen, die Projekte einzurichten und die Grundlagen zu vermitteln. Am vorletzten Tag wird wieder eine Gruppenarbeit am Projekt erfolgen. Auch hier werden wieder ein Review und Vergleich der Ergebnisse durchgeführt. Am letzten Tag wird als Abschluss des Praktikums spielerisch programmiert. Dazu wird Scratch genutzt. Scratch ist eine Webseite, mit welcher verschiedene Animationen oder auch Spiele grafisch programmiert werden können.

6.2 Durchführung

Es werden am ersten Tag wie geplant die Aufgaben zu HTML behandelt. Die Schüler erzielen schnell Resultate bei den ersten Aufgaben und kommen relativ schnell voran. Es herrscht eine sehr proaktive Kommunikation. Wenn Fehler auftreten wird immer erst nachgefragt, bevor selbst Fehler gesucht werden. Obwohl es nur selten Momente gibt, in denen die Schüler nicht weiterkommen, treten häufiger Probleme auf, welche durch das kopieren und einfügen von Code entstehen, ohne dabei den Code konkret nachzuvollziehen. Nur ein paar Aufgaben waren in der Umsetzung etwas schwierig. Darunter zählen vorrangig Aufgaben in denen Videos oder Icons hinzugefügt werden sollen. Das Suchen und Einbauen dieser Medien ist zu zeitaufwändig.

Am zweiten Tag haben alle Schüler die einführenden Aufgaben abgeschlossen und beginnen mit den normalen Aufgaben. Es treten teilweise noch häufiger Fehler auf, welche durch falsche Syntax zu Stande kommen. Besonders bei HTML werden häufig Tags nicht richtig geschlossen. Die meisten Schüler fangen an die ersten Aufgaben mit JavaScript zu behandeln. Bei dem Verständnis von JavaScript gibt es größere Probleme. Da keiner der Schüler selbst in dem angegebenen Begleitmaterial sucht, kommt es hier zu vielen Hilfe-Anfragen. Weder das Prinzip noch die Funktionsweise von JavaScript sind für die Schüler nachvollziehbar. Wegen der auftretenden Probleme wird der Zeitplan etwas abgeändert. Zum einen wird die erste Phase auf den

dritten tag ausgeweitet und zum anderen wird nochmal eine kurze Erklärung von JavaScript abgehalten.

Am dritten Tag wird aufgrund der erkennbaren Verständnisprobleme für JavaScript nochmals eine Erklärung gegeben, und eine kurze Übungsrunde, in der jeder Schüler ein paar Zeilen Code nachvollziehen muss und die Auswirkungen erklären soll. Außerdem hat jeder Schüler eine JavaScript Datei erhalten, in welcher die grundlegenden Code Elemente erklärt werden. In der Übung konnte jeder Schüler mit leichten Hilfestellungen die Fragen beantworten und den JavaScript Code nachvollziehen. Allerdings gibt es im Verlaufe der Aufgaben wieder Rückschläge. Das Verständnis von JavaScript ist immer noch nicht gegeben. Aufgrund sinkender Motivation wird die Aufgabe für den Rest des Tages etwas abgeändert. Die Schüler welche unmotiviert sind, können weiter an der Gestaltung ihrer Seite arbeiten. Bis auf einen Schüler hat keiner die JavaScript Aufgaben weiterbearbeitet. Die Bearbeitung der ersten Phase wird damit beendet.

Am vierten Tag wird zunächst das Projekt der zweiten Phase eingerichtet. Um ähnliche Probleme, wie bei JavaScript zu vermeiden wird der ganze Tag dazu genutzt die Java Grundlagen zu vermitteln und mit dazugehörigen Übungen den Kenntnisstand zu prüfen. Es treten trotz mehrfachen Erklärungen weiterhin Syntax Probleme auf. Bei den Übungen und Beantwortung der Fragen haben die meisten Schüler allerdings positive Resultate erzielt.

Am darauffolgenden Tag wird mit den Aufgaben der Phase Zwei begonnen. Zunächst sind sehr schnelle Fortschritte bei der Lösung der Aufgaben zu verzeichnen, aber auch hier kommt es besonders durch die verschiedenen Technologien immer wieder zu diversen Verständnisproblemen. Vor allem das Prinzip von Thymeleaf ist nicht klar. Andere häufige Probleme hängen besonders mit dem Umgang mit Spring Boot zusammen, da die Applikation häufig nicht richtig neugestartet wird und es so zu

Problemen kommt. Trotz mehrfacher Erklärung des Fehlers und der Fehlernachricht wird das Problem nicht von den Schülern selbst behoben.

In der zweiten Woche wird mit einer Retrospektive zum Praktikum begonnen. In dieser Retrospektive gibt jeder Schüler Rückmeldung zu den Themen, was hat gut funktioniert, was hat schlecht funktioniert, was haben sie gelernt und was hat sie überrascht (siehe Anlage 1). Im Allgemeinen wurde besonders die Kommunikation, die Zusammenarbeit und die Gestaltung der Webseite als positiv und gut funktionierend eingeschätzt. Die Schwierigkeit der Aufgaben wird jedoch als zu hoch empfunden. Viele Schüler führen HTML, CSS und JavaScript unter den gelernten Themen auf. Der Punkt „Überrascht“ beinhaltet besonders zum einen die Schwierigkeit der Aufgaben, zum anderen aber auch die Auswirkungen von kleinen Syntax Fehlern.

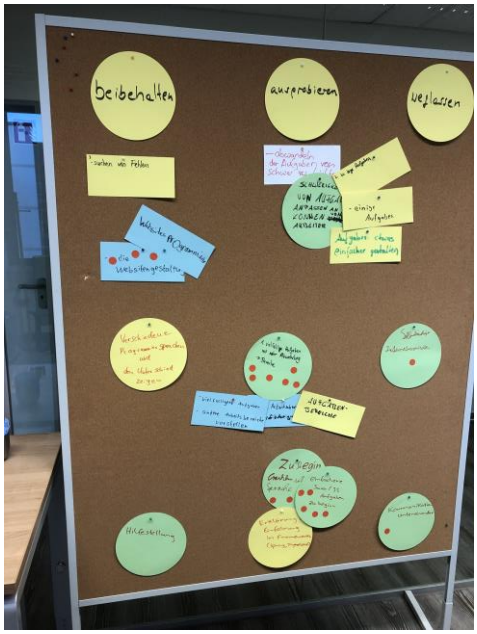

Abbildung 15: Retrospektive

Als nächstes wird in der Retrospektive festgehalten, welche Änderungsvorschläge die Schüler haben und was beibehalten werden soll (siehe Abbildung 15). Bei den Änderungen wird ziemlich oft vermerkt, dass es weniger schwierige Aufgaben geben sollte. Außerdem stimmen einige für die Einführung von mehr leichten Aufgaben. Im

Anschluss ordnen die Schüler ihre Vorschläge einer Wertigkeit zu und verteilen dazu zwei Punkte an die Vorschläge. Neben diversen anderen Vorschlägen erhält bei der Wertung vor allem der Punkt, dass mehr leichte Aufgaben existieren sollen die höchste Wertung. Das Thema mit weniger schlechten Aufgaben erhält hingegen keine Wertung.

Am nächsten Tag wird ebenfalls mit den Aufgaben der zweiten Phase fortgefahren. Die Motivation nimmt stark ab und nur wenige arbeiten teilweise konzentriert an einer Lösung für die Aufgaben. Um dem entgegen zu wirken wird mit der Gruppenarbeit begonnen. Die Praktikanten werden in zwei Gruppen eingeteilt und beide Gruppen sollen sich eine Seite überlegen, welche sie erstellen wollen und ein Konzept dafür entwickeln. Die beiden Gruppen präsentieren ihre Ergebnisse und beide haben sich für die Erstellung eines Browserspiels entschieden.

Am darauffolgenden Tag arbeiten beide Gruppen an der Umsetzung ihrer Ideen und auch hier ist wieder eine sinkende Motivation zu verzeichnen. Am Ende des Tages haben beide Gruppen das Layout und Design für die Seite fertiggestellt, allerdings ist noch keine Funktionalität zu verzeichnen.

Am letzten Tag wurde wie geplant zur Motivationssteigerung mit Scratch grafisch programmiert. Hier erstellen alle Schüler verschiedene Spiele, wie beispielsweise Pong, Snake und Super Mario. Es sind schnelle Erfolge festzuhalten. Jeder Schüler ist motiviert und hat bereits nach kurzer Zeit das erste Spiel erstellt.

6.3 Auswertung

Besonders in der ersten Woche werden bereits viele Probleme beim Verständnis der Grundlagen verzeichnet. Selbst genaue Erklärungen tragen nicht zur Klärung der Probleme bei. Das Begleitmaterial wird nicht genutzt und auch die von Programmen zurückgegebenen Fehlermeldungen werden nicht betrachtet. Die Retrospektive zeigt,

dass es mehr leicht lösbare Aufgaben geben muss, um die Motivation zu erhalten. Das Prinzip, dass sich Schüler an Aufgaben länger aufhalten sollen, um sie zu lösen zeigt weniger positive Resultate. Die Schüler sind, wenn sie nicht weiterkommen sehr schnell demotiviert. Hingegen sind leichte Aufgaben eher motivierend und werden gerne bearbeitet.

Es müssen noch ein paar Anpassungen an dem Aufgabenkonzept vorgenommen werden. Zum einen müssen mehr einfachere Aufgaben zur Verfügung gestellt werden, um die Schüler nicht so schnell mit schwierigeren Aufgaben zu konfrontieren. Zum anderen sollten einige Tage für grundlegende Einführungen in die Technologien verplant werden. Besonders im Bereich Java und JavaScript müssen im Vorfeld einführende Vorträge und Übungen festgelegt werden. Das soll dazu dienen, die Nachfrage bei Fehlern von Seiten der Schüler zu verringern. Die Informationen zu Begleitmaterialien können größtenteils weggelassen werden, da diese keine Anwendung fanden. Die Planung für die zeitliche Einteilung des Praktikums muss auch angepasst werden, da die Aufgaben ein so schnelles Durchgehen der Phasen nicht ermöglichen. Die Anzahl der Gruppenarbeiten kann ebenfalls reduziert werden. Durch die proaktive Kommunikation der Schüler untereinander ist die Bearbeitung der Aufgaben teilweise als eine Gruppenarbeit zu bezeichnen. Es fehlt zwar die Planung und Konzeption innerhalb eines Teams, allerdings wird die Umsetzung meistens durch gegenseitiges Austauschen realisiert.

7 Zusammenfassung

In dieser Projektarbeit wurde ein Konzept für die Heranführung an die Programmierung für Schülerpraktikanten entworfen. Zur Erstellung dieses Konzeptes wurde zunächst betrachtet, welche bestehenden Aufgaben in der dotSource GmbH vorhanden sind. Daraufhin wurden die Aufgaben konzeptioniert. In der Konzeption wurden die Grundlagen für technische Prototypen und den Ablauf eines Praktikums festgelegt. Anschließend wurde die Erstellung der technischen Prototypen beschrieben. Außerdem wurden die spezifischen Aufgaben zu den Prototypen festgelegt. Zum Schluss wurde ein Bericht und eine Auswertung zu einem Probedurchlauf des Aufgabenkonzeptes verfasst und anhand dessen analysiert, welche Änderungen noch vorgenommen werden müssen.

Die in der Konzeption entworfene Planung konnte in der Durchführung nicht eingehalten werden und musste abgeändert werden. Trotzdem wurde der gewünschte Effekt erzielt, dass sich die Praktikanten an für sie schwierigen Aufgaben aufhalten.

Es konnte ein Aufgabenkonzept erstellt werden, welches der Heranführung an die Programmierung dient. Es müssen noch einfachere Aufgaben hinzugefügt werden, aber dennoch stellen diese Aufgaben ein Konzept dar, welches keinen hohen Einarbeitungsaufwand für betreuende Mitarbeiter darstellt. Die Beschäftigung selbst von bereits erfahreneren Praktikanten ist mit diesen Aufgaben über mehrere Wochen problemlos möglich. Auch ein Anstieg der Lernkurve ist zu vermerken, da eine Steigerung von der Erstellung von HTML Dateien über das Arbeiten mit JavaScript bis hin zur Nutzung bereits existierender Anwendungen erzielt wurde. Die gestellte Anforderung wurde also erreicht und das Konzept wird von der dotSource GmbH in weiteren Schülerpraktika eingesetzt.

Damit in Zukunft die Planung eingehalten werden kann muss darauf geachtet werden, dass bei der Konzeption der Aufgaben darauf geachtet wird, dass das technische Verständnis der Praktikanten sehr gering sein kann. Es müssen also nicht nur

genügend leicht lösbare Aufgaben eingeplant werden, sondern auch genügend Zeit, welche für die Einführung und Erklärung einzelner Themenbereiche gedacht ist. Auch wenn ohne eine genaue Einführung in die Technologien der Betreuungsaufwand gegeben ist, sollten die ersten Tage, bzw. die Anfänge der einzelnen Phasen mit Grundlagen und Übungen beginnen. Es sollte unbedingt weiterhin Aufgaben geben, an denen sich die Schüler länger aufhalten, um sie zu lösen. Denn auch wenn dadurch nicht sofort ein Erfolgserlebnis gegeben ist, spiegelt dieses Szenario die Arbeit eines Entwicklers wieder. Denn es kommt in der Entwicklung häufiger vor, dass man vor Probleme gestellt wird, welche in der Lösung nicht einfach sind.

V Literaturverzeichnis

[Gra18] Gradle, „Gradle User Manual", o.O., 2018.
https://docs.gradle.org/current/userguide/userguide.html
Abruf am: 06.02.2019

[Agi] Agile Alliance, „Automated Build", o.O., o.J..
https://www.agilealliance.org/glossary/automated-build
Abruf am: 06.02.2019

[Boo] Bootstrap, „Bootstrap", o.O., o.J..
https://getbootstrap.com/
Abruf am: 06.02.2019

[Thy] Thymeleaf, „Thymeleaf", o.O., 2018.
https://www.thymeleaf.org/
Abruf am: 06.02.2019

[Spr] Spring, „Spring Boot", o.O., o.J..
https://spring.io/
Abruf am: 06.02.2019

[Cos17] Cosmina, I., Harrop, R., Schaefer, C., Ho, C.: „Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools", 5. Auflage, Verlag Apress, o.O., 2017

[Wal15] Walls, C.: „Spring Boot in Action", o.A., Verlag Manning, o.O., 2015

[Bit18] Bitkom, „82.000 freie Jobs: IT-Fachkräftemangel spitzt sich zu", o.O., 2018.
https://www.bitkom.org/Presse/Presseinformation/82000-freie-Jobs-IT-Fachkraeftemangel-spitzt-sich-zu
Abruf am: 11.02.2019

[Mon] MongoDB, „What is MongoDB?", o.O., o.J..
https://www.mongodb.com/what-is-mongodb
Abruf am: 25.02.2019

VI Anlagen

hat gut funktioniert	hat nicht gut funktioniert	gelernt	überrascht
Kommunikation untereinander	Aufgaben teilweise zu schwer	HTML & Java kann ich jetzt teilweise verstehen	Schwierigkeit der Aufgaben
Zusammenarbeit	manchmal gab es sehr schwere Aufgaben	Aufbau von Webseiten	Vielseitigkeit von Aufgaben
die Zusammenarbeit in der Gruppe	Grundlagen Vermittlung hat nicht richtig funktioniert	Grundlagen in HTML, CSS und JavaScript	die Komplexität eigentlich simpler Aufgaben
das Programmieren und die Lösungssuche im Team		viele Tipps, Tricks und Informationen zum Thema Webseiten programmieren (Grundlagen für HTML, CSS, JavaScript)	ein Zeichen kann ausschlaggebend sein
gestalten der Webseite		HTML, CSS, JavaScript	dass hinter einer Webseite so viel Arbeit steckt
Organisation und Absprachen wurden eingehalten		Anwendung von HTML / CSS	
		Grundlagen HTML, CSS, JavaScript	

Anlage 1: Retrospektive

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich,

1. dass ich meine Studienarbeit mit dem Thema:

   Ausarbeitung eines praktischen Lernkonzeptes zur Heranführung an die Programmierung mit sukzessive ansteigender Lernkurve

   ohne fremde Hilfe angefertigt habe,
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe und
3. dass ich meine Studienarbeit bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

_____ _____

Ort, Datum