Praktikumsbericht
Datenbanken und verteilte Systeme

Gruppe 6

Baum123

Hagen Peukert

Natalie Gornicki

Oliver Wichmann

David Wedeleit

Verlauf und Reflektion des Praktikums

Dieser Bericht soll die Arbeit im Praktikum “Verteilte Systeme und Datenbanken”, sowie die in dieser Zeit erworbenen Erfahrungen widerspiegeln und einen Überblick in den Entwicklungsprozess des Online-Shops geben, der innerhalb der drei zur Verfügung stehenden Wochen entstanden ist.

In einem ersten Schritt soll hierbei die Tutorialphase beschrieben werden, in der auf die genutzten Technologien und das Kennenlernen selbiger eingegangen wird. Danach soll das Vorgehen in der Entwurfsphase geschildert werden, wobei der Funktionsanalyse und dem daraus resultierenden Entwurf eine besondere Bedeutung zukommt. Im Anschluss erfolgt eine Beschreibung der ersten Entwicklungsphase zusammen mit der Realisierung der im Vorfeld erläuterten Funktionalitäten, sowie den entstandenen Abweichungen vom Erstentwurf. Es folgt die zweite Entwicklungsphase, die sich primär mit der Ergänzung fehlender Funktionalitäten und dem Entfernen der letzten Fehler der Anwendung beschäftigt, sowie ein abschließendes, aus der Veranstaltung gezogenes Fazit.

Tutorialphase:

Nach dem ersten Zusammenfinden der Gruppenmitglieder und einem ausführlichen Überdenken der Möglichkeiten, die zum Herstellen einer Datenbank-Anwendung gegeben waren, wurde sich dafür entschieden, ein Projekt mithilfe von JavaScript und Baqend zu entwickeln. Grund hierfür war bei allen Beteiligten das Bedürfnis sich mit einer Webanwendung auseinander zu setzen und der Wunsch, im Zuge des Praktikums neue Technologien kennen zu lernen.

Die ersten drei Tage des Praktikums wurden daher verstärkt in die Aquirierung des Fachwissens investiert, das in den kommenden zweieinhalb Wochen zur Anwendung kommen sollte. Hierzu zählte das Erlernen oder Auffrischen der bereits genannten Skriptsprache JavaScript, der Umgang mit Baqend, HTML und CSS, sowie die Verwendung der Entwicklungsumgebung IntelliJ und der Versionskontrolle durch Git.

Während der Einarbeitung in die Funktionalitäten Baqend's entstand aus dem eher zufällig gewählten Gruppen-Namen "Baum123" die Idee, einen gleichnamigen Online-Shop für Pflanzen- und Garten-Produkte zu entwickeln. Im Zuge des Übens des praktischen Umgangs mit HTML und CSS wurde bereits in der Tutorial-Phase auf einen Prototypen einer mehrseitigen Shop-Oberfläche hingearbeitet, der als Treppenstufe für die kommenden Iterationen der Anwendungen dienen sollte.

Entwurfsphase

Da das Programmierteam hinter “Baum123” mit vier Personen eine sehr geringe Größe aufwies, wurde sich dafür entschieden, Entwurf und Planung gemeinsam durchzuführen. Zwar existierte unter den Programmieren eine grobe Aufteilung in den Frontend-Bereich für das Design und Umsetzen der Website, der Funktionalität

der Seite und dem Bereich der Datenbank, allerdings wurden alle Entwicklungsentscheidungen gemeinsam, unabhängig von der Zuteilung, getroffen.

Die detaillierte Ausarbeitung der Grundidee erfolgte hauptsächlich über das mündliche Gespräch und unter Zuhilfenahme digitaler Notizzettel in verschiedenen Anwendungen, sowie Skizzenerstellungen auf der Website www.draw.io .

Der erste Schritt in der Planung der Software war eine Anforderungsanalyse. Hier wurden die Möglichkeiten und Wünsche eines Kunden von typischen Onlineshops genommen und auf die wesentlichen Anforderungen reduziert, die für das entstehende Programm obligatorisch sein würden.

Anforderungen für den Kundenbereich

Die **Orientierung im Produktbestand** durch eine Suche oder Vorschläge ist die wohl wichtigste Anforderung an einen Onlineshop, um Waren übersichtlich zu präsentieren und so eine Auswahl der Produkte erst möglich zu machen. Zusätze wie Favoriten, Filter oder Sortierverfahren geben dem Anwender weitere Optionen zur Verkleinerung des Suchraums und ergänzen diese Grundanforderung sinnvoll.

Die zweite Säule eines Webshops ist die **Produktdarstellung im Detail**. Da der Kunde die Ware selbst nicht untersuchen kann, muss diese Darstellung alle wichtigen Kennzahlen des Produkts verständlich aufbereiten. Ein schnell zu erreichender Warenkorb erleichtert an dieser Stelle zudem die Kaufentscheidung.

Die dritte Hauptanforderung für den Kundenbereich ist die unkomplizierte **Abwicklung des Kaufvorgangs**. Ein stets erreichbarer digitaler Warenkorb, der zum Kauf bestimmte Produkte sammelt, übersichtlich darstellt und schnelle Korrekturen ermöglicht, vermittelt dem Kunden ein Gefühl der Kontrolle. Wenige Handgriffe bis zum Kauf laden zu einem schnellen Abschluss des Geschäfts ein. Um Käufe Kunden zuordnen zu können ist es hier zweckmäßig, eine Art Konto mit Login einzurichten.

Anforderungen für den Administrationsbereich

Während die Anforderungen auf Kundenseite praktisch untersucht werden konnten, waren die auf der Seite des Vertreibers nur in der Theorie ermittelbar. Als Betreiber der Webpräsenz muss dieser unter anderem in der Rolle des Administrators des Datenbestands agieren, verfügt jedoch in der Regel über wenig technische Kenntnisse, wie dieser manipuliert werden kann, was beachtet werden muss.

Eine **übersichtliche Auflistung** der Produkte ist die Grundlage für die **Pflege der Daten**. Dafür benötigt der Betreiber ein Formular, indem die digitalen Repräsentationen der realen Waren mit ihren Kennzahlen aufgelistet werden und der Preis, die Stückzahl und weitere hinterlegte Angaben verändert werden können.

Dieses Formular dient der Pflege schon bestehender Produkte, wenn zum Beispiel eine neue Lieferung ankommt, oder dem Entfernen solcher, wenn sie nicht mehr angeboten werden.

Für noch nicht eingegliederte Artikel benötigt der Administrator zudem ein Formular, mit welchem er neue virtuelle Gegenstücke erstellen kann.

Abschließend dürfen alle diese Funktionen nur dem Verkäufer zustehen, aber keinem anderen. Um das zu realisieren, ist ebenso eine Form des Logins notwendig.

Fast alle Ergebnisse dieser Anforderungsanalyse wurden als Zielsetzung für die eigene Applikation übernommen, wenn auch unter Bedingung, sie von der Umsetzung her simpler zu gestalten als bei den „professionellen Mitbewerbern“. Im folgenden soll der erste Entwurf der Anwendung vorgestellt werden.

Entwurf des Datenbankschemas

Das erstellte Datenbankschema orientiert sich direkt an den vorher formulierten Anforderungen für den Onlineshop.

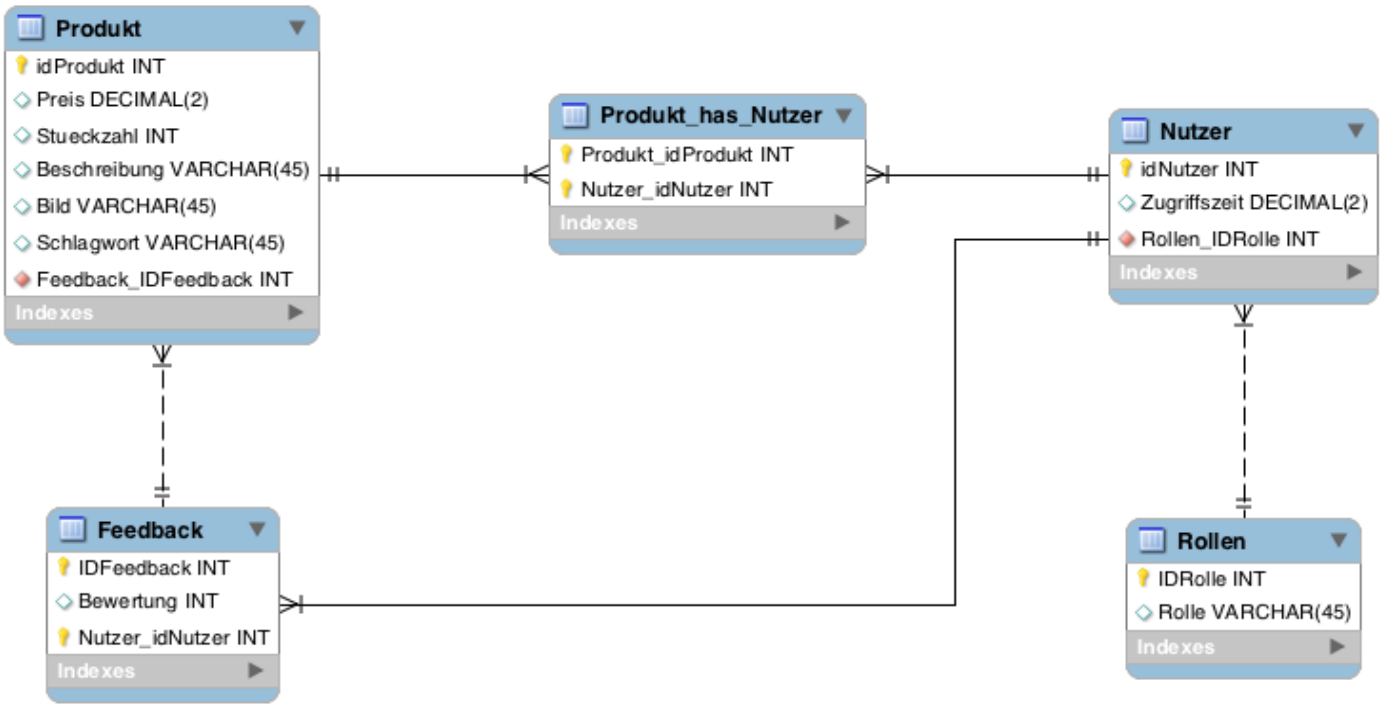


Abb1: Entity-Relationship-Diagram für den Onlineshop Baum123

Die Tabelle PRODUKT enthält die Daten, mit denen der Hauptteil der Funktionalität des Online-Shops umgesetzt wurde. Über diese Tabelle wird die Anzeige, Suche, Filterung aller wichtigen Produktinformationen sowie der Wareneingang gesteuert.

Neben dem Produktnamen enthält sie den Preis, die noch vorhandene Stückzahl, eine Beschreibung, eine Bildreferenz und ein Datenfeld für Schlagwörter.

Um jedem Produkt eine Bewertung zuzuordnen, ist die Entität FEEDBACK als 1:n Beziehung modelliert. So können jedem Produkt beliebig viele Bewertungen zugewiesen werden. Jeder Bewertung kann jedoch auch der Nutzer eindeutig zugeordnet werden, da über einen Fremdschlüssel auf die Tabelle NUTZER gezeigt wird. Auch hier liegt eine 1:n-Beziehung vor: Jeder Nutzer kann beliebig viele Bewertungen vornehmen.

Die Realisierung einer 1:n-Beziehung wird in Baquend mittels einer Listen-Datenstruktur umgesetzt. Dies bedeutet, dass zum Beispiel die Tabelle FEEDBACK als ein Listenfeld in der Tabelle PRODUKT angelegt wird. Dabei enthält jedes einzelne Listenelement die Bewertungen und die Nutzer. Da die NUTZER-Tabelle hier zu den Standardeinstellung des Baqend Frameworks gehört, wird sie wiederum nicht als Liste eingebunden, sondern erhält eine Verlinkung auf die ID der Nutzer.

Um die Nutzer mit ihren gekauften Produkten zu verbinden, wurde eine n:m-Beziehung über die Tabelle KAUF realisiert. Sie ordnet jeweils die Schlüssel der Produkte, den Schlüsseln der Nutzer zu. Zudem wurde sich dafür entschieden, der Tabelle KAUF zwei weitere Datenfelder - Kaufzeitpunkt und Anzahl der gekauften Produkte - hinzuzufügen, die sich als besonders nützlich für die Funktionalität der Kennzahlenberechnung erweisen würden (nicht in Abb 1). Aus diesem Grund wurde auch der Tabelle NUTZER noch das Datenfeld Zugriffszeit hinzugefügt.

Letztlich sind die Tabellen NUTZER und ROLLEN standardmäßig als 1:n modelliert. Jeder Nutzer erhält genau eine Rolle. Dies braucht sinnvollerweise nicht ergänzt oder verändert zu werden.

Entwicklungsphase 1

Da sich in der Einarbeitungsphase bei allen Mitgliedern der Gruppe verschiedene Stärken im Umgang mit den eingesetzten Technologien herauskristallisiert hatten, wurde versucht, eine Arbeitsaufteilung zu gewährleisten, die diese Stärken am besten ausspielt. Für jeden Arbeitstag wurden daher kleine, in Abhängigkeit zum technischen Einsatzgebiet stehende Teilziele festgelegt, die im Verlaufe dieses Tages erreicht werden sollten. Es konnte beobachtet werden, dass diese Arbeitsweise in den Anfängen der Entwicklung durchaus gute Erfolge erzielte, mit dem Voranschreiten der Herstellung jedoch immer weniger angewandt wurde, weil erst später entstehende, komplexere Aufgabenstellungen oftmals mehr als einen der Programmierer erforderten, um sie zu lösen.

Auf den folgenden Seiten sollen die in dieser Zeit entstandenen Umsetzungen näher besprochen werden.

Website

Für die Herstellung der Webfront für den Kunden konnte auf den bereits erwähnten Prototypen zurückgegriffen werden. Anders jedoch, als im ursprünglichen Entwurf geplant, entwickelte sich diese im Verlauf der Entwicklungsphase von einer mehrseitigen Anwendung zu einer “One-Page-Application“, nachdem im Nachhinein die Vorteile der anfangs nicht eingeplanten Einbindungen „Jquery“ und „Bootstrap“ erkannt wurden. Durch das Nutzen dieser konnte die Umsetzung der Shop-Oberfläche auf einer HTML-Seite übersichtlicher und dynamischer erfolgen als auf mehreren. Interaktionen mit verschiedenen Komponenten bewirkten nun anstatt Verlinkung, dass Elemente versteckt oder wieder angezeigt wurden.

Der Administrationsbereich wurde auf einer seperaten HTML-Oberfläche umgesetzt, die optisch und technisch einfacher aufbereitet wurde, da sie keine Werbewirkung entfalten muss.

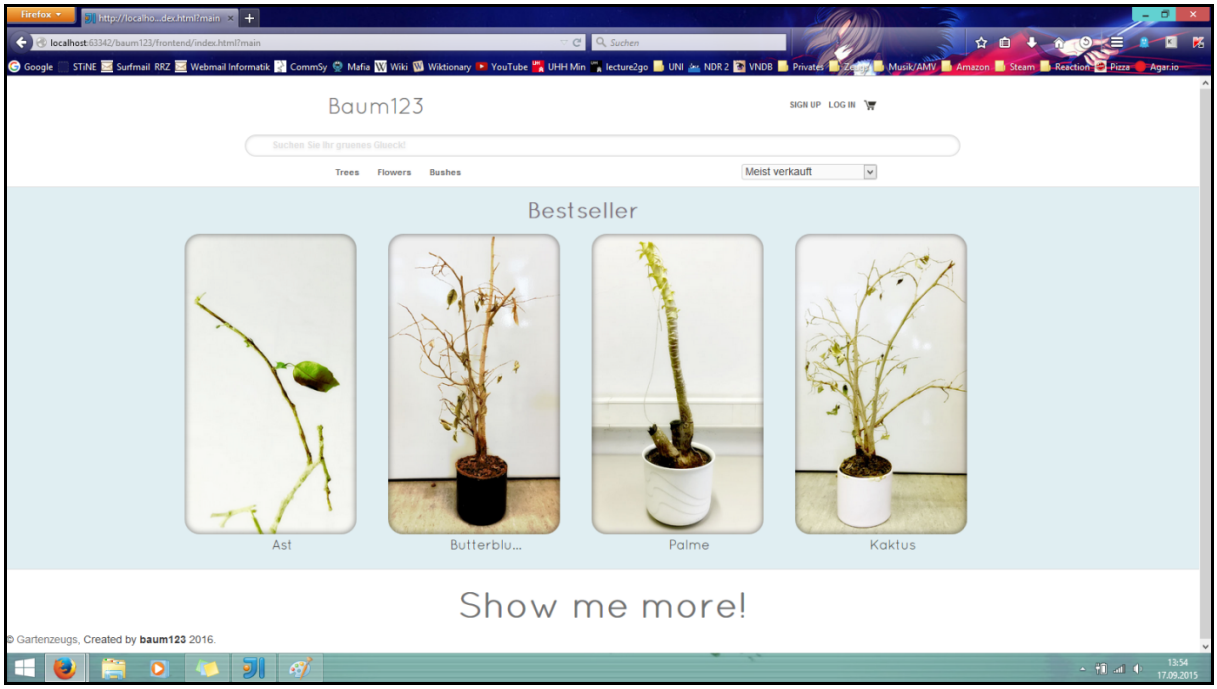


Abb2: Hauptseite

Abb2:
Auf der Hauptseite werden die Top 4 der meistverkauften Produkte angezeigt, allerdings nur dann, wenn sie auch auf Lager sind. Weiterhin werden nur Produkte berücksichtigt, die auch ein Bild besitzen, um Fehldarstellungen zu vermeiden. Jedes Produkt kann angeklickt werden, woraufhin in eine Produktdetailansicht gewechselt wird (Abb5).

Zudem ist im oberen Bereich ist die Möglichkeit gegeben sich einzuloggen. Hat der Nutzer dabei die Rolle des Administrators, wird er auf die Produkt-Administrationsseite weitergeleitet. Weiterhin zu sehen ist ein Warenkorbsymbol. Hier wurde dafür gesorgt, dass beim Anklicken in den Warenkorb navigiert wird. Dieser konnte in der ersten Entwicklungsphase vorerst nur in eingeschränktem, noch nicht voll funktionsfähigem Maße realisiert werden.

Von der Landing-Page gibt verschiedene Wege, um sich mehr Produkte anzeigen zu lassen:

- Über die **Suchleiste** können Suchbegriffe eingegeben werden. Suchergebnisse werden in Echtzeit angezeigt (Abb3).
- Über „**Show me more**“ werden einem weitere Produkte nach dem Kriterium der meist Verkauften Artikel aufgelistet. An dieser Stelle werden auch Produkte berücksichtigt, die aktuell nicht auf Lager sind.

- Über die Auswahl eines der drei **Filter** „Trees“, „Flowers“ und „Bushes“ werden jeweils alle Artikel zu einer Kategorie angezeigt (Abb4).
- Über das **Change-Menü** kann gewählt werden, nach welchem Kriterium man sich die Produkte anzeigen lassen möchte, findet hier eine Änderung statt, werden alle Produkte sortiert nach dem gewählten Kriterium geladen .

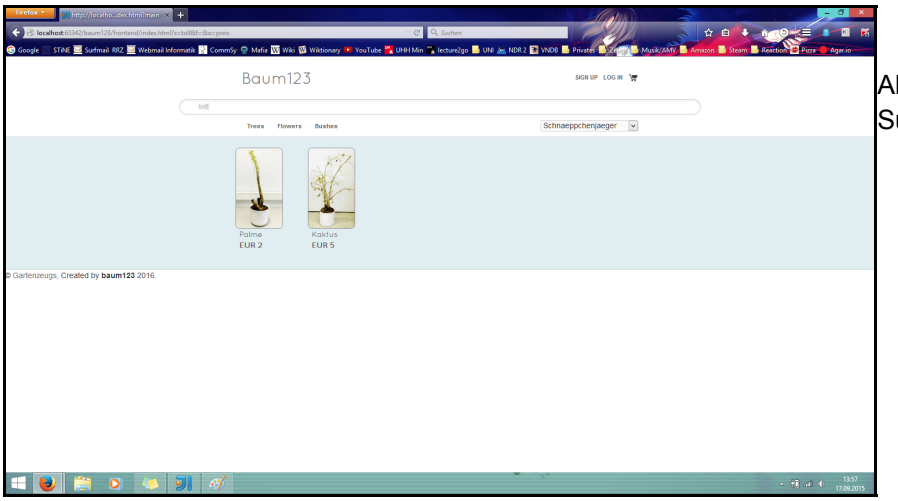


Abb3: Ergebnisse bei einer Suche

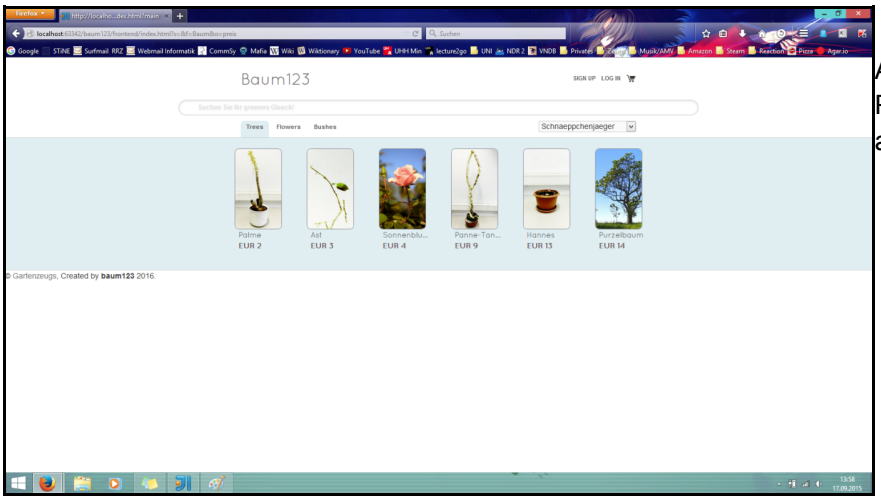


Abb4: Anzeige von Produkten wenn ein Filter aktiviert ist

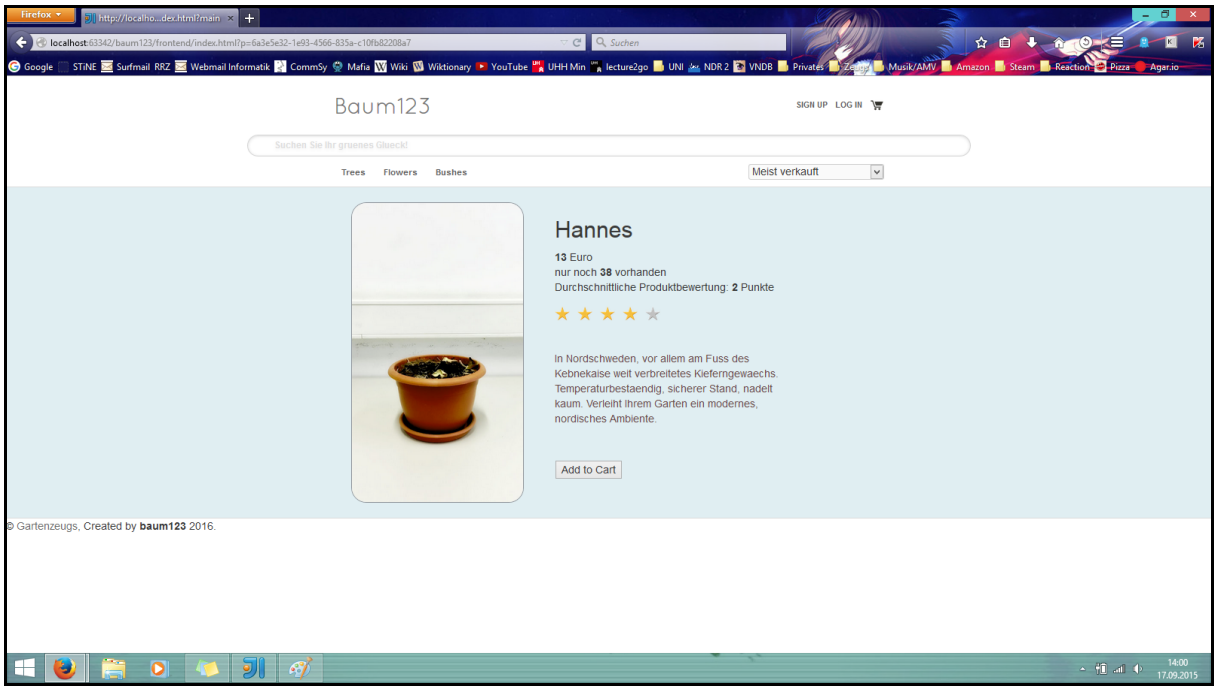


Abb5: Detailseite eines Produktes

Abb5:
Auf der Produktdetailseite werden alle relevanten Daten des Produktes angegeben. Bild, Name, Preis und Beschreibung sind statisch und ändern sich nur, wenn durch einen Administrator jeweilige Änderungen vorgenommen worden sind.

Die angezeigte Stückzahl aktualisiert sich in Echtzeit; sobald der Button „Add to Cart“ geklickt wird, reduziert sich diese um „1“, während beim Einkaufswagensymbol die Zahl der enthaltenen Produkte hochgezählt wird.

Von dieser Änderung ist dabei nicht nur die Oberfläche betroffen, auch in der Datenbank wird der Lagerbestand des betroffenen Produktes verringert, während gleichzeitig die Verkaufsstatistik erhöht wird.

Technisch realisiert ist der Wechsel des Inhaltes auf der Webseite durch JQuery Methoden, die abhängig von den Interaktionen mit der Website unterschiedliche DIV sichtbar oder unsichtbar machen. Außerdem wurden über JQuery weitere Methoden aufgerufen, welche Komponenten bei Bedarf komplett neu erstellen und in ein DIV füllen.

```
<div class="jumbotron">
    <div class="container">
        <div class="row bestsellerText">Bestseller</div>
        <div class="row bestsellerRow" id="topProducts"></div>
        <div class="row moreBestseller" id="moreTopProducts"></div>
        <div class="row singleView" id="singleProduct"></div>
        <div class="row cartTopView" id="cartTop">
            <div class="col-md-2"></div>
            <div class="col-md-2">Name</div>
            <div class="col-md-2">Price</div>
            <div class="col-md-2">Quantity</div>
        </div>
```

```
        <div class="row cartView" id="cartPage"></div>
        <div class="row fullPriceView" id="fullPrice"></div>
    </div>
</div>
</div>
```

Für das Darstellen mehrerer Produkte wurde „Bootstrap“ verwendet. Über Klassen wie zum Beispiel „col-md-2“ war es möglich, ein Raster zu benutzen, ohne eine Tabelle um die Komponenten zu erstellen.

Zeitweise entstand an dieser Stelle die Überlegung, ob „Handlebars“ als weitere Technologie für die Oberfläche eingebunden werden sollten. Da das Projekt aber bereits fortgeschritten war und viele schon bestehende Komponenten hätten umgeschrieben werden müssen, wurde sich gegen einen den Einsatz entschieden. Der Makel, dass dynamisch eingefügte Komponenten stets mit HTML in die DIVs übergeben werden mussten, wurde zugunsten des Projektfortschritts in Kauf genommen.

Ausschnitt aus einer Methode, welche solch eine Komponente zusammensetzt:

```
$("#singleProduct").append(
    "<div class=\"col-md-3 singleViewDiv\"><img src=\"" + product.bild +
"\"></div>" +
    "<div class=\"col-md-3 singleViewContent\">" +
    "<div class=\"productTD\"><h2>" + product.name + " </h2></div>" +
...
```

Suche und Ausgabe der Ergebnisse

Im vorangegangenen Abschnitt über den Aufbau der Website wurde bereits angemerkt, nach welchen Interaktionen die Anzeige einer Liste an Waren erfolgt: In Echtzeit nach Eingabe von Suchbegriffen in das dafür vorgesehene Input-Feld, bei Änderung der Einstellungen für Filter oder Sortierung sowie nach einem Klick auf den „Show me more“-Link. Bei jeder dieser Aktionen kommt die Suchfunktion zum Einsatz. Aufgerufen wird diese mit dem Bezeichner “startSearch()”.

Ursprünglich arbeitete die Funktion mit dem im Feld „Name“ abgelegten String eines Produktes als Vergleichsgegenstand, doch nach dem Überdenken dieser Vorgehensweise wurde diese als zu limitierend empfunden. Stattdessen wurde von dem im Entwurf angedachten Datenbankschema abgewichen und der Produkt-Tabelle eine neue Spalte namens „tags“ hinzugefügt. Gesucht wird nun nach Begriffen, die als String in diesem Feld hinterlegt wurden. Dieser enthält immer den Namen des Produktes, wahlweise aber auch andere Wörter, auf die die Suche reagieren soll. Dadurch können auch nicht namens- aber sinnverwandte Ergebnisse zurückgegeben werden, was die Mächtigkeit der Suche, die richtige Wartung vorausgesetzt, ungemein erweitern kann.

Die Aufgaben von startSearch() können ungefähr mit “Wahl der richtigen Baqend-Anfrage auf Basis der ermittelten Informationen”, “Aktualisierung der URL” und “Aufbau der Ergebnisansicht” beschrieben werden.

Bei Aufruf von startSearch() führt diese zuerst showProductOverviewOnly(), eine Funktion, die die Oberfläche des Webshops für die Anzeige der Ergebnisse modifiziert. Im Anschluss springt der Kontrollfluss zu searchBarAction(), in der der Baqend-Befehl ermittelt und die URL modifiziert wird.

Dort wird in einem ersten Schritt der Suchbegriff aus dem Input-Feld ausgelesen und in eine Regular Expression umgewandelt. Diese soll später für eine Query-Anfrage mit “match(...)” zum Einsatz kommen. Im nächsten Schritt wird die Einstellung der Combo-Box, bzw. ihr Value, für die Sortierverfahren ausgelesen, sodass mit einer Switch-Funktion (“sortSwitch()”) die richtige Anfrage für Baqend ausgewählt werden kann. Eine Aufteilung in verschiedene Unterfunktionen, von denen, je nach Sortierung, nur eine aufgerufen wird, war deshalb nötig, weil sich die Querys in bestimmten Punkten gravierend unterscheiden. Die Ordnung nach Feedbacks zum Beispiel kann aufgrund der Repräsentation der Rückmeldungen als Liste von mehreren Objekten, die die Wertung als Feld enthalten, keinen vorgegebenen Query-Befehl aufgreifen, sondern muss erst die resultierende Liste am Ende des Datenbankaufrufs mit “sort(...)” und einer eigens geschriebenen Entscheidungsfunktion ordnen.
Die Anfragen an die Datenbank erfolgen in der Regel nach dem Schema, dass alle Produkte ausgewählt werden,

1. deren Feld “Liste” zu dem in der Variable filter hinterlegten RegEx passt, der mit jedem Klick auf die Kategorie-Reiter aktualisiert wird,
2. deren String im Feld “tags” der aus dem Input gebildeten Regular Expression gleicht,
3. und die ein Bild aufweisen (unfertig eingetragene Produkte werden so herausgefiltert).

Danach erfolgt gegebenenfalls der Aufruf einer aufsteigenden oder absteigenden Sortierung nach dem bereits im Switch verwendeten sort-Parameter und die Bildung der Ergebnisliste. Andernfalls müssen die Inhalte sortiert werden, nachdem sie mit einer Liste zusammengefasst wurden. Im Anschluss wird die Ergebnisliste mit der Funktion „printItemsSmall(...)“ auf der Oberfläche ausgegeben.

Das letzte Ausführung der Suchfunktion ist es, mit der Unterfunktion „urlRefresh(...)“ die URL entsprechend der Eingaben zu modifizieren und für sie einen Eintrag in der Browserhistorie anzulegen. Dies konnte in der ersten Entwicklungsphase allerdings noch nicht bewerkstelligt werden.

Rechte und Rollen

Gemäß der Anforderungsanalyse aus der Entwurfsphase sind zwei Rollen zu definieren: Käufer und Verkäufer. Beide Rollen wurden der Tabelle *User* des Baqend- Datenbankservers über die grafische Schnittstelle hinzugefügt. Potentielle Nutzer des Onlineshops können sich über ein Formular auf der Website registrieren und erhalten dann automatisch die Rolle eines Käufers.

```
var register = function() {
  var user = $(“#username”);
  var pwd =  $(“#pwd”);
   //Kunde als Nutzer in User eintragen
  DB.User.register(user,pwd).then(function () {
     //User als Rolle Kaeufer eintragen
```

```
    DB.Role.load(11).then(function(role){
        role.addUser(DB.User.me);
        role._metadata.writeAccess();//Nach Bug entfernen
        role.save();
    });
  });
}
```

Mit der Registrierung wird zuerst ein neuer Nutzer in *User* angelegt. Danach wird die ID der Käuferrolle aus der Datenbank geladen und über das Objekt des neu registrierten Nutzers in die Tabelle *Rolle* in das Feld *User* als Element im Set hinterlegt und gespeichert.

Der Verkäufer als Betreiber des Shops hingegen kann wieder direkt in die Datenbank geschrieben werden, da dies nur einmal vollzogen werden muss. Dafür wurde ein weiterer Nutzer, *Administrator*, in *User* angelegt und der Rolle Verkäufer in der Tabelle *Rolle* zugeordnet. Die Access Control Listen (ACL) wurden für die Verkäufer auf die Rolle admin und bei den Käufern für die Rolle des Verkäufers gesetzt. So obliegt es dem Shopbetreiber die Rechte der Käufer neu zu definieren; die Rechte der Verkäufer bleiben jedoch unter Kontrolle des Datenbankadministrators.

Da wir uns aufgrund des noch recht einfachen und übersichtlichen Designs des Onlineshops gegen die Umsetzung mit einem Framework und eines Single-Page-Designs mittels Handlebars in der Planungsphase entschieden hatten, könnte sich in der konkreten Umsetzung das Problem eines effizienten Seitenzugriffs in Abhängigkeit von den Rollen ergeben. Da aber nur zwei Seiten - die Management-Seite des Administrators und die Warenkorbseite des Käufers - für Fremdzugriffe gesperrt werden müssen, bestand eine optimale Lösung darin, lediglich das entsprechende Javascript, welches die jeweilige Seite generiert, auf Rollenrechte zu prüfen.

```
//Connect
DB.ready().then(function() {
  return DB.User.me && DB.Role.load(10);
}).then(function(role) {
  if (!role || !role.hasUser(DB.User.me)) {
     throw Error('Not logged in');
  }
}).then(function() {
…
}).catch(function() {
  throw error('Fehler aufgetreten');
});
```

Zur Anmeldung des Shopbetreibers oder bereits registrierter Käufer werden ebenso wie für die Registrierung die baqend Login Methoden genutzt mit einem Verweis auf die IDs der entsprechenden Eingabefelder.

```
var login = function() {
  var usr = $("#usr");
  var passwd = $("#passwd");
```

```
  DB.User.login(usr, passwd).then(function() {
...
  });
}
```

Das Logout erfolgt über die ebenfalls vorgefertigte baqend Funktionalität nach Aufruf einer entsprechenden Funktion und Angabe der ID des Klick-Events, $('#logout').click(function() {DB.User.logout().then(function (){...}});. Zukünftig könnte man noch das Problem lösen, den Nutzer nach zu setzenden Cookies abzufragen und darüber aufzuklären, dass bei nicht gesetzten Cookies, eine Registrierung nicht möglich ist. Ebenso könnte man noch beim Schließen des Browsers oder Tabs, die Logout-Funktion auslösen, da ansonsten beim Öffnen des Browser die Session ID noch Bestand hat und eine erneute Anmeldung ausgibt, dass der Nutzer noch angemeldet sei.

Entwicklungsphase 2

Gegen Ende der zweiten Arbeitswoche wurde ersichtlich, dass die ursprüngliche Planung, die zweite Entwicklungsphase primär für Feinarbeit, Fehlersuche und das Erstellen der schriftlichen Dokumentation von Quelltext und Praktikum zu nutzen, utopisch formuliert war und nicht aufgehen würde. Funktionalitäten wie Administratoransicht, die Bedienung des Warenkorbs oder die dynamische Zuweisung der URL zur Rückverlinkung waren noch in nicht einsetzbaren Vorversionen vorhanden; weitere, wie zum Beispiel das Bewertungssystem, befanden sich noch gar nicht in der Umsetzung. Zudem trübten diverse kleinere Fehler bei Interaktion mit der Webseite das Gesamtbild.

In Anbetracht dieser Umstände wurde sich dafür entschieden, den Fokus der Bearbeitung zunächst darauf zu legen, die unfertigen Teilaufgaben zu vollenden und ihre Fehler so weit wie möglich auszumerzen, um der Qualität der Nutzung der Quantität der Nutzungsmöglichkeiten voranzustellen.

Nachfolgend soll auf die Elemente der Anwendung eingegangen werden, auf die sich vorwiegend in der dritten Arbeitswoche konzentriert wurde.

URL-Modifikation

Mit dem Wandel der Kundenansicht von einer mehrseitigen HTML-Struktur zu einer einseitigen ergab sich das Problem, dass zu Beginn nur ein einziger Link vorlag, der einen mit der Landingpage verband. Ein direkter Zugriff auf zum Beispiel eine Produkt-Detailansicht war zwar erwünscht, aber nicht möglich. Um für den Nutzer dennoch die „Mehrseitigkeit“ einer normalen Webpräsenz zu simulieren, mussten JavaScripte geschaffen werden, die die URL in der Adressleiste dynamisch zum angezeigten Inhalt anpassten und umgekehrt bei Eingabe einer Adresse in die Leiste die dargestellten Inhalte an diese anglichen.

Wird nun die Applikation in den Browser geladen oder in ihr mit den Vor- und Zurück-Buttons hin und her gesprungen, so kommt es in der JavaScript-Datei

websiteAction.js zum Aufruf der Funktion „createPage()“. In dieser wird die URL mithilfe von „window.location.href“ ausgelesen und dann mit einer Reihe von Regular Expressions verglichen. Bei einem Treffer werden die in der Bedingung aufgeführten Aufbaumaßnahmen der Seite angeleitet, wobei der String der URL gegebenenfalls so zerlegt wird, dass durch „?“ und „&“ abgeteilte Parameter wie Suchbegriff (Suchergebnisansicht) oder Produkt-ID (Produkt-Detailansicht) zur weiteren Nutzung übergeben werden können.

Umgekehrt wird bei fast jeder Änderung an der Darstellung auf dem Bildschirm mithilfe von „window.history.pushState“ ein Eintrag in der Browserhistorie angelegt, in dem die aktuelle URL hinterlegt wird, oder mit „window.history.replaceState“ ersetzt, was bei der Echtzeitsuche von großer Wichtigkeit ist, da die Historie sonst mit einer Fülle an unnötigen Einträgen belastet wird.

Warenkorb

Grundfunktionen des Warenkorbs waren bereits in der ersten Entwicklungsphase fertig gestellt worden. So bestand bereits ein Array, in das die Warenkorbprodukte vorrübergehend abgelegt worden sind. Durch die Vorarbeit war das Erstellen der Warenkorbseite keine große Herausforderung, da hier vorerst nur die Produkte aus dem Array herausgelesen und angezeigt werden mussten. Die Vorgangsweise war also vergleichbar mit den vorherigen erstellten Seiten. Auch hier wurde wieder die Mitte der Index-Seite genutzt, um ein Warenkorb-View zu erstellen.

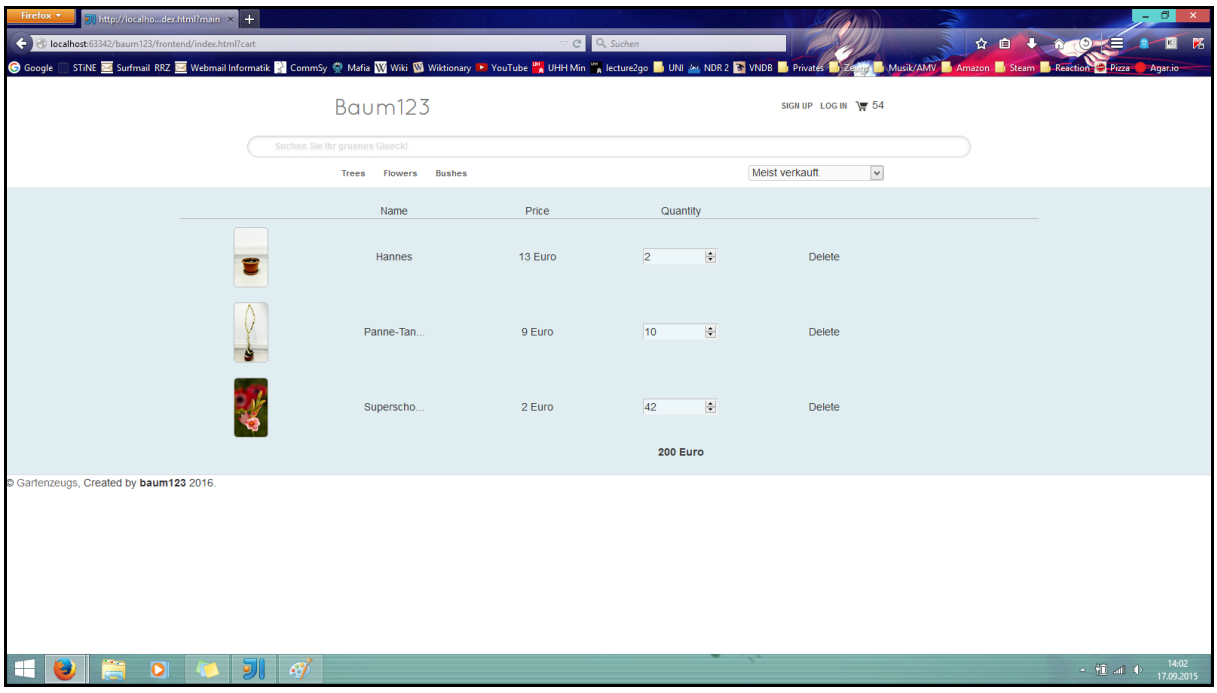
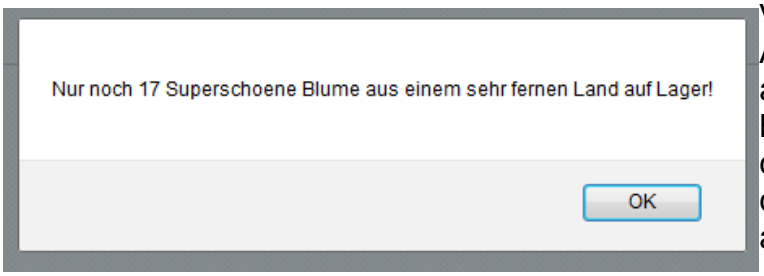


Abb7: Warenkorbansicht

Schwieriger war es, die Quantity-Felder zu änderbaren Eingabe-Feldern zu machen und die bereits vorhandene Funktionalität so zu erweitern, dass alle betroffenen Bereiche und Komponenten sich in Echtzeit anpassen.

So musste die vorher eher simple Methode zur Berechnung des Gesamtpreises stark erweitert werden. Auch das Array und die Datenbank mussten entsprechend

angesprochen werden, damit die neue Produktanzahl in allen Bereichen korrekt übertragen wird. Zuletzt gab es einige Probleme mit der Anzeige der Gesamtanzahl im Warenkorb, durch das Verändern der einzelnen Produktanzeigen kam es häufig zu fehlerhaftem Verhalten. Als „Known Bug“ gehört dabei das bis zuletzt bestehende Problem, dass bei schneller Veränderung der Produktanzahl, die Zahl am Warenkorb manchmal eine Ziffer verschluckt und dann nicht mehr mit der Anzahl im Warenkorb übereinstimmt.



Versucht man im Warenkorb eine Anzahl anzugeben, die den aktuellen Bestand übersteigt, bekommt man eine Meldung mit dem Hinweis darüber, wie viele des gewünschten Produktes noch auf Lager sind.

Ungültig sind Angaben die 0 betragen oder darunter liegen. Auch dann bekommt der Benutzer eine Hinweismeldung.

Möchte man ein Produkt aus dem Warenkorb entfernen, ist dies über den „Delete“ link möglich. Beim Klicken wird das Produkt sofort aus dem Warenkorb entfernt, die Stückzahl und der Gesamtpreis passen sich dementsprechend an. Das Objekt ist dann auch aus dem Array entfernt und auf der Datenbank wird der Lagerbestand wieder hochgezählt, während sich die Zahl der verkauften Produkte wieder vermindert.

Bewertung

Eine oberflächliche Suche im Internet zeigte, dass zahlreiche fertige Skripte für die Produktbewertung, die als Plugin eingebunden werden können, in allen Varianten vorhanden sind. Diese Skripte sind lediglich über den Funktionsaufruf verfügbar (z.B. raty() oder die Möglichkeiten, die Bootstrap bietet). Aus Gründen des besseren Verständnisses und des Übens der für die Gruppenmitglieder noch neuen Skriptsprache, wurde sich dafür entschieden, eine eigene Funktion für die Bewertung zu implementieren. Zwar diente eine fertige Lösung als Inspiration, jedoch wurde der ursprüngliche Code stark verändert, um den Bedürfnissen der Programmierer und die Maßgabe des asynchronen Designs zufriedenzustellen. Ebenso wurden die für die Bewertung eingebundenen Grafiken (grauer und gelber Stern) selbstständig erstellt.

Das Grundprinzip besteht darin, eine HTML-Listenstruktur mittels CSS und Javascript entsprechend des Maus-Events anzuzeigen oder auszublenden. Bei einem Klick-Event soll der gerade angezeigte Wert (verfügbar in $(‘#rating’)) über eine Funktion in die Datenbank geschrieben werden. Wichtig ist hier, dass die Funktion on() mit den entsprechenden Parametern verwendet wird, da dies vom asynchronen Design so gefordert ist. Ebenso ist “mouseover” und nicht “hover” zu verwenden.

An dieser Stelle ist anzumerken, dass innerhalb der dritten Woche die oberflächliche Funktionalität aufgebaut werden konnte, es jedoch nicht möglich war in dieser letzten

Phase das Bewertungssystem mit der Datenbank zu verbinden. Dies stellt noch ein Projekt für die Zukunft dar.

Produktverwaltung

Die Seite „produktaenderung.html“ ist als eine Maske zum Pflegen des Produktbestandes entworfen worden und erhält die Funktionalität aus einer eigenen Javascript-Datei. Hier konnten die Funktionen aus der Anforderungsanalyse größtenteils einfach umgesetzt werden und es gab keinen Bedarf weitere Funktionen im späteren Verlauf hinzuzufügen.

Zuerst werden alle Produkte in alphabetischer Reihenfolge eingeblendet und die Möglichkeit gegeben jeweils den Preis und die Stückzahl zu verändern. In diesem Formular kann ein Mitarbeiter bei Eingang von Waren diese beiden Kennzahlen einzupflegen. Hinter jedem dieser Datensätze ist ein Button, um eine detailliertere Ansicht des jeweiligen Produktes zu erhalten. Hier kann dieses Produkt aus der Datenbank entfernt werden und die weiteren Felder des Produktes bearbeitet werden. Außerdem ist ein Formular zum Eintragen eines neuen Produktes vorhanden. Hier können die Felder Produktname, Tags, Beschreibung, Preis, Stückzahl, Bild und Liste eingetragen werden. Diese Seite ist als Single-Page aufgebaut, bei der das Formular zum Eintragen eines Produkts und die Übersicht über Alle ausgeblendet wird, sobald die Detailübersicht eines Produktes angezeigt wird.

Diese Seite ist passwortgeschützt, damit man nicht zufällig auf diese Seite gelangen kann und die Inhalte der Datenbank manipulieren kann. Deshalb werden nicht nur die dynamischen, sondern auch die statischen Inhalte aus dem Javascript erzeugt und die HTML-Datei enthält lediglich das Grundgerüst.

Da diese Produktverwaltungsseite nicht vom Kunden einzusehen ist, war das Design im Gegensatz zur Funktionalität zuerst zweitrangig und eine optionale Aufgabe im Projekt.

Die Funktionalität der Seite war im Projekt eine der einfacheren Aufgaben. Da jedoch erst im späteren Verlauf der Login hinzugefügt wurde musste noch Änderungen an der Seite vorgenommen werden. Da die Login-Funktion die gesamte Seite einklammert, mussten die Funktionen nun global zugreifbar gemacht werden.

Fazit

Im Laufe des Praktikums hat sich unser Entwurf von einer Seite bestehend aus mehreren HTML-Dateien zu einer One-Page-Applikation entwickelt. Dadurch mussten bereits gebaute Elemente noch einmal neu entwickelt werden, um sie an die One-Page-Applikation anzupassen. Diese Probleme hätten durch eine längere und genauere Planungsphase verhindert werden können.

Die Verteilung von Aufgaben, die einzeln bearbeitet werden, hat sich meist als gut funktionierende Arbeitsweise für unsere Gruppe erwiesen. Auch das anschließende Berichten der erfolgreich bearbeiteten Elemente an die Gruppe war ständiger Bestandteil unserer Arbeitsweise. Somit ist sichergestellt, dass alle Gruppenmitglieder über das gesamte Projekt Bescheid wussten. Bestanden

hingegen Probleme, wie in späteren Stadien des Praktikums, so war jeder der Teilnehmer flexibel genug, um sich mit den Problemstellungen der anderen auseinander zu setzen und zur Lösung beizutragen.

Trotz teilweise vorhandenen Vorkenntnissen in Javascript, HTML und CSS stellte der Umfang des Neuen hin und wieder eine Herausforderung dar, Verständnisprobleme konnten jedoch dank selbstständigen Übens in Theorie und Praxis, der gegebenen Lektüre sowie der Unterstützung durch Teammitglieder und Praktikumsleiter zügig ausgeräumt werden.

Es war sehr interessant im Grundsatz funktionierende Web-Anwendung in einem kleinen Team zu entwickeln und dabei neue Erfahrungen zu sammeln. Der Umgang mit Baqend hat generell gut funktioniert wie auch die Anbindung der Applikation zur Datenbank. Für zukünftige Projekte können wir uns daher durchaus vorstellen wieder auf Baqend zurück zu greifen.