Duale Hochschule Baden-Württemberg
Mannheim

Projektdokumentation Restaurant Food Palace

Studiengang Wirtschaftsinformatik

Studienrichtung Software Engineering

Verfasser/in:	Dominik Albert, Talha Butt
Abteilung:	Softwareentwicklung
Kurs:	WWI14SEA
Studiengangsleiter:	Prof. Dr. Thomas Holey
Bearbeitungszeitraum:	11.09.2018 – 14.10.2018

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Planung

1.1 erste Überlegungen

Zu Beginn unserer Planung haben wir uns überlegt, wie wir am besten an das Projekt herantreten wollen. Da wir beide keine beziehungsweise wenig Kenntnis über HTML, CSS, JavaScript und PHP hatten, haben wir uns dazu entschieden, uns diese erst einmal unabhängig voneinander anzugucken. Ziel war es dabei, erstmal ein grundsätzliches Verständnis für die Technologien zu entwickeln. Zudem haben wir uns überlegt, wie wir zusammen an unserem Projekt arbeiten und wie wir den Code untereinander austauschen können. Für die Fallstudie in diesem Semester haben wir bereits ein Git-Repository angelegt und für die Zusammenarbeit genutzt, daher entschieden wir uns auch hier für ein Git-Repository.

1.2 Festlegen von Funktionen

Mit einem grundlegenden Verständnis von HTML, CSS, JavaScript und PHP und einer Idee, wie wir die Zusammenarbeit realisieren können, ging es nun also daran, die Webseite an sich zu planen. Aufgrund der Aufgabe eine Webseite für ein Restaurant zu entwerfen, war schnell klar, dass gewisse Funktionen auf jeden Fall realisiert werden müssen beziehungsweise gewisse Informationen zur Einsicht für die Gäste bereitgestellt werden müssen. Daher einigten wir uns vorerst auf die folgenden Seiten:

- Eine Seite zur Präsentation des Restaurants
- Eine Seite zur Kontaktaufnahme beziehungsweise zum Informieren über Kontaktinformationen

- Eine Seite mit der Speisekarte

Damit die Speisekarte auf der Webseite immer die aktuelle Speisekarte ist, war klar, dass wir dies über eine Datenbank realisieren wollen. Wir entschlossen uns, es erst einmal bei diesen Anforderungen zu belassen und die Webseite später zu erweitern.

Da wir beide noch keine Kenntnis über PHP und das Arbeiten mit Datenbanken hatten, entschlossen wir uns dazu, die Aufgaben aufzuteilen. Talha sollte sich primär mit dem Erstellen der Datenbank und PHP beschäftigen, während Dominik sich mit dem Design der Webseite und JavaScript beschäftigen sollte.

2 Bauen der Webseite (Kapitel von Dominik)

2.1 Generelle Struktur

Zum Erstellen der Webseite habe ich mich Bootstrap bedient. Über die Seite der W3School fand ich verschiedene Templates, die zur freien Verfügung stehen. Da das Ziel war, die Webseite aus mehrere Seiten aufzubauen statt eines One-Pagers, entschied ich mich für ein Template mit integrierter Navbar. Diese Navbar erlaubt es, von einer Seite auf eine andere mittels einer einfachen `<a href>` zu wechseln. Probleme machte dagegen etwas anderes: Der Versuch, die aktuelle Seite immer am linken Rand der Navbar anzeigen zu lassen. Zwar hatte ich keine Probleme damit, ein neues Item in der Navbar einzufügen, sondern damit, dass es dem Format entspricht, in dem ich es gerne haben wollte. Nach mehreren fehlgeschlagenen Versuchen entschied ich mich jedoch dafür, dies erstmal wieder zurückzustellen und die anderen Anforderungen zu erfüllen.

Das Erstellen der verschiedenen Seiten verlief ohne Probleme. Die Home-Seite besteht aus nichts weiter als HTML und CSS. Diese Seite dient lediglich der Präsentation des Restaurants. Daher gibt es hier auch nur Text und Bilder. Auf der Kontakt-Seite besteht die Möglichkeit, Kontakt mit dem Restaurant aufzunehmen. Daher ist auf dieser Seite das Impressum mit Kontaktadresse, E-Mail und Telefonnummer zu finden. Die nächste Seite ist die Menü-Seite. Über diese Seite können Gäste sich die Speisekarte und direkt Essen bestellen. Der letzte Bereich ist die Seite für das Personal. Diese ist mit einem Passwort geschützt, um Gästen keinen Zugang zu erlauben. Auf

```
.container {
    padding: 80px 120px;
}
```

Abbildung 2.1: Container Klasse

der Personalseite ist es der Küche und dem Service möglich, sich alle ausstehenden Bestellungen anzeigen zu lassen.

Bei dem Design der Webseite sind immer wieder Probleme ausgetreten, wenn es darum ging, die Seite so aussehen zu lassen, wie ich das möchte. Dies lag besonders daran, dass ich mich zu Beginn nicht genug mit CSS auseinander gesetzt habe und die meisten `<div>` Elemente lediglich als `<div class="container text-center»` definiert habe. Ich dachte, die sei lediglich dafür verantwortlich, dass der Text zentriert dargestellt wird. Die erwähnte Klasse `container` wird in Abbildung 2.1 gezeigt.

Da ich diesen Abstand mehrfach verwendet hat und ihn für angemessen hielt, habe ich mich dazu entschieden, zusätzliche Klassen für das Formatieren zu schreiben, die alle eine unterschiedliche Größe haben. So habe ich es schließlich geschafft, die Elemente auf der Seite deutlich besser anordnen zu können. Meine eigenen Container sind in 2.2 auf der nächsten Seite zu sehen.

2.2 Umsetzung der Bestellung

2.2.1 Speichern und Anzeigen der Bestellung

Der größte Teil meiner Aufgabe bestand in der Umsetzung des Bestellvorgangs. Dies habe ich gänzlich mit JavaScript realisieren können. Dafür ist in der `menue.php` Seite ein großer Teil PHP integriert. Dieser Code sorgt dafür, dass die Speisen aus der Speisentabelle unserer Datenbank abgefragt wird. Zudem wird dynamisch eine HTML Tabelle erstellt, die neben den Informationen aus der Datenbank, auch in jeder Reihe zwei Buttons erstellt. Diese Button sind für mich das entscheidende, denn sie dienen zum Hinzufügen oder Abziehen von Speisen zu der Bestellung.

```
.containerWObot {
    padding-top: 80px;
    padding-left: 120px;
    padding-right: 120px;
}
.containerWOtop {
      padding-bottom: 80px;
      padding-left: 120px;
      padding-right: 120px;
}
.containerWObottop {
    padding-left: 120px;
    padding-right: 120px;
}
.smallcontainer {
    padding: 40px 120px;
}
```

Abbildung 2.2: Meine Container Klassen

Um die Button gezielt ansprechen zu können, wird den Button dynamisch eine ID zugewiesen. Diese ID besteht aus den Schlagwörtern "plusöder "minusünd der ID der Spalte. Zusätzlich besitzen alle Button die Klasse `orderbutton`, sodass ich über einen jQuery-Listener auf einen Klick reagieren kann. Diese Funktion wird immer dann aufgerufen, wenn ein Button dieser Klasse geklickt wird. Die gesamte Funktion ist in Abbildung 2.3 auf der nächsten Seite dargestellt.

Das Speichern der endgültigen Bestellung hat mir große Probleme bereitet. Ich hatte geplant, mithilfe einer weiteren Datenbankabfrage dynamisch ein Array erstellen zu können, in dem alle Speisen gespeichert werden. Dies ist mir allerdings nicht gelungen. Schließlich habe ich ein eigenes Array erstellt, das die Länge 34 hat und überall mit 0 initialisiert ist. Die Länge 34 ergibt sich daraus, dass wir insgesamt 34 Speisen in unserer Datenbank gespeichert haben. Dieses Array `$bestellung` speichert somit die gesamte Bestellung über den Index. Das Bestellen einer Speise sorgt für eine Erhöhung des entsprechenden Indexes im Array. Die Funktion bedient sich dabei der ID der Button. Sie erkennt so, ob ein "plusöder "minus"Button geklickt wurde und welche Nummer der Button hat, sprich welche Speise gemeint ist. Zudem wird gleichzeitig aus diesem Array ein String erstellt, der am Ende der Seite angezeigt wird. So lässt

```
167  $('button.orderbutton').click(function() {
168    var $attr = $(this).attr('id');
169    var $ar = $attr.split("-");
170    var $type = $ar[0];
171    var $id = $ar[1];
172
173  if ($type === "plus") {
174    add($id);
175  }
176  else {
177    sub($id);
178  }
179
180  var el = document.getElementById("bestellung");
181  $bes = new Array();
182  for (var $i = 0; $i < 34; $i++) {
183    if ($bestellung[$i] != 0) {
184      var $a = $bestellung[$i] + "x mal Speise " + ($i+1) +"<br>";
185      $bes.push($a);
186    }
187  }
188  $final = $bes.toString();
189  while ($final.indexOf(",") !== -1) {
190    $final = $final.replace(",", "");
191  }
192
193  el.innerHTML = $final;
194  checkStatus();
195  checkPreis();
196  });
```

Abbildung 2.3: Orderbutton Listener

sich für den Gast immer sehen, wie die momentane Bestellung aussieht und wie viel es kostet. Zur Berechnung des Preises wird ebenso ein Array genutzt, das ich mit den entsprechenden Preisen initialisiert habe. Auch hier werden die Preise leider nicht aktuell abgefragt, sondern stehen direkt in der Datei.

2.2.2 Verbindung mit der Datenbank

Das Ziel unserer Seite für das Personal war es, alle aktuellen Bestellungen sehen zu können, ohne die Seite immer wieder aktualisieren zu müssen. So weiß die Küche und das Personal immer, was noch aussteht. Dafür haben wir eine zweite Tabelle in der Datenbank angelegt. Meine nächste Aufgabe bestand also darin, die Bestellung nicht

nur anzuzeigen, sondern beim Klicken des Buttons `Bestellen` die Bestellung zudem in die Datenbank zu schreiben. Dafür habe ich zuerst noch ein Textfeld eingebaut, in dem die Gäste ihre Tischnummer angeben müssen. So weiß das Personal automatisch, an welchen Tisch die Speisen geliefert werden müssen und die Küche weiß, welche Speisen möglichst zeitgleich bereit sein müssen. Das Einfügen in die Datenbank machte dagegen mehrere Probleme. Ich entschied mich, für jede bestellte Speise eine eigene Anfrage an die Datenbank zu starten, da sich dies mit dem Array der Bestellung gut umsetzen ließ. Das Klicken auf den Button löst also eine Funktion auf, die das gesamte Array durchgeht. Überall dort, wo der Wert nicht Null ist, wird dann eine Anfrage an die Datenbank gestartet.

Nach einer schnellen Suche im Internet fand ich heraus, dass ich dies mithilfe von jQuery und PHP realisieren musste. Die Idee der Funktion in JavaScript ist es, die Daten mithilfe der `POST` Methode von jQuery and eine eigene PHP Datei zu übergeben. Von dort werden sie dann mithilfe eines SQL Befehls in die Datenbank geschrieben. Mein Problem war jedoch, dass ich einen Syntaxfehler bei der JavaScript Funktion hatte und diesen nicht bemerkt habe. Gemerkt habe ich es lediglich daran, dass in der Datenbank eine neue Reihe erschien, die jedoch keine Daten enthielt. Somit war klar, dass der eigentliche `INSERT` Befehl funktionierte, es jedoch zu einem Problem beim Übertragen der Daten kam. Da ich mich mit PHP nicht viel mehr beschäftigt hatte, ging ich erstmal davon aus, der Fehler müsse in der `insert_bestellung.php` Datei liegen. Nach etwa 90 Minuten erfolgloser Suche und googlen bemerkte ich endlich, dass der Fehler nicht in der PHP Datei lag, sondern in der JavaScript Funktion. Da ich als Editor Visual Studio Code verwendet habe, welches keine Syntaxprüfung wie gewöhnliche Entwicklungsumgebungen hat, fiel mit dieser Fehler nicht früher auf. Der falsche Code ist in Abbildung 2.4 auf der nächsten Seite dargestellt, der richtige ist zum Vergleich in Abbildung 2.5 auf der nächsten Seite zu sehen. Wie zu erkennen ist, lag der Fehler lediglich darin, dass ich eine Klammer falsch gesetzt habe. In der falschen Funktion wird der `POST` Befehl schon direkt nach dem Aufruf der PHP Funktion beendet, jedoch bevor ich die Daten mitgegeben habe. Daher resultierten die leeren Zeilen in der Datenbank.

Die gleiche Logik, wie ich sie hier verwendet habe, habe ich danach auch genutzt, um die Reservierung eines Tisches über die `kontakt.php` Seite zu realisieren. Leider

```
function insert($id, $anz, $tisch) {
  $.post("insert_bestellung.php"), {
    speise: $id,
    anzahl: $anz,
    tischID: $tisch
  };
}
```

Abbildung 2.4: Falsche Insert Funktion

```
function insert($id, $anz, $tisch) {
  $.post("insert_bestellung.php", {
    speise: $id,
    anzahl: $anz,
    tischID: $tisch
  });
}
```

Abbildung 2.5: Richtige Insert Funktion

haben wir es zeitlich nicht mehr geschafft, einen zweiten Reiter im Personalbereich zu implementieren, der die aktuellen Reservierungen anzeigt.

2.3 Zeitaufwand

Mein gesamter Zeitaufwand ist in Tabelle 2.1 aufgeführt.

Aktivität	**Zeitaufwand**
`Installation von Xampp`	1 Stunde
`Einarbeitung in Bootstrap und Javascript`	15 Stunden
`Arbeit an der menue.php Seite`	5 Stunden
`Arbeit an den anderen Seiten`	7 Stunden
`Schreiben der Dokumentation`	5 Stunden

Tabelle 2.1: Zeitaufwand von Dominik

3 Datenbank und PHP (Kapitel von Talha)

3.1 Einarbeitung in PHP und PHPMyAdmin

Um sich mit der Thematik der Webprogrammierung besser auseinandersetzen zu können, wurde zunächst eine Einarbeitungsphase durchgeführt. Dazu wurden Internet-Rescourcen wie Google, W3schools, Stackoverflow und Youtube genutzt. Ein Einführungsvideo zu PHP von der Website FreeCodeCamp half dabei sich Schritt für Schritt in die Systematik, Syntax und Funktionalität der Sprache einzuarbeiten. Eine wichtige Rolle bei dem Erlernen der Sprache war die praktische Anwendung und das Ausprobieren in der Entwicklungsumgebung. Dabei wurde stets mit einem „Hello World“-Beispiel begonnen und der Komplexitäts-Grad der Programmieraufgabe Schritt für Schritt erhöht. So wurde trotz des Beginns als absoluter Laie eine gewisse Wissensbasis und ein Verständnis aufgebaut, welches für die Entwicklung einer Website notwendig ist. Die Einarbeitungsphase allein in HTML und PHP nahm aufgrund fehlender Vorkenntnisse 2 Wochen a 12 Stunden in Anspruch. Gleiches Prozedere fand in gekürzter Fassung auch für die Einarbeitung in „PHPMyAdmin“ statt. In gekürzter Fall deshalb, da zwar MySQL neu war, aber bereits Wissen über Datenbanken an sich vorhanden war(aufgrund eines Unternehmensprojekts).

Die erworbenen Kenntnisse kamen beispielsweise auf der `menue.php` Seite zum Einsatz. Dort ging es darum, die Speisekarte mithilfe von einer in PHP implementierten Verbindung mit der Datenbank, per SQL Abfrage abzurufen. Diese Abfrage ist in 3.1 auf der nächsten Seite dargestellt. Dabei wurden Vorspeisen, Hauptspeisen, Desserts

```php
<?php
header("Content-Type:text/html;charset=utf-8");
   $servername = "localhost";
  $dbname    = "website_food_palace";
  $username = "root";
  $password = "";
   // Create connection
  $conn = mysqli_connect($servername, $username, $password, $dbname);
   // Check connection
  if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
    }
  //setzt charset auf utf8
  mysqli_set_charset($conn, "utf8");
  // Create query:
  //Vorspeise
  $sql = "SELECT * FROM speisentabelle WHERE ID BETWEEN 1 AND 4;";
  //Hauptspeisen
  $sql2 = "SELECT * FROM speisentabelle WHERE ID BETWEEN 5 AND 20;";
  //Dessert
  $sql3 = "SELECT * FROM speisentabelle WHERE ID BETWEEN 21 AND 25;";
  //Getränke
  $sql4 = "SELECT * FROM speisentabelle WHERE ID BETWEEN 26 AND 34;";
```

Abbildung 3.1: SQL Abfrage für das Menü

und Getränke einzeln abgefragt, um sie dem Website-Besucher einfach und übersichtlich anzuzeigen.

Eine Hürde bezieungsweise ein Fehler welcher zunächst beim Aufsetzen der Datenbank unterlief, war das Festlegen der *Collation*. Da beim Erstellen der ersten Tabelle keine Angabe in diesem Feld gemacht wurde, stellte der PHPMyAdmin die Default-Angabe `swe7_swedish_ci` ein. Dies führte dazu, dass die Umlaute der deutschen Sprache nicht durch den Unicode angezeigt wurden. Diese Tatsache fiel bereits früh nach Erstellung und Abfrage der ersten Tabelle `speisetabelle` auf und wurde zunächst durch Ände-

rung in PHPMyAdmin auf `utf8_general_ci` und hinzufügen einer Header-Angabe in `charset=utf-8` in PHP bereinigt.

3.2 AJAX-Implementierung im Personalbereich mit `load_orders.php`

AJAX wurde im Personalbereich implementiert. Im Personalbereich werden die im Menü aufgegebenen Bestellungen für das Personal angezeigt. So ist das hypothetische Küchenpersonal stets auf dem neusten Stand der Bestellungen und kann diese nach Fertigstellung auch aus der Bestellungsliste löschen.

Eine AJAX-Funktionalität bietet sich hier an, da AJAX eine automatisierte Aktualisierung der Seite ermöglicht, welche im Fall des Küchenpersonals das mühselige Neu-Laden des Personalbereichs erspart.

Mithilfe der Funktion `setInterval()` wird die Seite in regelmäßigen Zeitabschnitten mit einer `load()` Funktion, welche auf die `load_orders.php` Datei verweist, asynchron aufgerufen werden. Die `load_orders.php` Datei beinhaltet die SQL-Abfrage der Bestellungen, sowie die Erstellung der Tabelle mit HTML in `echo`. Abbildung 3.2 auf der nächsten Seite zeigt die `setInterval()` und die `load()` Funktionen, Abbildung 3.3 auf der nächsten Seite zeigt die Erstellung der Tabelle in der `load_orders.php` Datei.

Die Datei beinhaltet auch die Implementierung eines `fertig-button`, welcher dem User die Möglichkeit bietet, Einträge aus der Datenbank zu löschen. Der `fertig-button` wurde logisch als Javascript `$('button.fertigbutton').click(function()` Funktion umgesetzt. Wird der Knopf gedrückt, folgt ein SQL DELETE-Befehl, welcher die Zeile der button-id aus der Datenbank löscht. Da diese Funktionalität sich ebenfalls in der `load_orders.php` befindet, werden die gelöschten Einträge nach Zeitintervall in der aktualisierten Tabelle auch nicht mehr angezeigt. Dies ermöglicht dem hypothetischen Küchenpersonal nach Fertigstellung der gewünschten Bestellung, diese aus der Tabelle zu löschen.

```
67      timer();
68
69      function timer() {
70            setInterval( function(){
71              load();
72            }, 7000);
73         }
74
75     function load() {
76       $orderCount = $orderCount+3;
77       $('#orders').load("load_orders.php",{ });
78     }
```

Abbildung 3.2: Timer und Load Funktion

```
$result = mysqli_query($conn, $sql);

echo '<table border="1">';
echo "<tr>
      <th>Bestell-ID</th>
      <th>Bestellung</th>
      <th>Anzahl</th>
      <th>Tisch</th>
    </tr>";

  foreach ($result as $zeile)
  {
    echo "<tr>";
    echo "<td>". $zeile['id'] . "</td>";
    echo "<td>". $zeile['speise'] . "</td>";
    echo "<td>". $zeile['Anzahl']. " </td>";
    echo "<td>". $zeile['tisch']. " </td>";
    echo "<td><button id=".$zeile['id']." class='fertigbutton'>fertig</button></td>";
    echo "</tr>";
```

Abbildung 3.3: Erstellen der Tabelle

3.3 Installation der Entwicklungsumgebung und Zeiteinteilung

Die Installation von Xampp erfolgte problemlos. Apache konnte zu Beginn nicht gestartet werden. Nach untersuchen und ändern des Ports, obwohl Port 80 und 443 eigentlich frei waren, klappte es immer noch nicht. Es wurde nach kurzer Zeit festgestellt, dass Xampp nicht als Admin gestartet wurde, was dazu führte, dass Apache und MySQL nicht gestartet werden konnten.

Die gesamte Zeiteinteilung ist in Tabelle 3.1 aufgeführt.

Aktivität	**Zeitaufwand**
`Installation von Xampp`	1 Stunde
`Einarbeitung in PHP und HTML`	3 Wochen a 6 Stunden
`Arbeit an der Website neben Einarbeitung`	4 Wochen a 3 Stunden
`Schreiben der Dokumentation`	4 Tage a 1 Stunde

Tabelle 3.1: Zeitaufwand von Talha