

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe. Diese Versicherung bezieht sich sowohl auf Textinhalte sowie alle enthaltenen Abbildungen, Skizzen und Tabellen. Die Arbeit wurde in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde zur Erlangung eines akademischen [1] Grades vorgelegt.[2]

Karlsruhe, den 04. Juli 2014

Matthias Misior

Anerkennungen

Zunächst möchte ich mich an dieser Stelle bei meinem Betreuer Maximilian Liesegang bedanken und meine Anerkennung ihm gegenüber zum Ausdruck bringen. Seine Fachkompetenz und sein Enthusiasmus haben mich über die gesamte Projektdauer begleitet und waren stets eine große Hilfe und eine wichtige Motivationsquelle für mich.

Des weiteren gilt mein persönlicher dank Frau Professorin Laubenheimer, dessen ständige Erreichbarkeit und schneller E-Mail Verkehr es mir erst ermöglicht haben, diese Arbeit in dieser Form rechtzeitig zum Abgabetermin fertigzustellen.

Zu Guter Letzt gilt mein persönlicher Dank dem gesamten esentri-Team. Das Team hat mich inspiriert, andere Wege zu gehen und ich habe dadurch viel über die Arbeit sowie über mich selbst gelernt. Die Fachkompetenz und Ratschläge des Teams waren dabei stets eine große Hilfe. Ohne die Unterstützung des esentri-Teams wäre diese Arbeit so nicht möglich gewesen.

Ich hoffe ich konnte mit diesen einfachen Worten allen Beteiligten meine Dankbarkeit zum Ausdruck bringen.

Zusammenfassung

Die Wirtschaftsjuvenen Karlsruhe(WJ) mit rund 160 aktiven Mitgliedern aus allen Bereichen der Wirtschaft veranstalten alle zwei Jahre ein sogenanntes „Staffeleessen“. Das Staffeleessen ist eine traditionelle Veranstaltung, bei der jeder Teilnehmer einen von drei Gängen zubereiten soll. Jeder Teilnehmer kann wählen, ob er die Vorspeise, den Hauptgang oder das Dessert zubereiten möchte. Bei den anderen Gängen ist er Gast. Die Teilnehmer jedes Ganges werden neu zusammengesetzt, so dass sich die selben Personen im Laufe des Abends nicht zweimal treffen können, Paare bleiben jedoch immer zusammen. Bei dieser Veranstaltung geht es darum sich untereinander besser kennenzulernen, neue Kontakte zu anderen Branchen zu knüpfen und ältere Bekanntschaften zu pflegen.

Bisher wurden die Teilnehmer händisch in einfache Excel Tabellen eingetragen und die Zuordnung der unterschiedlichen Gänge wurde per Hand durchgeführt. Nun hat die Nova GmbH, eine Internet Agentur in Karlsruhe, den Auftrag erhalten eine Webanwendung zu entwickeln, die das Organisieren dieses Staffeleessens vereinfachen soll.

Die vorliegende Bachelor-These befasst sich mit dem Entwurf und der Implementierung eines Prototypen der als Webanwendung sowohl für die Organisatoren als auch für die Teilnehmer des Staffeleessens gedacht ist. Schwerpunkte dieser Arbeit liegen auf dem Responsiv Webdesign wie auch auf der Serverseitigen Implementierung der Webanwendung.

Abstract

The Wirtschaftsjuvenen Karlsruhe(WJ) with around 160 active members from all branches of economics organize every two years a socalled „Staffeleessen“. Staffeleessen is an traditional event where every participant has to prepare one of three dishes. Every participant can choose if he wants to prepare the appetizer, main course or dessert and is guest for the other courses. The participants are reset for every dish so that the same people never meet twice during the duration of the evening, couples always stay together. This event is about getting to know each other better, to make new contacts with other branches or maintain older acquaintances.

Up until now, the participants were entered manually in simple Excel tables and even the dish assignment has been made by hand. The Nova GmbH, an web agency located in Karlsruhe, received the task to develop an webapplication which would make it easier to organize this event.

This thesis deals with the design and implementation of a prototype that serves as an web-application for the organizers and participants. Focus of this work lies on the responsive web design as well as the server side implementation of the web-application.

Inhaltsverzeichnis

1	Aufgabenbeschreibung	6
1.1	Ablauf des Staffelessen	6
1.2	Aufgaben der Webanwendung	6
1.2.1	Anmeldung der Teilnehmer	6
1.2.2	Zuordnung der Teilnehmer	7
1.3	Ziel der Webanwendung	8
2	Grundlagen	9
2.1	Begriffsdefinition	9
2.2	Technologien	10
2.2.1	PHP	10
2.2.2	HTML	12
2.2.3	MySQL	14
2.2.4	jQuery	15
2.2.5	Bootstrap 3.0	18
2.3	SCRUM	20
2.3.1	SCRUM-Rollen	20
2.3.2	Der Ablauf eines SCRUM-Prozesses	21
3	Anforderungsanalyse	23
3.1	Fachliche Anforderungen	24
3.2	Nichtfachliche Anforderungen	28
3.3	Use Cases	29
3.3.1	Beschreibung der Use Cases	29
4	Konzept	33
4.1	Beschreibung der Teilnehmer-Zuordnung	34
4.1.1	Zuordnungskriterien	34
4.1.2	Problemstellung Teilnehmer-Zuordnung	34
4.1.3	Konzeptioneller Entwurf der Teilnehmer-Zuordnung	37
4.1.4	Prozess: Teilnehmer-Verteilung	38
4.1.5	Prozess: Gerichte-Verteilung	40
4.1.6	Prozess: Gäste hinzufügen	41
4.1.7	Prozess: Vegetarier-Verteilung	42
4.1.8	Prozess: Teilnehmer-Zeitüberwachung	45
4.2	Beschreibung der Teilnehmer-Anmeldung	47
4.2.1	Prozess: Teilnehmer-Informationen erfassen	48
4.2.2	Prozess: Geodaten-Ermittlung	50
4.2.3	Prozess: Fahrdistanz-Berechnung	50
4.2.4	Anmerkungen zur Teilnehmer-Anmeldung	52
4.3	Beschreibung Instruktionen drucken	54
4.3.1	Aufbau der Instruktionen	54
4.3.2	Anmerkungen zum Use Case Instruktionen drucken	55

4.4	Teilnehmer-Editierung	55
4.4.1	Darstellung der Teilnehmer-Editierung	55
4.4.2	Manipulation der Datenbank	57
4.4.3	Änderungen der Datenbank festhalten und Ausgeben	58
5	Screendesign	59
5.1	Allgemeines Seiten-Layout	59
5.1.1	Strukturieren der Unterseiten	59
5.2	Die Sitemap	61
5.2.1	Use Cases und Anforderungen in der Sitemap	62
5.2.2	Startseite	63
5.2.3	Administrator Login	63
5.2.4	Admin Funktion 1: Gästeliste generieren	63
5.2.5	Admin Funktion 2: Gästeliste ansehen	64
5.2.6	Admin Funktion 3: Teilnehmer-Editierung	66
5.3	Responsive-Design	67
5.3.1	Grundlagen des Responsive-Design	68
6	Implementierung	69
6.1	System Architektur	70
6.2	Implementierung des Konzeptes	71
6.2.1	Struktur des Skriptes	72
6.2.2	Die Klassen GruppenZuweiser, Gruppe und Gast	73
6.3	Datenbank Struktur	76
6.3.1	Datenbank Tabellen	76
7	Fazit	80
8	Ausblick	81
8.1	Features	81
8.2	Architektur	82

1 Aufgabenbeschreibung

Diese Thesis befasst sich mit dem Entwurf und der Implementierung einer Webanwendung, welche die Abläufe und Prozesse beim Organisieren des Staffelessens der Wirtschaftsjunioren Karlsruhe effizienter gestalten soll. Es soll eine Webanwendung entstehen, die dem Organisator die komplexen Aufgaben der Teilnehmer-Zuordnung abnimmt und ihm dabei dennoch menschlichen Handlungsspielraum lässt. Neben dem Entwurf und Implementierung der Webanwendung liegt der Schwerpunkt dieser Arbeit auf der Erstellung eines benutzerfreundlichen Seitenlayouts und einer intuitiven Seitennavigation für den Organisator und die Teilnehmer des Staffelessens. Im folgendem Abschnitt wird auf den Ablauf des Staffelessens detaillierter eingegangen.

1.1 Ablauf des Staffelessens

An einem Staffelessen nehmen in der Regel 50 bis 100 Personen teil. Jeder dieser Personen nimmt ein drei Gänge Menü mit Vorspeise, Hauptgang und Dessert zu sich. Ein Teilnehmer entscheidet während der Anmeldung, die im Vorfeld des Staffelessens stattfindet, welchen der drei Gänge er zubereitet. Der Teilnehmer ist einerseits ein Gastgeber für seinen eigenen Gang, sowie ein Gast bei den zwei übrigen Gängen. Der Reiz der Veranstaltung liegt für die Teilnehmer darin, dass sie weder vorab wissen, bei welchem Gastgeber sie welches Gericht zu sich nehmen, noch, wen sie als Gast erwarten dürfen. Dadurch wird ein aktives *Networking* zwischen den teilnehmenden Personen gefördert.

Jeder Gang wird als Runde oder Gericht bezeichnet. Nach jeder Runde erhalten die Teilnehmer neue Instruktionen in denen ihr nächster Gastgeber für die folgende Runde mit Anschrift vermerkt ist. Die neue Runde setzt sich aus anderen Teilnehmern zusammen. Dadurch soll verhindert werden, dass sich zwei oder mehr Teilnehmer im Verlauf der drei Runden doppelt begegnen. Vor der ersten Runde treffen sich alle Teilnehmer an einem vereinbarten Treffpunkt, wo ihnen mitgeteilt wird, wer ihr Gastgeber für die Vorspeise ist. Zum Schluss gibt es dann ein gemeinsames Abschlusstreffen mit allen Teilnehmern.

1.2 Aufgaben der Webanwendung

Die Anwendung soll eine Softwarelösung für die Organisation darstellen. Dafür muss die bisherige Vorgehensweise bei der Planung des Staffelessens berücksichtigt werden. Das Staffelessen teilt sich insgesamt in zwei Aufgabenbereiche, die wiederum aus mehreren Prozessen und Prozessschritten bestehen. Zum einen die Abwicklung der Teilnehmer-Anmeldung und zum anderen das Zuordnen der Gäste zu den Gastgebern durch den Organisator des Staffelessens. Diese beiden Aufgabenbereiche müssen nun softwaretechnisch mit dieser Webanwendung umgesetzt werden.

1.2.1 Anmeldung der Teilnehmer

Dieser Teil der Anwendung wird vom Teilnehmer genutzt. Auf der Startseite soll dem Teilnehmer ein Anmeldeformular zur Verfügung gestellt werden, in das er seine Informationen wie Name, Adresse und Kontaktdaten einträgt. Zusätzlich sollen dem Teilneh-

mer Angaben zum bisherigen Verlauf des Staffelessens angezeigt werden. Die Angaben sollen Informationen wie bisher angemeldete Gerichte oder die Anzahl aller bisherigen Teilnehmer enthalten. Anhand dieser Angaben kann sich der noch zu registrierende Teilnehmer orientieren. Diese Suggestionen dienen der Optimierung und sollen dabei helfen die gleichmäßige Verteilung aller Gänge sicherzustellen.

Anmeldungen oder Registrierungen von Benutzern folgen immer dem selben Schema. Zu Beginn werden die Daten des Benutzers erfasst. Anschließend müssen die Daten validiert werden, da die Datenqualität von großer Wichtigkeit ist und um die Anwendung vor Hacker-Angriffen zu schützen. Zum Schluss müssen die Daten in einer Datenbank oder einem ähnlichen Speichermedium hinterlegt werden damit die gewünschten Informationen für die Anwendung jederzeit zugänglich sind.

1.2.2 Zuordnung der Teilnehmer

Die wichtigste Aufgabe der Anwendung ist das Zuordnen und Gruppieren der Teilnehmer. Dafür übernimmt der Organisator des Staffelessens die Rolle des Administrators. Er erhält eine Übersicht von der aus er die Zuordnung starten kann. Die Übersicht enthält die selben Informationen wie die Anmeldung und soll dem Administrator Aufschluss über den Status des Staffelessens geben. Durch sie kann der Administrator zum Beispiel ablesen, ob sich bereits genügend Teilnehmer registriert haben, da die Zuordnung aus mathematischen Gründen erst ab einer Mindestanzahl von X Teilnehmern sinnvoll funktionieren kann. Anhand dieser Daten kann der Administrator anschließend entscheiden, ob er die Zuordnung startet oder nicht.

Die Aufgabe der Zuordnung ist es, eine Gästeliste zu erzeugen (Abb.1) anhand derer der Administrator erkennt wie sich die Tische der Gastgeber und ihrer Gäste zusammenstellen. *Tisch* meint dabei eine Personengruppe während eines Ganges bei einem Gastgeber. Tische werden nach ihrem Gang sortiert und als einfache Tabelle dargestellt. Die Tische sollen die Informationen der Teilnehmer visuell wiedergeben. Es soll sichtbar werden, welcher Teilnehmer z.B Vegetarier ist oder ob ein Gastgeber ein vegetarisches Essen zubereitet. Hinzukommen noch die Gäste die einen Partner haben und dadurch als Paar am Staffelessen teilnehmen. Sie müssen ebenfalls so angezeigt werden das man erkennen kann, wer einen Partner hat und wer nicht.

Als letzten Schritt soll der Administrator die Gästeliste speichern und die Instruktionen ausdrucken können. Die Instruktionen für die Teilnehmer werden anhand der Gästeliste erzeugt, so das der Organisator die Ausdrücke lediglich in Briefumschläge verpacken muss.
























































Vorspeise	
SÜTTERLIN Andreas	    
RAHÄUSER David	      
KOCH Stefan	     
STRECKER Arno	    
TRIEBEL Volkmar	   
WALZER Sandra	   
ROSEN-KÖNIG Yvonne	     
SCRABACK Tino	     
GÜNTHER Dirk	     
MURAWSKI Alexander	     

Abbildung 1: Beispiel einer Gästeliste für alle Vorspeisen.

1.3 Ziel der Webanwendung

Die Anwendung soll den Verwaltungs- und Organisationsaufwand für den Organisator reduzieren. Im speziellen soll das zeitaufwändige und komplizierte zuordnen der Teilnehmer durch die Anwendung abgewickelt werden. Die Tätigkeiten des Organisators würden sich dann lediglich auf das prüfen der Zuordnungen oder das drucken der Instruktionen beschränken. Der Teilnehmer wird in der Lage sein seine Anmeldung Online, egal ob mobil oder lokal, ohne Zutun des Organisators bewerkstelligen zu können und ist somit flexibel und unabhängig vom Organisator. Die Informationen für die Benutzer (Organisator u. Teilnehmer) werden Online zur Verfügung gestellt. Somit erhält jeder Benutzer Einsicht in den momentanen Verlauf der Organisation des Staffelessens. In Zukunft soll die gesamte Organisation über die Webanwendung abgewickelt werden.

2 Grundlagen

Das Projekt wurde nach den Standards und den Verfahrensweise der nova GmbH entwickelt. Sowohl Entwicklungs- als auch Projektplanung wurden nach einem angepassten Scrumverfahren durchgeführt, da ein Scrumverfahren optimal zur Entwicklung eines Prototypen geeignet ist. Um das Projekt besser vor Augen zuführen, werden in diesem Kapitel die grundlegenden Begriffe, welche zum besseren Verständnis der Webanwendung gedacht sind, erklärt. Zusätzlich werden in diesem Kapitel die verwendeten Technologien aufgeführt.

2.1 Begriffsdefinition

Die Definitionen die in diesem Abschnitt aufgeführt und erläutert werden sind ausschließlich aus dem Realweltmodell übernommen, da bisher keine anderen Herleitungen möglich sind. Sie werden bei der Modellierung der Architektur verwendet um Klassen und andere Kernstücke für die Anwendung zu definieren. Da die Definitionen im Verlauf dieser Arbeit öfters aufgeführt werden und sie ein wichtiger Teil des Entwicklungsprozesses sind, ist es erforderlich ihre Bedeutung zu kennen.

Der **Organisator** ist der Administrator der Anwendung und kann auch Teilnehmer beim Staffeessen sein. Als Administrator erhält er Sonderrechte, die es ihm erlauben, zusätzliche Operationen innerhalb der Anwendung durchzuführen. So kann er z.B die Teilnehmer-Zuordnung starten, Teilnehmer editieren und die Instruktionen einsehen und drucken.

Der **Teilnehmer** stellt den Benutzer innerhalb der Webanwendung und somit eine Person, die am Staffeessen teilnimmt, dar. Durch das Anmelden wird der Benutzer zu einem Teilnehmer und erhält danach einen Datensatz in der Anwendung. Jeder Teilnehmer existiert nur einmal in der gesamten Anwendung.

Der **Gastgeber** ist eine spezielle Rolle, die der Teilnehmer einnimmt. Im Idealfall sollte jeder Teilnehmer ebenfalls ein Gastgeber sein, was aber nicht sein muss. Gastgeber müssen anhand ihres Gerichtes unterschieden werden. Ein Gastgeber darf immer nur ein Gericht zubereiten. Der Organisator kann Einfluss darauf nehmen, welches Gericht der Gastgeber zubereitet oder ob der Teilnehmer überhaupt als Gastgeber am Staffeessen teilnimmt, falls dies erforderlich sein sollte.

Ein **Gast** ist eine weitere spezielle Rolle die der Teilnehmer einnehmen kann. Als Gast werden hier Teilnehmer bezeichnet die zu einem Gericht eines Gastgebers zugeordnet wurden. Bei diesem Projekt ist es vorgesehen, das ein Teilnehmer genau einmal in die Rolle eines Gastgebers und zweimal in die Rolle eines Gastes schlüpft. Sollte es zu dem Sonderfall kommen, das ein Teilnehmer kein Gericht vorbereitet, zählt dieser Teilnehmer nur als Gast.

Ein **Paar** oder Pärchen ist eine spezielle Form eines Teilnehmers. Ein Paar ist eine Menge von genau zwei Benutzern die sich als Paar bei der Anmeldung registriert haben. In der Webanwendung zählt ein Paar als nur ein Teilnehmer da die Benutzer nie getrennt

werden. Da Gastgeber und Gäste lediglich nur Rollen eines Teilnehmers sind gibt es auch Gastgeberpaare und Gästepaare.

Ein **Tisch** ist eine Menge die aus mehreren Teilnehmern besteht. Einer dieser Teilnehmer ist der Gastgeber des Tisches und die anderen Teilnehmer sind lediglich die Gäste. Die Tische werden untereinander anhand ihres Gastgebers unterschieden und lassen sich ebenfalls anhand des Gerichtes kategorisieren. Erzeugt und befüllt werden die Tische während der Laufenden Zuordnung der Teilnehmer.

Die **Gästeliste** ist eine Menge, die als Elemente die zusammengesetzten Tische besitzt. Sie ist das Resultat der Teilnehmer-Zuordnung und darf lediglich vom Organisator eingesehen werden. Die Gästeliste wird Online gespeichert sodass der Organisator jederzeit die Informationen Abfragen kann. Aus der Gästeliste werden später die Instruktionen für die Teilnehmer generiert.

Die **Instruktionen** haben die Aufgabe, einen Teilnehmer Informationen über seinen folgenden Gang zu vermitteln. Die Instruktionen werden dabei durch den Organisator an die Gastgeber verteilt. Der Gastgeber verteilt die Instruktionen an alle seine Gäste (sich selbst eingeschlossen), sodass alle Teilnehmer ihr nächstes Ziel erfahren. Dieses *Networking* ist der besondere Reiz des Staffelessens, da jedes Gericht eine Etappe darstellt an der weitere Instruktionen auf den Teilnehmer warten.

2.2 Technologien

Da es sich bei diesem Projekt um die Entwicklung einer Webanwendung handelt werden die hierfür typischen Technologien verwendet. Das bedeutet das sowohl HTML 5 als auch PHP benutzt wird, um ein dynamisches Seitenlayout der Webanwendung zu gestalten. Da die Anwendung Daten der Teilnehmer speichert, muss ebenfalls eine Datenbank angebunden werden, über die Daten gespeichert oder wiedergegeben werden. Um weitere dynamische Features zu realisieren wird das jQuery Framework genutzt, das auf Javascript aufbaut. Aus Gründen der Entwicklungseffizienz und um die Webanwendung Benutzerfreundlicher zu gestalten, wird das Framework Bootstrap 3.0 verwendet, das eine Bibliothek von vorgefertigten CSS und jQuery Komponenten zu Verfügung stellt. Die einzelnen Technologien werden im folgendem im Detail erläutert.

2.2.1 PHP

PHP (Hypertext Preprozessor, ursprünglich Personal Home Page) ist eine serverseitig ausgeführte Skriptsprache, welche speziell für das Web entwickelt wurde. Durch das einbetten von PHP-Code in eine statische HTML-Seite ist diese in der Lage, dynamische Inhalte anzuzeigen. PHP ist mittlerweile so weit verbreitet das laut einer Statistik der PHP Group vom Januar 2013 ungefähr 90 Millionen Aktive Webseiten existieren, die von PHP unterstützt werden¹.

¹<http://www.php.net/usage.php>

Arbeitsweise von PHP

Wie bereits erwähnt wird der PHP-Code in die HTML-Seite eingebettet. Somit ist das Skript immer ein Teil des HTML-Codes und wird genau dann ausgeführt wenn das HTML-Dokument vom Client angefordert wird. Sollte der Client die HTML-Seite mit dem PHP-Code über den Browser anfordern, so startet der Server den PHP-Interpreter und übersetzt den PHP-Code. Das Resultat der Interpretation des PHP-Codes wird in das selbe HTML-Dokument geschrieben welches danach an den Client gesendet wird. Der Client sieht lediglich die resultierende HTML-Seite und bemerkt den PHP-Code nicht einmal [?]. Dadurch unterscheidet sich PHP von anderen Skriptsprachen wie z.B. Perl oder TCL, bei denen das Skript mit der Antwort an den Client gesendet wird.

„Die Webseite wird auf diese Weise dynamisch, d.h. erst ab dem Zugriffszeitpunkt durch den Client, erstellt und kann daher in Abhängigkeit von Benutzerinteraktion noch vor dem Versenden an den Client modifiziert werden“ [?, S.25].

Damit der Interpreter mit dem Server interagieren kann wird eine Virtuelle Maschine, die sogenannte *Zend-Engine*, zwischengeschaltet. Die Engine nimmt hierfür den vorhandenen PHP-Code und kompiliert diesen in den entsprechenden Maschinen-Code, welcher dann vom Server ausgeführt wird. Die Zend-Engine erlaubt es Caches während der Kompilierung hinzu zunehmen, welchen redundanten Bytecode speichern. Dadurch wird weniger Bytecode generiert und die Ausführung des Skriptes wird effizienter [?].

PHP gilt als vollständige Programmiersprache, da sie über ein Variablenkonzept, Kontrollstrukturen und eine Vielzahl von vordefinierten Funktionen besitzt. Hinzugekommen ist mit PHP 4 die Objektorientierung, die als Ersatz für die in PHP verwendeten Funktionskonzepte gilt. Dank der Objektorientierung ist es einfacher, komplexe Webanwendungen zu strukturieren und zu modifizieren. Bei der Nova GmbH wird PHP als Standard Implementierung für Webanwendungen benutzt.

Vorteile von PHP

Viele der Vorteile wurden bereits genannt und werden hier lediglich nochmals aufgeführt um zu verdeutlichen, warum PHP so weitverbreitet ist und wieso PHP überhaupt genutzt werden sollte. Dies sind natürlich nicht alle Vorteile sondern lediglich die wichtigsten im Bezug auf die Nutzung von PHP [?].

Vorteile:

- **Kompatibilität:** Der vielleicht größte und somit der wichtigste Vorteil von PHP ist dessen Kompatibilität mit beinahe jeder Hardware, Betriebssystem oder Serversystem. Als solches ist der PHP-Code portabel und kann auf beinahe allen Betriebssystemen und Serversystem ausgeführt werden (Einige der PHP eigenen Funktionen lassen sich ausschließlich in einer bestimmten Umgebung nutzen, diese sind jedoch in der Dokumentation von PHP vermerkt).
- **Performance:** Durch die Nutzung der Zend-Engine ist PHP schneller und performanter als andere Skriptsprachen.

- **Datenbank Anbindung:** Verbindungen mit Datenbanken werden über ODBC (Open Database Connectivity Standard) aufgebaut. Dadurch ist PHP in der Lage mit jeder Datenbank, welche über einen ODBC-Treiber verfügt, eine Verbindung aufzubauen. Zusätzlich gibt es eine Abstraktionsschicht die über *PHP Data Objekte*(PDO) realisiert wird [?]. Die Abstraktionsschicht sorgt dafür, das PHP die selben Funktionen zur Datenabfrage und Datenspeicherung auch bei verschiedenen Datenbanken nutzen kann ohne die SQL-Befehle umzuschreiben oder fehlende Features zu emulieren.
- **Bibliothek:** Da PHP für die Implementierung von Webseiten entwickelt wurde besitzt die Bibliothek von PHP bereits nützliche Funktionen und Features die für das Arbeiten im Web unerlässlich sind. PHP ist in der Lage XML zu parsen, kann E-Mails versenden und mit Cookies arbeiten. All dies mit nur wenigen Zeilen Code.

2.2.2 HTML

HTML steht für Hypertext Markup Language und ist eine Auszeichnungssprache für Dokumente. Sie wird zur Darstellung und Implementierung von Webseiten genutzt und dient als Standard des World Wide Web [?]. Als Auszeichnungssprache hat HTML den Vorteil das man mit jedem Text Editor HTML-Seiten erstellen kann. HTML wurde für die Präsentation von statischen Inhalten entwickelt und eignet sich zur Einbindung von Texten, Bildern oder Links.

Aufbau einer Webseite

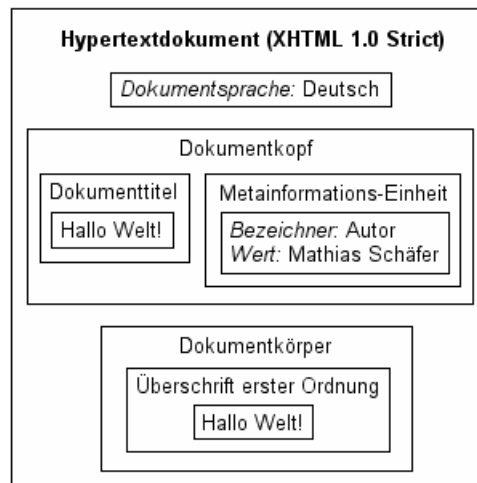
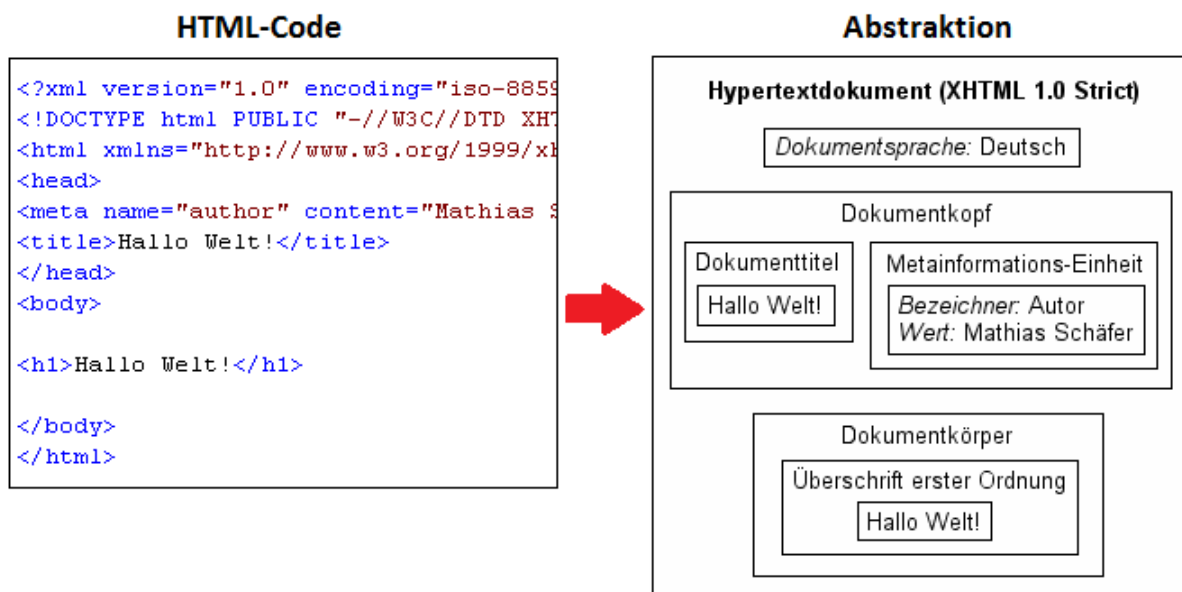
Eine Webseite (HTML-Dokument) teilt sich in *Elemente* auf. Ein Element kann dabei in verschiedene Gruppen von Elementen eingeteilt werden. So gibt es z.B Input-Elemente, welche Interaktionen mit dem Benutzer einer Webseite erlauben oder Paragraphen-Elemente, welche lediglich Texte darstellen [?].

Die zwei wichtigsten HTML-Elemente sind der Dokumentkopf(Header), welcher Meta-Informationen über die Webseite enthält und der Dokumentkörper(Body), welcher den darzustellenden Inhalt der Seite enthält(Abb.2).

Wie in Abb.2 erkennbar ist, bilden die Elemente eine Verschachtelungshierarchie. Das bedeutet jedes Element kann als Inhalt weitere Elemente enthalten. Die Elemente werden aber nicht in dem HTML-Dokument implementiert sondern sind lediglich die Abstraktion des HTML-Codes. Im Code selber wird über ein *Tag* der Beginn und das Ende eines Elementes markiert. Zusätzlich beschreibt ein Tag an welcher Hierarchie-Position sich besagtes Element befindet und um welche Art von Element es sich handelt. Im Starttag werden ebenfalls die Element-Attribute gesetzt. Der Inhalt des Elementes steht zwischen dem Starttag und dem Endtag.

Dokument Objekt Modell

Um die Webseite nun im Browser darzustellen sendet dieser eine Anfrage an den Webserver auf dem wiederum die HTML-Seite gespeichert ist. Das HTML-Dokument wird nun an

Abbildung 2: Die Elemente eines HTML-Dokumentes, Quelle².Abbildung 3: Abstraktion des HTML-Codes, Quelle².

den Browser als Antwort der Anfrage gesendet. Der Parser des Internetbrowsers parst den HTML-Code und generiert aus diesem ein Dokument Objekt Modell (DOM). Der DOM-Baum wird nach seiner Generierung in den Arbeitsspeicher abgelegt. Aus dem DOM erzeugt der Browser nun die Darstellung des Dokumentes [?].

Das DOM enthält im Normalfall lediglich die Inhalte der Webseite, nicht jedoch das

² <http://molily.de/dokumentmodelle>

Seiten-Layout. Das Layout und damit das Design der Elemente wird meistens in eine CSS-Datei geschrieben (Cascading Stylesheet). Die Information wo das Stylesheet hinterlegt ist befindet sich im Header des HTML-Dokumentes als Metainformation. Diese strikte Trennung von Inhalt und Layout ist einer der Vorteile von HTML.

HTML ist grundsätzlich nicht in der Lage, dynamische Inhalte darzustellen. Um dieser Beschränkung entgegenzuwirken wurden im Laufe der Zeit mehrere Erweiterungen an HTML vorgenommen. Zu den bekanntesten Interaktiven Erweiterungen zählen *Flash* und *Java-Applets*. Dank diesen Programmen, die im Hintergrund ausgeführt werden, sind immer mehr Funktionen hinzugekommen, die es erlauben auch Interaktive HTML-Seiten zu generieren [?].

2.2.3 MySQL

MySQL ist ein relationales Datenbanksystem (RDBMS). Das heißt MySQL-Datenbanken arbeiten mit relationalen Datensätzen und Attributen die in Relation zueinander stehen [?]. Dafür werden die zu speichernden Daten in Tabellen abgebildet.

MySQL-Datenbanken verwenden die Client/Server-Architektur wobei der Server das Programm ist, das tatsächlich die Datenbank bearbeitet, während der Client seine Absichten über SQL-Abfragen (Structured Query Language) an den Server übermittelt [?]. Dem Client werden die resultierenden Daten vom Server zurückgesendet und können anschließend weiterverarbeitet werden.

MySQL-Datenbanksystemen bieten eine Reihe von Eigenschaften:

- MySQL ist ein Open-Source-Projekt. Dadurch fallen selbst bei kommerzieller Nutzung keine Gebühren an.
- MySQL ist im Vergleich zu anderen Datenbanksystemen äußerst schnell und die Performanceverluste bei der Kommunikation mit Schnittstellen sind minimal.
- Der Datenbankserver ist mit einfachen und wenigen Befehlen zu bedienen.
- MySQL ist weitverbreitet und somit gibt es viele Programmiersprachen (C, Perl, PHP, etc.) die bereits integrierte Schnittstellen für MySQL anbieten.

Neben diesen spezifischen Eigenschaften von MySQL wären noch die generellen Vorteile von Datenbanksystemen aufzuführen. Diese sind beispielsweise die Kompatibilität mit SQL und die Portabilität durch die Client/Server-Architektur [?].

Aufgaben einer relationellen Datenbank

Die wichtigste Aufgabe einer Datenbank ist die Verwaltung beliebiger Daten [?]. Unter dem Begriff *Verwaltung* werden nun alle Funktionen zusammengefasst, welche die Daten innerhalb der Datenbank beeinflussen. Dazu gehört unter anderem die Funktion, neue Daten in die Datenbank zu speichern (Insert), Daten aus der Datenbank zu löschen (Delete), vorhandene Daten innerhalb der Datenbank zu verändern (Update), und Daten aus der Datenbank auszulesen (Select).

Damit die Datenbank die Verwaltung effektiv und effizient umsetzen kann ist es wichtig die Daten korrekt und strukturiert in die Datenbank anzulegen.

Structured Query Language

Die Structured Query Language, kurz SQL, ist eine Datenbanksprache die in fast allen Datenbanken verwendet wird. Wie der Name *MySQL* vermuten lässt gehört auch MySQL zu diesen Datenbanken. SQL ist als Werkzeug ein Teil der Datenbank und bildet eine Schnittstelle zwischen einem Benutzer und der Datenbank. Dank SQL ist der Benutzer nicht zwingend darauf angewiesen über den Aufbau und die Struktur der Datenbank Bescheid zu wissen. Der Benutzer schreibt mit kurzen *Queries* seine Befehle und SQL liefert dann die angeforderten Daten aus der Datenbank.

SQL selbst besteht aus vier Teilen, wobei jeder Teil eine andere Aufgabe in Bezug auf die Datenbank oder die Daten innerhalb der Datenbank aufweist:

Datendefinition: Dieser Teil erzeugt die Datenstruktur. Er erzeugt unter anderem die verschiedenen Tabellen welche in der Datenbank vorkommen und definiert die Felder dieser Tabellen.

Datenmanipulation: Mit der Datenmanipulation werden die Datensätze bearbeitet. Die Datensätze können in Tabellen eingetragen, gelöscht oder verändert werden.

Datenabfrage: Über die Datenabfrage werden die Daten nach den eingegebenen Kriterien des Benutzers abgefragt und aus der Datentabelle ausgelesen.

Datenschutz: Dieser Sprachteil schützt die vorhandenen Daten innerhalb der Datenbank, sodass unbefugte Benutzer nicht an die Daten herankommen oder die Daten eventuell manipulieren können.

Neben der einfachen Nutzung und Funktionsweise von SQL gibt es einen weiteren wichtigen Grund der erklärt, warum MySQL oder andere relationale Datenbanken soweit verbreitet sind. Die SQL-Befehle können nicht nur im Datenbanksystem genutzt werden sondern können über Schnittstellen wie z.B ODBC-Treiber oder IDAPI von extern geliefert werden. So ist es einer Applikation möglich einen entsprechenden SQL-Befehl über eine Schnittstelle an die Datenbank zu senden und so auf die Datenbank zugreifen zu können. Über die Schnittstelle kann die Applikation somit Daten aus der Datenbank lesen, schreiben oder löschen.

MySQL bietet solche Schnittstellen für verschiedene Programmier- und Skriptsprachen. Darunter ebenfalls eine für PHP. Das ist ebenfalls der Grund weshalb PHP und MySQL so oft zusammen von Entwicklern genutzt werden.

2.2.4 jQuery

jQuery ist ein auf Javascript basierendes Framework. jQuery stellt eine Menge von praktischen Funktionalitäten und Features zu Verfügung, welche die Manipulation und das

Navigieren innerhalb des DOM-Baums ermöglichen. Zusätzlich bietet das Framework kompletten Ajax-Support und sorgt dadurch für eine Performancesteigerung der Webanwendung [?].

Dank jQuery ist ein Besucher in der Lage, mit der Webseite zu interagieren. So lässt sich z.B zusätzlicher Inhalt ein- bzw. ausblenden oder die Webseite kann durch visuelle Effekte und Animationen den Benutzer informieren oder warnen. Dies sorgt beim Benutzer für ein angenehmeres „Surferlebnis“.

Das Framework wird zusätzlich zur Validierung von Benutzereingaben benutzt. Die Eingaben eines Formulars können also bereits vor der Anfrage an den Server auf Richtigkeit geprüft werden. Durch diese clientseitige Validierung wird verhindert, dass Schadcode an den Server übermittelt wird.

Einbinden von JQuery

Da es sich bei JQuery um eine Javascript Bibliothek handelt wird jQuery auf die selbe Art und Weise in ein HTML-Dokument eingebunden wie Javascript. Das Einbinden erfolgt in zwei Schritten. Der erste Schritt liegt darin, im Header des Dokumentes einen Verweis auf die Verwendung von Javascript anzulegen. Das sind die Metainformationen die dem Dokument mitteilen, von wo die Javascript Funktionalität anzusprechen ist. Dabei wird die jQuery-Bibliothek entweder auf den Server gespeichert oder man verlinkt auf eine Onlineversion der Bibliothek auf der jQuery-Webseite.

Im zweiten Schritt wird im Body des HTML-Dokumentes ein *Script-Container* oder *Script-Element* angelegt. Innerhalb des Tags `<script>` wird nun der JQuery-Code implementiert. Dies ist nun der Script-Bereich. Beim Parsen des Dokumentes wird das Script ausgeführt sobald dieses Element bzw. dieser Bereich geparkt wird.

Arbeitsweise von jQuery und Javascript

Die Hauptaufgabe von jQuery ist es, eine Webseite Dynamisch zu erweitern. Um dies besser bewerkstelligen zu können durchläuft das Script mehrere Phasen [?].

- **Empfang des Dokumentes:**

Das Script wird geparkt und dabei einmal ausgeführt. Dadurch werden die Objekte und Funktionen, welche im Script vorhanden sind, definiert. Das Script ist zu diesem Zeitpunkt nicht in der Lage auf das gesamte HTML-Dokument zuzugreifen.

- **Dokument vollständig geladen:**

Das Script kann nun auf das gesamte Dokument zugreifen. Somit kann Javascript das DOM manipulieren indem es die Elementknoten anspricht und diese um einen Event-Handler erweitert. Dadurch werden die Elemente Interaktiv. Zusätzlich kann das Script den Inhalt oder die Darstellung von bestehenden Elementen verändern und dem DOM neue Elementknoten hinzufügen. Das Ganze wird unter dem Begriff *DOM-Manipulation* zusammengefasst.

- **Benutzer Interaktionen:**

Löst der Benutzer nun einen Event bzw. Ereignis aus, so reagiert der entsprechende Handler des Elementes. Der Event-Handler führt anschließend die Handler-Funktionen aus die im Script definiert wurden.

Aufgaben und Funktionsweise eines Event-Handlers

Ein Script läuft nicht dauerhaft, stattdessen „wartet“ es, bis ein Benutzer ein Ereignis auslöst. Es gibt verschiedene Arten von Ereignissen innerhalb von JQuery(onclick, onpush, onchange etc). Diese werden einem Element der Webseite zugeteilt. Wird ein passendes Ereignis auf dem Element der Webseite vom Javascript-Interpreter *registriert* so wird die Funktion, welche dem Ereignis zugeordnet wurde, ausgeführt.

Somit besteht ein Event-Handler in JQuery aus drei einzelnen Teilen. Dem Element das Überwacht wird(element, Abb.4), dem passenden Event(onevent, Abb.4) und der vom Script auszuführenden Funktion(handlerfunction, Abb.4).



```
element.onevent = handlerfunktion;
```

Abbildung 4: Beispiel eines Event-Handlers, Quelle³.

jQuery und Ajax

jQuery sorgt nicht nur dafür das Webseiten dynamischer werden sondern bietet vollen Ajax(Asynchronous JavaScript and XML) Support. Die Grundidee bei Ajax ist es, eine zusätzliche Schicht auf der Clientseite anzubringen. Diese Schicht ist die sogenannte Ajax-Engine. Die Engine soll die Interaktion der Besucher beobachten und eventuelle Anfragen an den Server übernehmen. Die Antworten werden dann interpretiert und die Seitendarstellung wird aktualisiert. Diese Client-Server Prozesse werden im Hintergrund durchgeführt, sodass der Benutzer diese nicht merkt. HTTP-Anfragen werden demnach durch Engine-Anfragen ersetzt, wobei Inhalte der Webseite immer automatisch angepasst werden, sodass ein erneuter Aufbauvorgang der kompletten Seite nicht mehr notwendig ist [?].

Die Ajax-Engine verwendet für diese asynchrone Datenübermittlung die im Webbrowser integrierte XMLHttpRequest Schnittstelle. Die Engine erzeugt ein XMLHttpRequest-Objekt, welches über Attribute und Methoden asynchron oder synchron auf Serverdaten zugreifen kann und die DOM-Manipulation übernimmt. JQuery bietet bereits vordefinierte Möglichkeiten um mit dem XMLHttpRequest-Objekt zu arbeiten. Neben der Eigenschaft der asynchronen Datenübermittlung und dem damit verbundenen Vorteil die Webseite mit neuem Inhalten zu versehen ohne diese neu laden zu müssen, hat Ajax eine weitere Eigenschaft. Dank Ajax lassen sich Prozesse nämlich Parallelisieren, was dazu führen kann die Performance einer Webseite stark zu verbessern.

³<http://molily.de/js/event-handling-grundlagen.html>

2.2.5 Bootstrap 3.0

Bootstrap 3.0 ist ein Open-Source Framework das Erweiterungen für die Front-End Entwicklung einer Webseite bietet. Es enthält HTML-Klassen, CSS-Komponenten und Javascript-Erweiterungen und erleichtert die Arbeit eines Entwicklers, der nun nicht mehr Standard-Komponenten wie z.B Buttons oder Dropdown-Menüs selbst implementieren muss. Mark Otto, einer der Entwickler von Bootstrap dazu:

„CSS moved beyond type, forms and grids. People get tired to create the same stuff [?].“

Eine wichtige Änderung von Bootstrap 2 auf 3 ist die Ausrichtung auf das „Mobile-First“ Prinzip. Während Bootstrap 2 noch optionale Mobil-Unterstützung für Webseiten anbot, wurde Bootstrap 3 so ausgerichtet, das es von Anfang an auf Responsiv-Webdesign abzielt. Als Vorteil von Responsiv-Webseiten zählt, das sich das Seiten-Layout der Bildschirmgröße des Browsers anpasst. Dadurch wird ein gleichbleibendes Design für verschiedene Endgeräte wie z.B Smartphones oder Tablets erreicht. Designer können somit auf das Gestalten von separaten Mobil-Webseiten verzichten [?]. Bootstrap hat das Entwerfen einer Oberfläche für Internetpräsenzen oder Webanwendungen sowohl für Entwickler als auch für Designer optimiert.

Grid-System von Bootstrap

Bootstrap basiert auf einem 12-spaltigen Raster-Layout System. Dieses Grid-System sorgt für die Umsetzung des Layouts der mit Bootstrap entworfenen Website. Dieses System erlaubt es, schnell und einfach Layouts zu erstellen und anzupassen. Des Weiteren sind die Spalten des Rasters responsive, was bedeutet, das sich das Layout fließend an die Höhe und Breite des Bildschirms anpasst.

Das Layout wird so aufgebaut das man mithilfe der Klassen-Attribute eines HTML-Elementes definieren kann, welche Position das Element im Raster einnimmt. Das Grid-System wird selbst ebenfalls mit speziellen Klassen-Attributen definiert. So gibt es die Klasse *row* welche eine einzelne Reihe im Raster bildet und die Klasse *col* welche eine der 12 Spalten innerhalb der *row* bildet.

Die Grid-Layout Klassen in Bootstrap

Wie bereits erwähnt wird ein Bootstrap Layout über die Klassen-Attribute realisiert. Es gibt für beinahe jede erdenkliche Layout-Funktion eine eigens entwickelte Klasse. In diesem Kapitel sollen die wichtigsten Klassen aufgeführt und deren Aufgaben erklärt werden.

container: Die Klasse in die später die *rows* eingefügt werden. Sie dient dem zentrieren des Raster-Systems und ist bereits durch das Padding und durch das Margin angepasst.

row: Diese Klasse erstellt eine einzelne Reihe innerhalb des Rasters und teilt diese in maximal 12 gleichgroße Spalten auf. Wird die Spaltenanzahl überschritten so werden die folgenden Layout-Elemente einfach in einer zweiten *pseudo-Reihe* unterhalb der Reihe platziert.

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1
.col-md-8									.col-md-4			
.col-md-4				.col-md-4				.col-md-4				
.col-md-6						.col-md-6						

Abbildung 5: Beispiel des Grid-Layouts, Quelle⁴.

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
Container width	None (auto)	750px	970px	1170px
Class prefix	.col-xs-	.col-sm-	.col-md-	.col-lg-
# of columns	12			
Column width	Auto	60px	78px	95px
Gutter width	30px (15px on each side of a column)			
Nestable	Yes			
Offsets	Yes			
Column ordering	Yes			

Abbildung 6: Die in Bootstrap möglichen Raster Optionen, Quelle⁴.

col: Die Klasse *col* bildet ein Spalten-Element innerhalb des Rasters. Der Inhalt einer Spalte wird automatisch Responsive und passt sich somit der Bildschirmauflösung an. durch den Prefix welcher der *col*-Klasse hinzugefügt wird, wird bestimmt, bei welcher Bildschirmauflösung sich das Layout neu strukturiert. Diese *Breakpoints* sorgen für ein optimiertes Layout bei unterschiedlichen Bildschirmauflösungen. Die vier unterschiedlichen Breakpoints können der Abb.6 entnommen werden. Nachdem Breakpointprefix (*col-md-*) wird nun noch durch eine Zahl angegeben wie viel Platz die Spalte in der Reihe einnimmt(Abb.5). Somit nimmt die Klasse *col-md-12* eine gesamte Reihe in Beschlag und in diese Reihe kann keine weitere Spalte, zumindest Horizontal, eingefügt werden.

⁴ <http://getbootstrap.com/css/>

2.3 SCRUM

SCRUM ist ein modernes und agiles Vorgehensmodell in der Softwareentwicklung. Als Vorgehen wird hier die Umsetzung eines zu entwickelten Produktes verstanden. SCRUM arbeitet dabei empirisch, inkrementell und iterativ [?]. Bei SCRUM wird das zu bearbeitende Produkt in Teilaufgaben unterteilt welche dann in sogenannten Sprints abgearbeitet werden sollen. Die Dauer der Sprints ist variabel und kann zwei bis vier Wochen betragen. Die Sprints wiederholen sich dabei solange bis die Teilaufgabe vollständig erfüllt ist.

Die Erfahrung hat gezeigt das ein Softwareprodukt nicht schneller oder besser entwickelt wird nur weil detailliertere Anforderungen definiert wurden. Vielmehr stehen die Funktionalitäten eines Produktes im Vordergrund und müssen deswegen auch ausformuliert und beschrieben werden. SCRUM misst diesen Funktionen der Anwendersicht große Bedeutung zu. Sollte das Teilprodukt nicht die gewünschten Funktionen abdecken oder haben sich diese im Laufe des Sprints geändert, so wird dieser Sprint einfach nochmals unter Berücksichtigung der neuen Bedingungen durchgeführt.

SCRUM ist hervorragend geeignet um Prototypen zu entwickeln, da sich die Anforderungen und Funktionalitäten eines solchen Produktes während der Entwicklung jederzeit ändern können und hier die Vorteile der flexible Sprints optimal eingesetzt werden können. Durch regelmäßige Evaluationen am Ende eines jeden Sprints sind die Teilprodukte jederzeit Auslieferungsfähig. Somit ist der Kunde, wenn nötig, in der Lage, bereits Teile seines Produktes zu verwenden obwohl das gesamte Produkt noch nicht vollständig entwickelt wurde.

2.3.1 SCRUM-Rollen

SCRUM kennt explizit drei Rollen die für die Umsetzung eines Produktes verantwortlich sind (laut Wikipedia gibt es noch zusätzliche Rollen die jedoch extern Einfluss auf SCRUM nehmen und hier nicht näher beschrieben werden sollen). Zu den drei Rollen gehört zum einen der Product-Owner, der SCRUM-Master und das Entwicklerteam. Zusammen bilden diese Rollen das SCRUM-Team und teilen sich dabei in verschiedene Aufgabengebiete auf, die in diesem Abschnitt detaillierter beschrieben werden.

Product-Owner: Der Product-Owner hat die Aufgabe eine klare *Produktvision* (Ziel des Produktes) aus den Kundenwünschen zu erarbeiten. Häufig wird der Kunde ebenfalls in diesen Prozess integriert. Außerdem obliegt dem Product-Owner die Verantwortung über die konzeptionelle Planung und Priorisierung der Produkteigenschaften. Somit werden auch die vom Entwicklerteam durchgeführten Sprints vom Product-Owner auf deren Funktionalität überprüft. Er allein entscheidet ob ein Sprint bei qualitativen Mängeln erneut durchgeführt werden soll [?].

Zusätzlich trifft der Product-Owner die Entscheidung über den Auslieferungszeitpunkt, die Funktionalität des Produktes und die Kosten.

SCRUM-Master: Der SCRUM-Master ist für das Gelingen des SCRUM-Verfahrens ver-

antwortlich. Er führt die SCRUM-Regeln ein und überprüft ob sich das SCRUM-Team an diese hält. Bei Meetings ist er der Moderator und Ansprechpartner für alle Beteiligten.

Eine weitere Aufgabe des SCRUM-Master ist es, Hindernisse, welche die Umsetzung des Projektes gefährden, zu erkennen und zu beseitigen. Als Werkzeug steht dem SCRUM-Master hierfür das Impediment Backlog zu Verfügung, in die Impediments(Hindernisse) eingetragen werden können.

Als Hindernisse können zum Beispiel persönliche Differenzen innerhalb des Entwickler-teams oder des SCRUM-Team oder Störungen von Außen wie z.B Änderung der Funktionalität durch den Kunden während eines Sprints [?] gezählt werden.

Entwicklerteam: Das Team Sorgt für die Umsetzung der vom Product-Owner beschlossenen Funktionalitäten. Dabei muss sich das Entwicklerteam an die geforderten Qualitätsstandards halten, welche wiederum in Rücksprache mit dem Product-Owner validiert werden.

Ein Entwicklerteam sollte in der Lage sein, einen Sprint ohne äußere Einflüsse umsetzen zu können. Das Team organisiert sich dafür selbst und setzt die Aufgaben so um, wie sie es für richtig halten (ohne Einwirkungen des Owners oder des Masters). Deswegen ist es wichtig das die Mitglieder des Entwicklerteams interdisziplinär zusammengestellt werden.

Die Aufgaben des Entwicklerteams sind das Abschätzen des Umfangs eines *Backlogentries* welche eine der Produktfunktionalitäten symbolisiert, die vom Product-Owner aufgestellt wurde. Diese Backlogs, aus denen ein Sprint aufgebaut ist, werden vom Entwicklerteam in *Tasks* eingeteilt von denen jeder nicht länger als einen Tag zum realisieren bräuchte. Über diese Tasks wird das Projekt während der Implementierungsphase vom Entwicklerteam innerhalb der Sprints umgesetzt.

Ein Entwicklerteam sollte aus mindestens drei bis maximal neun Mitgliedern bestehen. Es müssen genügend Mitglieder sein, um alle benötigten Kompetenzen abzudecken. Die Gruppe sollte allerdings nicht zu groß werden, da sonst der Koordinationsaufwand innerhalb des Entwicklerteams stark zunimmt [?].

2.3.2 Der Ablauf eines SCRUM-Prozesses

Discovery Phase

Die Discovery Phase ist der erste Schritt in einem SCRUM-Prozess. In dieser Phase stellt ein Kunde seine Projektidee einem Product-Owner vor. Dieser entwickelt dann alleine, oder in Zusammenarbeit mit dem Entwicklerteam oder dem Kunden, eine *Produktvision*. Anschließend werden die Funktionalitäten des Produktes, die sogenannten *Product Backlog Items*, aus der Produktvision erarbeitet. Die Backlog Items(User Stories) werden in einer Liste, priorisiert vom Product-Owner, zusammengetragen. Diese Liste wird *Product Backlog* genannt. Ist das Product Backlog erstellt, ist die Discovery-Phase abgeschlossen [?].

Strategische Planung

Nachdem das Product Backlog nun vorhanden ist, wird dessen Aufwand vom Entwicklerteam in der Strategischen Planung geschätzt. Dadurch wird der Umfang eines Backlog Items erkennbar und der Product-Owner erhält Informationen darüber, welche Kosten anliegen und wie viel Zeit das Projekt in Anspruch nehmen könnte.

Nicht nur für den Product-Owner ist diese Planung wichtig, sondern auch für das Entwicklerteam selbst. Die Entwickler können nun die Risiken und die Funktionsweise des Produktes besser bewerten und es wird verständlicher, welche Erwartungen an das Entwicklerteam gestellt werden [?].

Taktische Planung

Nach der Strategischen Planung folgt die Taktische Planung. In dieser Phase werden nun die *Backlog Items* in die Sprints unterteilt. Somit wird definiert, welche Produktfunktionalität innerhalb eines Sprints realisiert wird. Dafür trifft sich das SCRUM-Team in einem *Sprint Planning Meeting*.

Die vom Entwicklerteam zu erbringenden Backlog Items werden, nachdem sie dem Sprint zugeordnet wurden, *Selected Product Backlogs* genannt. Wenn allen Beteiligten verständlich geworden ist, was das *Selected Backlog* alles enthält, wird ein *Commitment* (Versprechen) zwischen dem Product-Owner und dem Entwicklerteam festgehalten. Nach dem Meeting stehen die Ziele für einen einzelnen Sprint fest.

Nun folgt ein zweites *Sprint Planning*, indem die *Selected Product Backlogs* vom Entwicklerteam analysiert werden, um das weitere Vorgehen zu planen. Dieses Sprint Planning kann man auch als Design-Workshop verstehen, bei dem Entwickler Skizzen für die einzelnen *Selected Backlog Items* entwerfen oder sich auf die benötigte Architektur einigen. Es entsteht eine Liste aus *Tasks* (Aufgaben) die wiederum erfüllt werden müssen um die *Selected Backlogs Items* zu gewährleisten. Die somit aus den *Selected Backlogs Items* entstandenen *Tasks* werden im *Sprint Backlog* zusammengetragen. Werden alle *Tasks* erfüllt, so ist der Sprint abgeschlossen.

Nach dem zweiten Sprint Planning kann damit begonnen werden, den Sprint zu realisieren. In einem Daily Scrum werden die *Tasks* besprochen und an die Entwickler verteilt. Eventuell kann ein *Task* auch verändert werden, falls dieser zu kompliziert ist, als das man ihn an einem Tag bewältigen könnte. Es können aber auch *Tasks* hinzukommen welche möglicherweise in den vorherigen Planungsphasen übersehen wurden. In der Implementierungsphase werden dann die *Tasks* umgesetzt und implementiert.

Bereits während der aktuelle Sprint noch nicht abgeschlossen ist, werden die nächsten Sprints im SCRUM-Team besprochen und vorbereitet. Falls Änderungen an den Backlogs nötig sind werden diese aktualisiert und nehmen dadurch selbst auf den aktuellen Sprint Einfluss.

Am Ende des Sprints präsentiert das Entwicklerteam in einem *Sprint-Review* die in diesem Sprint implementierte Funktionalität. Erfüllt die Review die Qualitätsmerkmale und die Ziele(Commitment) kann dieser Teil bereits als sogenanntes *Potential Shippabel Product Increment* an den Kunden ausgeliefert werden. Über die Reviews wird ebenfalls deutlich, wo die Produktentwicklung steht und welche Produkteigenschaften noch Umgesetzt werden müssen [?].



Abbildung 7: Darstellung eines SCRUM-Prozesses, Quelle⁵.

3 Anforderungsanalyse

In diesem Abschnitt werden die Anforderungen für die Webanwendung ermittelt und analysiert. Die Anforderungsanalyse wird durchgeführt um zu garantieren, dass die Anwendung alle Wünsche des Kunden erfüllt. Die Anforderungen ergeben sich aus den im Kapitel 3.2 genannten Aufgaben, der Teilnehmer-Anmeldung und der Teilnehmer-Zuordnung, welche der Kunde im Gespräch explizit erläutert hat. Aus der vom Kunden aufgeführten Beschreibung der Aufgaben lassen sich nun die benötigten Anforderungen und deren resultierende Funktionen für die Anwendung ermitteln.

Die Anforderungen werden hierfür in Funktionale-Anforderungen und Nichtfunktionale-Anforderungen unterschieden. Die Funktionalen-Anforderungen beschreiben welche Funk-

⁵<https://www.3m5.de/scrum/>

tionen ein System, in diesem Fall die Webanwendung, ausführen können muss [?]. Als Nichtfunktionale-Anforderungen werden technische Qualitätsmerkmale wie Benutzbarkeit, Verfügbarkeit oder Sicherheit aufgeführt [?]. Die Umsetzung der Analyse wird schriftlich festgehalten und dient im Folgeschritt dem Entwerfen der Use-Cases und dem definieren der Anwendungslogik.

Jede Anforderung erhält eine eindeutige Nummer mit der sie identifiziert werden kann sowie einen Präfix. Der Präfix gibt an ob es sich bei der Anforderung um eine Funktionale-Anforderung (FA) oder um eine Nichtfunktionale-Anforderungen (NA) handelt. Außerdem wird der Anforderung ein passender Name gegeben und die Anforderung wird zusätzlich mit ein bis zwei Sätzen erläutert [?]. Anhand einer Referenz lassen sich Abhängigkeiten zwischen den Anforderungen ablesen.

3.1 Fachliche Anforderungen

Die Anforderungen werden tabellarisch in einer Liste aufgeführt.

Name	Teilnehmer-Anmeldung
Beschreibung	Der Teilnehmer muss sich mit seinen persönlichen Daten in die Teilnehmerliste eintragen können.
Nummer	FA 1
Referenz	-

Name	Partner hinzufügen
Beschreibung	Der Teilnehmer muss in der Lage sein, bei der Anmeldung seinen Partner hinzuzufügen.
Nummer	FA 2
Referenz	FA 1

Name	Teilnehmer Vegetarier
Beschreibung	Der Teilnehmer muss angeben ob er ein Vegetarier ist oder nicht.
Nummer	FA 3
Referenz	FA 1

Name	Gastgeber Optionen
Beschreibung	Der Teilnehmer muss bei der Anmeldung entscheiden ob er als Gastgeber oder als Gast am Staffeleessen Teilnehmen möchte.
Nummer	FA 4
Referenz	FA 1

Name	Gastgeber Optionen: Gericht
Beschreibung	Der Gastgeber muss bei der Anmeldung angeben, welches der drei Gerichte er zubereitet. Zusätzlich gibt er an, ob es sich bei seinem Gericht um eine vegetarische Speise handelt und wie viele Gäste er erwartet.
Nummer	FA 5
Referenz	FA 1, FA 4

Name	Administrator-Anmeldung
Beschreibung	Der Organisator muss sich bei der Anwendung als Administrator einloggen können.
Nummer	FA 6
Referenz	-

Name	Organisator selektiert einen Teilnehmer
Beschreibung	Der Organisator muss einen Teilnehmer aus der Teilnehmerliste auswählen können.
Nummer	FA 7
Referenz	FA 6

Name	Organisator editiert einen Teilnehmer
Beschreibung	Der Organisator muss alle Angaben des Teilnehmers verändern dürfen.
Nummer	FA 8
Referenz	FA 6, FA 7

Name	Organisator löscht einen Teilnehmer
Beschreibung	Der Organisator muss einen Teilnehmer aus der Teilnehmerliste entfernen dürfen.
Nummer	FA 9
Referenz	FA 6, FA 7

Name	Organisator speichert Änderungen
Beschreibung	Der Organisator speichert die Änderungen oder das Löschen eines Teilnehmers. Die geänderten Informationen werden dabei an die Anwendung gesendet.
Nummer	FA 10
Referenz	FA 6, FA 7

Name	Starten der Teilnehmer-Zuordnung
Beschreibung	Nur der Organisator soll die Zuordnung starten können, welche aus der Teilnehmerliste die Gästeliste erzeugt. Die Gästeliste soll nach den Anforderungen der Teilnehmer-Zuordnung erstellt werden.
Nummer	FA 11
Referenz	FA 6

Name	Teilnehmer-Zuordnung: Verteilen
Beschreibung	Die Teilnehmer-Zuordnung muss einen Teilnehmer so verteilen das er alle drei Gerichte zu sich nimmt.
Nummer	FA 12
Referenz	FA 13, FA 14, FA 15

Name	Teilnehmer-Zuordnung: Begegnungen überwachen
Beschreibung	Teilnehmer dürfen sich nicht doppelt begegnen.
Nummer	FA 13
Referenz	FA 12, FA 14, FA 15

Name	Teilnehmer-Zuordnung: Vegetariern
Beschreibung	Teilnehmer die Vegetarier sind sollen wenn möglich nur zu vegetarischen Gerichten eingeteilt werden. Nicht-Vegetarier können sowohl in vegetarische als auch in normale Gerichte zugeordnet werden.
Nummer	FA 14
Referenz	FA 12, FA 13, FA 15

Name	Teilnehmer-Zuordnung: Fahrdauer
Beschreibung	Die Teilnehmer sollen eine vom Organisator eingestellte Fahrdauer von einem Gang zum anderen nicht überschreiten.
Nummer	FA 15
Referenz	FA 12, FA 13, FA 14

Name	Speichern der Gästeliste
Beschreibung	Die Gästeliste, welche aus der Teilnehmer-Zuordnung berechnet wurde, muss speicherbar sein.
Nummer	FA 16
Referenz	FA 6, FA 11-15

Name	Drucken der Instruktionen
Beschreibung	Die Gästeliste muss für den Organisator einsehbar sein. Aus der Gästeliste müssen nun die Instruktionen erstellt und gedruckt werden. Dabei ist auf den Inhalt und auf die Art der Instruktion zu achten.
Nummer	FA 17
Referenz	FA 6, FA 11-15, FA 16

Name	Teilnehmerliste sortieren
Beschreibung	Die Teilnehmerliste, mit welcher der Organisator die Editierung der Teilnehmer durchführt, soll sich sortieren lassen können.
Nummer	FA 18
Referenz	-

3.2 Nichtfachliche Anforderungen

Name	Rückmeldungen der Webanwendung
Beschreibung	Die Webanwendung soll durch Ladebalken den Teilnehmer über den Prozess der Teilnehmer-Anmeldung informieren.
Nummer	NA 1
Referenz	FA 1

Name	Gästeliste performant berechnen
Beschreibung	Die Gästeliste soll innerhalb eines Zeitrahmens von 10-15 Sekunden berechnet werden. Hierbei muss der Organisator über diesen Prozess aufgeklärt werden. Die Dauer kann je nach steigender Teilnehmerzahl ebenfalls wachsen.
Nummer	NA 2
Referenz	FA 9

Name	Unterstützung der Teilnehmer-Anmeldung
Beschreibung	Die Anwendung soll den Teilnehmer dabei unterstützen, das passende Gericht zu wählen. Für die Anwendung ist es wichtig, dass alle Gerichte gleichmäßig verteilt sind.
Nummer	NA 3
Referenz	FA 1

Name	Unterstützung der Teilnehmer Editierung
Beschreibung	Der Organisator muss von der Anwendung aufgeklärt werden welche Änderungen an der Teilnehmerliste vorzunehmen sind.
Nummer	NA 4
Referenz	FA 6

3.3 Use Cases

Aus den ermittelten Funktionalen-Anforderungen lassen sich nun die Use Cases definieren. Ein Use Case beschreibt, wie der Benutzer die Anwendung benutzt um sein Ziel zu erreichen [?] [?]. Mithilfe eines Use Cases lässt sich also ermitteln, welche Funktionalität ein System oder eine Anwendung bereitstellen muss, damit der Benutzer richtig mit der Anwendung arbeiten kann. Die Benutzer, die mit dem Use Case interagieren, heißen Akteure. Als Akteure für die Use Cases zählen an dieser Stelle sowohl die *Teilnehmer* als auch der *Organisator*. Ein weitere Akteur ist die Datenbank welche die gesammelten Informationen zum Staffeessen für die Anwendung bereitstellt oder sie speichert.

3.3.1 Beschreibung der Use Cases

Die Use Cases werden in Abb.8 aufgeführt. Die Abb.8 zeigt, wie die einzelnen Use Cases und die Akteure zusammenhängen. Eine detailliertere Beschreibung der Use Cases folgt in diesem Kapitel. Die Beschreibung soll das Diagramm sinnvoll erweitern. In ihr werden die Standardabläufe des entsprechenden Use Cases beschrieben und es wird geklärt, welche Funktionalen-Anforderungen vom Use Case abgedeckt werden. Zusätzlich werden die Bedingungen der einzelnen Use Cases aufgeführt und Alternativabläufe beschrieben.

Use Case:	UC 1
Name:	Eintragen in die Teilnehmerliste
Bedingung:	Keine
Standardablauf:	<p>Schritt 1: Auf der Startseite trägt der Teilnehmer seine persönlichen Daten ein.</p> <p>Schritt 2: Der Teilnehmer wählt aus, ob er einen Partner angeben will oder nicht und trägt eventuell dessen Namen ein.</p> <p>Schritt 3: Der Teilnehmer wählt aus, ob er ein Vegetarier ist, falls er einen Partner besitzt ist dieser ebenfalls ein Vegetarier.</p> <p>Schritt 4: Der Teilnehmer wählt aus, ob er ein Gastgeber ist. Falls er einer ist, gibt er noch sein Gericht, die Anzahl seiner Gäste und ob sein Gericht vegetarisch ist, an.</p> <p>Schritt 5: Der Teilnehmer bestätigt seine Angaben indem er auf den <i>speichern</i>-Button klickt.</p>
Alternativablauf:	Sollte der Teilnehmer vergessen ein Feld auszufüllen oder hat er einen Fehler beim eintragen begangen (z.B in das Postleitzahlfeld keine Nummer sondern seinen Straßennamen eingetragen), so wird ihm beim Bestätigen des <i>speichern</i> -Button das fehlerhafte Feld rot unterlegt angezeigt und seine Angaben werden nicht an die Anwendung übermittelt.
Anforderungen	FA 1, FA 2, FA 3, FA 4, FA 5

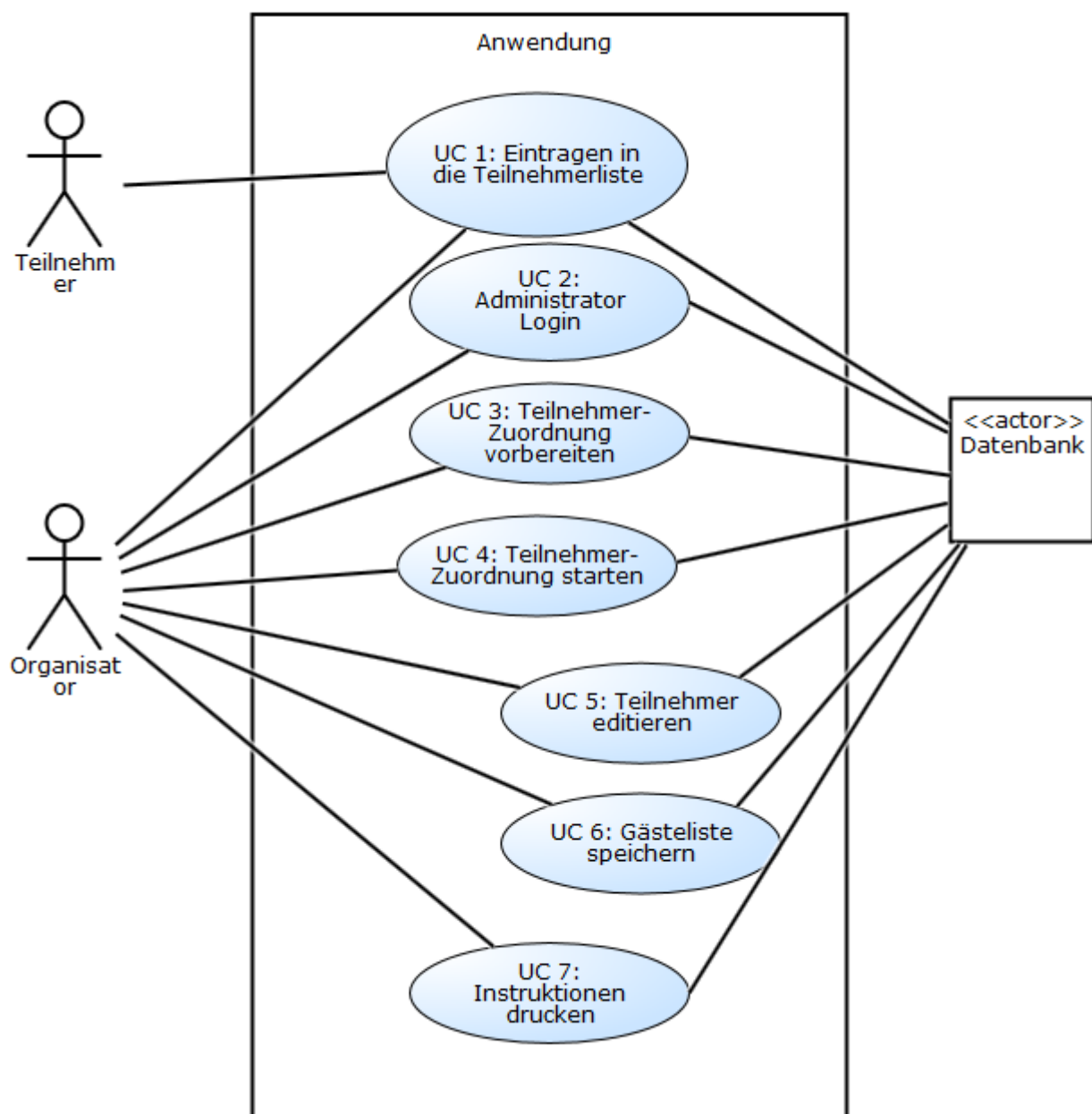


Abbildung 8: Use Case Diagramm.

Use Case:	UC 2
Name:	Administrator Login
Bedingung:	Keine
Standardablauf:	<p>Schritt 1: Der Organisator wählt in der Seitennavigation den Menüpunkt Login aus.</p> <p>Schritt 2: Der Organisator wird auf die Login-Seite weitergeleitet, wo er seinen Benutzernamen und sein Passwort einträgt.</p> <p>Schritt 3: Der Organisator bestätigt mit dem <i>anmelden</i>-Button seine Angaben. Der Organisator ist nun registriert.</p>
Alternativablauf:	Sollte das Passwort oder der Benutzername inkorrekt sein so werden die entsprechenden Felder rot unterlegt und der Organisator wird nicht als Administrator registriert.
Anforderungen	FA 6
Use Case:	UC 3
Name:	Teilnehmer-Zuordnung vorbereiten
Bedingung:	Der Organisator muss als Administrator eingeloggt sein.
Standardablauf:	<p>Schritt 1: Über die Seitennavigation wählt der Organisator aus, dass er auf die Seite <i>Überblick</i> weitergeleitet werden will.</p> <p>Schritt 2: Auf der Überblick-Seite werden dem Organisator alle Gastgeber, verteilt auf ihre Gericht, angezeigt. Der Organisator kann über <i>Drag and Drop</i> einen Gastgeber in ein anderes Gericht verschieben.</p> <p>Schritt 3: Der Organisator wählt die maximale Fahrdistanzdauer in Minuten aus.</p>
Alternativablauf:	Sollten die Bedingungen nicht erfüllt sein ist der Button <i>zuordnen</i> blockiert und ein Warnhinweis wird eingeblendet.
Anforderungen	FA 11

Use Case:	UC 4
Name:	Teilnehmer-Zuordnung starten
Bedingung:	Bedingung 1: Organisator muss eingeloggt sein. Bedingung 2: Die Gastgeber müssen gleichmäßig auf alle Gerichte verteilt sein. Bedingung 3: Die Anzahl von vegetarischen Gerichten muss ausreichen, um alle Vegetarier zu versorgen.
Standardablauf:	Schritt 1: Der Organisator startet die Zuordnung der Teilnehmer über den <i>zuordnen</i> -Button. Schritt 2: Die Anwendung berechnet nun, welche Teilnehmer geeignet sind und verteilt diese auf die entsprechenden Tische. Schritt 3: Dem Organisator wird als Resultat eine komplette Gästeliste auf einer neuen Seite angezeigt.
Alternativablauf:	Sollten die Bedingungen nicht erfüllt sein, ist der <i>zuordnen</i> -Button blockiert und ein Warnhinweis wird eingeblendet.
Anforderungen	FA 12, FA 13, FA, 14, FA 15

Use Case:	UC 5
Name:	Teilnehmer editieren
Bedingung:	Organisator muss eingeloggt sein.
Standardablauf:	Schritt 1: Der Organisator wählt in der Seitennavigation das Feld <i>Teilnehmer editieren</i> aus. Der Organisator wird auf eine Seite weitergeleitet welche die komplette Teilnehmerliste tabellarisch aufführt. Schritt 2: Nun wählt der Organisator einen Teilnehmer aus, indem er auf dessen <i>edit</i> -Button klickt. Eine neue Seite öffnet sich. Schritt 3: Die Seite <i>Teilnehmer editieren</i> ist mit einem Formular versehen, das die momentanen Informationen des Teilnehmers enthält. Diese können nun nach belieben angepasst werden. Schritt 4: Die Eingaben werden über die Betätigung des <i>speichern</i> -Buttons an die Anwendung übermittelt.
Alternativablauf:	Sollte der Organisator ein Feld falsch bearbeitet haben oder eines der Felder leer gelassen haben, so wird ihm dieses fehlerhafte Feld rot unterlegt angezeigt und die Änderungen werden nicht gespeichert.
Anforderungen:	FA 7, FA 8, FA 9, FA 10, FA 18

Use Case:	UC 6
Name:	Gästeliste speichern
Bedingung:	Bedingung 1: Organisator muss eingeloggt sein. Bedingung 2: Eine berechnete Gästeliste muss vorliegen.
Standardablauf:	Schritt 1: Nachdem die Teilnehmer-Zuordnung abgeschlossen ist, wird dem Organisator die Gästeliste auf einer neuen Seite präsentiert. Diese Gästeliste kann nun über den Button <i>speichern</i> gesichert werden, wobei die Informationen der Gästeliste an die Anwendung übermittelt werden.
Alternativablauf:	Sollte der Organisator mit dem Ergebnis der Zuordnung unzufrieden sein, so kann er durch eine Aktualisierung der Seite eine komplett neue Zuordnung starten und erhält eine neue Gästeliste.
Anforderungen	FA 16

Use Case:	UC 7
Name:	Instruktionen drucken
Bedingung:	Bedingung 1: Organisator muss eingeloggt sein. Bedingung 2: Eine gespeicherte Gästeliste muss vorliegen.
Standardablauf:	Schritt 1: Der Organisator wählt in der Seitennavigation das Feld <i>gespeicherte Gästeliste</i> aus. Es öffnet sich eine neue Seite. Schritt 2: Der Organisator kann nun auf der neuen Seite die Gästeliste begutachten. Die Instruktionen, welche aus der Gästeliste abgeleitet werden, können nun über den <i>drucken</i> -Button ausgedruckt werden.
Alternativablauf:	Sollte keine Gästeliste gespeichert sein, so wird der Organisator über einen Hinweis darüber in Kenntnis gesetzt.
Anforderungen	FA 17

4 Konzept

In diesem Kapitel wird aus den vorliegenden Use Cases (UC1-UC7) ein Konzeptplan erarbeitet. Dieser soll aufführen, wie die einzelnen Use Cases umgesetzt werden können und muss somit die Problemstellungen jedes Use Cases, sowie einen entsprechenden Lösungsansatz, enthalten. Dadurch lassen sich Risikofaktoren bereits vor der eigentlichen Implementierung erkennen und vermeiden. Als Zentrale Problemstellung wird hier der

Use Case der Teilnehmer-Zuordnung (UC4) aufgeführt, welche das Kernstück der Anwendung ist. Als Kernelement ist die Teilnehmer-Zuordnung von enormer Bedeutung und, wie in der Beschreibung erklärt wird, auch dementsprechend komplex. Deswegen ist es wichtig, vor Beginn der Systementwurfsphase zu erörtern, wie die Anforderungen an die Teilnehmer-Zuordnung umgesetzt werden und welche mathematischen Grenzen der Zuordnung und somit der Anwendung aufliegen. Diese Einschränkungen müssen mit den Anforderungen abgeglichen werden. Gegebenenfalls ist es erforderlich, die Anforderungen bzw. die Use Cases umzuformulieren.

4.1 Beschreibung der Teilnehmer-Zuordnung

Die Teilnehmer-Zuordnung soll die Teilnehmer nach bestimmten Kriterien verteilen. Insgesamt gibt es vier Kriterien die bereits bei den Fachlichen-Anforderungen aufgeführt sind. Die Kriterien werden genutzt um passende Prozesse zu entwickeln, welche den Ablauf der Zuordnung so steuern, dass die Anforderungen(FA12-FA15) erfüllt werden. Somit bilden diese Prozesse die Funktionalität der Teilnehmer-Zuordnung.

Die Anforderungen(FA12-FA15), welche den Use Case UC4 bilden, stehen häufig im Widerspruch zueinander. Damit die Kriterien den Ablauf einer Zuordnung nicht zu stark einschränken werden die Anforderungen nach ihrer Wichtigkeit für den Ablauf des Staffeßessens priorisiert.

4.1.1 Zuordnungskriterien

Kriterien der Zuordnung nach ihrer Priorität:

- Einordnen eines Teilnehmers in allen Gängen(FA12).
- Teilnehmer dürfen sich nicht doppelt begegnen(FA13).
- Vegetarier müssen zu einem vegetarischen Gericht zugeordnet werden(FA14).
- Ein Teilnehmer soll eine bestimmte Fahrzeit von einem Gang zum nächsten nicht überschreiten(FA15).

4.1.2 Problemstellung Teilnehmer-Zuordnung

Betrachtet man die Zuordnung für sich, handelt es sich bei dieser Problemstellung um ein Mengenproblem. Die Abb.9 zeigt, wie sich die Teilnehmer als Elemente der Mengen auf die verschiedenen Untermengen aufteilen. Aus dieser Grafik lassen sich einige Schlussfolgerungen ziehen.

1. Schlussfolgerung: Gleichmächtigkeit aller Mengen.

Die Menge aller Teilnehmer(T) und die Menge aller Vorspeisen(GV) sowie die Menge aller Hauptgänge(GH) und die Menge aller Desserts(GD) sind gleichmächtig. Der Grund dafür liegt in der Bedingung, das jeder Teilnehmer, egal ob er Gastgeber oder Gast für das Gericht ist, jedes Gericht einmal zu sich nehmen muss. Aus dieser Bedingung lässt sich ableiten, das ein Teilnehmer in jeder Menge(Vorspeise, Hauptgang und Dessert) einmal

vorkommen muss. Umgekehrt darf ein Teilnehmer aber nicht mehr als einmal in einem der Gerichte vorkommen, denn das würde ja bedeuten, dass er ein Gericht zweimal zu sich nimmt (was bei einem Drei-Gänge-Menü nicht erlaubt ist).

Erkenntnis: Jeder Teilnehmer kommt also in jeder Menge **genau einmal** vor.

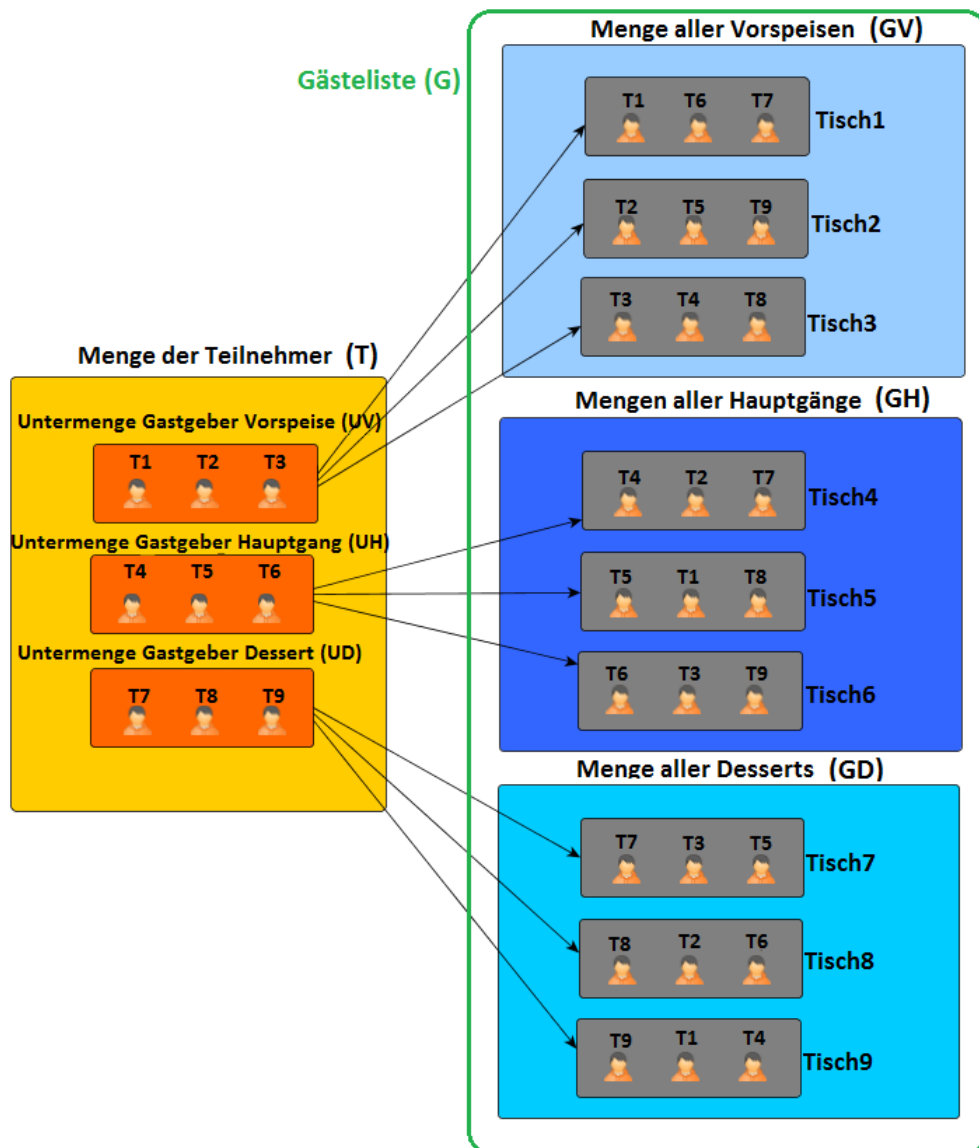


Abbildung 9: Die grafische Darstellung der Teilnehmer-Zuordnung.

2. Schlussfolgerung: Aufbau der Untermenge Tisch.

Die Untermenge *Tisch* enthält immer einen Teilnehmer aus der Untermenge Gastgeber Vorspeise *UV*, der Untermenge Gastgeber Hauptgang *UH* und der Untermenge Gastgeber

Dessert UD . Natürlich ist aber nur einer dieser Teilnehmer der Gastgeber des *Tisches*. Eventuell kann noch ein Gast, also ein Teilnehmer der kein Gastgeber ist, an einen *Tisch* hinzukommen (zur Veranschaulichung wird dieser aber ignoriert). Diese Permutation erlaubt die bestmögliche Verteilung der Teilnehmer, die Anhand der Abb.9 erklärt wird.

Betrachtet man in Abb.9 den Teilnehmer $T1$ ist dieser Gastgeber des *Tisch1*. Als solcher begegnet ihm der Teilnehmer $T6$, welcher einen Hauptgang zubereitet und der Teilnehmer $T7$, welcher der Gastgeber eines Desserts ist. Zu diesen beiden Gastgebern darf der Teilnehmer $T1$ also nicht mehr zugeordnet werden. Aus der Untermenge UH bleiben also noch die Teilnehmer $\{T4, T5\}$ welche als Gastgeber für $T1$ in Frage kämen und aus der Untermenge UD die Teilnehmer $\{T8, T9\}$. Durch die verbliebenen Teilnehmer ergeben sich vier verschiedene *Tisch*-Kombinationen(K1-K4):

$$K1 = \{T1, T4, T8\}$$

$$K2 = \{T1, T4, T9\}$$

$$K3 = \{T1, T5, T8\}$$

$$K4 = \{T1, T5, T9\}$$

Jede dieser Kombinationen kann für den Hauptgang oder für das Dessert gewählt werden. Deswegen ist es von Vorteil, einen *Tisch* aus Teilnehmern mit drei unterschiedlichen Gerichten zusammen zusetzen, da so die vielfältigsten Kombinationen erhalten bleiben. In der Abb.9 wurde K3 für den Hauptgang gewählt. Somit bleibt nur noch K2 für die Wahl des Desserts übrig. Es wäre aber ebenfalls möglich gewesen K1, K2 oder K4 für den Hauptgang zu wählen, dann müsste für das Dessert jeweils die noch verbleibende Kombination gewählt werden.

Erkenntnis: Ein *Tisch* besteht also immer aus **drei** Gastgebern die alle ein **unterschiedliches** Gericht vorbereiten (wie bereits erwähnt werden die Gäste noch ignoriert).

3. Schlussfolgerung: Anzahl der Teilnehmer

Die Anzahl der Teilnehmer leitet sich aus den Anforderungen FA12 und FA13 ab. Wenn ein Teilnehmer alle drei Gerichte speisen soll folgt daraus, das der Teilnehmer an drei verschiedene *Tische*, die sich in Vorspeise, Hauptgang und Dessert aufteilen, gesetzt werden muss. Als Beispiel dient wieder $T1$, dieser Teilnehmer wurde in Abb.9 folgenden *Tischen* zugeteilt:

$$\text{Vorspeise: } Tisch1 = \{T1, T6, T7\}$$

$$\text{Hauptgang: } Tisch5 = \{T1, T5, T8\}$$

$$\text{Dessert: } Tisch9 = \{T1, T4, T9\}$$

Wie bereits in der 2.*Schlussfolgerung* festgestellt wurde, muss ein *Tisch* immer aus drei Teilnehmern bestehen die alle ein unterschiedliches Gericht zubereiten. Für die Vorspeise ist $T1$ festgelegt. Dieser trifft an *Tisch1* auf den Gastgeber $T6$ des Hauptganges (der nun

Gast ist) und $T7$ des Desserts (ebenfalls Gast des Tisch1). Diese Teilnehmer werden jetzt also dem Teilnehmer $T1$ „bekannt“ gemacht.

$$Tisch1 = \{T1, T6, T7\}$$

$$Bekannt = \{T6, T7\}$$

Am $Tisch5$ werden dem $T1$ zwei noch unbekannte Gastgeber aus dem Hauptgericht bzw. aus dem Dessert zugeteilt. Diese werden ebenfalls in der Menge $Bekannt$ vermerkt.

$$Tisch5 = \{T1, T5, T8\}$$

$$Bekannt = \{T6, T7, T5, T8\}$$

Für den $Tisch9$ bleiben aus der Untermenge der Gastgeber Hauptgänge UH und aus der Untermenge der Gastgeber Dessert UD lediglich die Teilnehmer $UH = \{T4\}$ und $UD = \{T9\}$ übrig. Es bleibt also keine andere Möglichkeit als diese Teilnehmer an den $Tisch9$ zu setzen.

$$Tisch9 = \{T1, T4, T9\}$$

$$Bekannt = \{T6, T7, T5, T8, T4, T9\}$$

Es muss also von der Anwendung gewährleistet werden, dass an jedem $Tisch$ zwei „unbekannte“ Teilnehmer mit $T1$ gesetzt werden können. Einer dieser unbekannten Teilnehmer muss aus der Menge des Hauptganges (UH) und der andere muss aus der Menge des Desserts (UD) stammen. In diesem Beispiel trifft $T1$ alle Gastgeber des Hauptganges und des Desserts. Es müssen also sechs oder mehr Teilnehmer für $T1$ unbekannt sein, damit $T1$ niemanden Bekanntes trifft.

Erkenntnis: **Jeder** Teilnehmer, muss **sechs** unbekannte Gastgeber aus anderen Gerichten treffen. Diese sechs Gastgeber teilen sich gleichmäßig auf die verbleibenden Gerichte auf (in diesem Bsp. drei Gastgeber des Hauptgerichtes und drei Gastgeber des Desserts). Somit muss die Anzahl der Teilnehmer **pro Gericht** durch **drei teilbar** sein oder die Anzahl der Teilnehmer muss pro Gericht **größer** als sechs sein.

4.1.3 Konzeptioneller Entwurf der Teilnehmer-Zuordnung

Aus den *Erkenntnissen* und den *Schlussfolgerungen* lässt sich nun ein Konzeptioneller Entwurf für die Teilnehmer-Zuordnung entwickeln. Die Zuordnung soll die Teilnehmer anhand eines konkreten Musters verteilen. Dabei wird jeder Teilnehmer durch eine spezielle Permutation an einen passenden Tisch gesetzt. Die Permutationen, welche den Teilnehmer rotieren, bleiben immer gleich und lediglich die Position des Teilnehmers bestimmt dann, an welche Tische er platziert wird.

Dieses Verfahren hat den Vorteil, dass die Teilnehmer untereinander nicht Abgefragt werden müssen, da innerhalb der Zuordnung bereits verhindert wird, dass zwei inkompatible Teilnehmer zusammen gesetzt werden. Die Abfragen würden lediglich die Performance belasten und die Zuordnung unnötig beschränken.

4.1.4 Prozess: Teilnehmer-Verteilung

In diesem Prozess wird behandelt, wie die Teilnehmer verteilt werden können, ohne dass sie sich mehrmals begegnen. Der an dieser Stelle behandelte Lösungsansatz funktioniert ab einer gleichmäßig Verteilten Anzahl von insgesamt 21 Gastgebern. Diese Aussage deckt sich auch mit der *Erkenntnis* der *Schlussfolgerung* 3.

Die Gastgeber werden bei diesem Prozess in Gruppen zusammengefasst. Eine Gruppe bildet sich aus einem Gastgeber der Vorspeise, des Hauptganges und des Desserts. Die Gruppe ist dadurch genau so aufgebaut wie ein *Tisch*, hat aber mit diesem nur indirekt zu tun. Die Gastgeber, die eine Gruppe bilden, begegnen sich im gesamten Verlauf des Stafflessens nicht. Es wird also eine Gruppenliste erzeugt, die aus mindestens 7 Gruppen besteht (Abb.10).

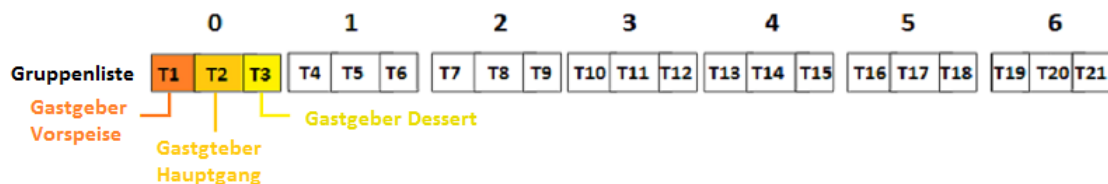


Abbildung 10: Die grafische Darstellung der Gruppenliste.

Nachdem die Gruppenliste erstellt wurde, wird diese im nächsten Schritt permutiert. Dabei wird der Index der Liste um 1 erhöht, somit wird die Gruppe die sich vorher auf der Position 0 befand mit der Gruppe an der Position 1 zusammengebracht. Das selbe geschieht natürlich auch mit den anderen Gruppen (Abb.11).

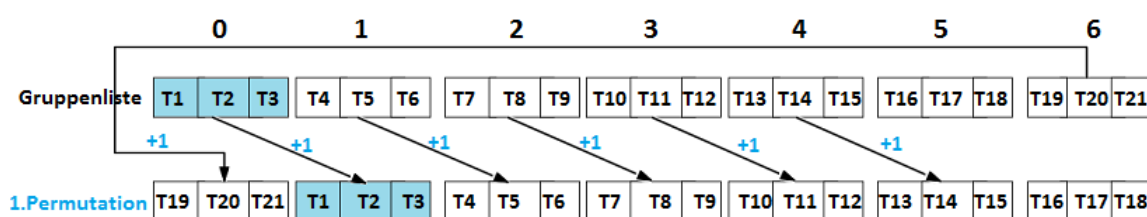


Abbildung 11: Die Gruppenliste nach der 1. Permutation.

Die 1. Permutation stellt nun die Gäste für die Gastgeber dar. Damit erhalten alle Gastgeber ihren spezifischen Gast. Betrachtet man den Gastgeber *T1* in der Abb. 11 so besitzt dieser den Gast *T19* und ist selber Gast beim Gastgeber *T4*. Wie dafür gesorgt wird, dass sich keine Gastgeber des selben Gerichtes treffen, wird im Prozess *Gerichte-Verteilung* unter Kapitel 4.1.5 ausführlich erklärt.

Da jeder Gastgeber zwei Teilnehmer als Gäste besitzen muss, muss die Gruppenliste erneut rotiert werden. Die 2. Permutation muss um eine Position weiter verschoben werden als die letzte, da sonst Überschneidungen mit Gruppen der letzten Permutation nicht zu vermeiden wären. Dafür wird nun der Index der 1. Permutation um 2 erhöht.

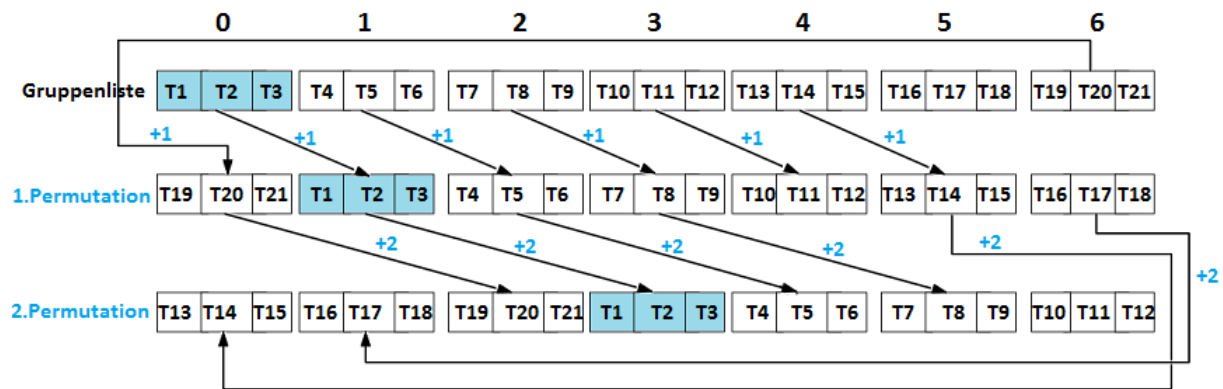


Abbildung 12: Die Gruppenliste nach der 2. Permutation.

Die 2. Permutation ist der vorletzte Schritt dieses Prozesses. Nun haben alle Gastgeber ihre Gäste zugeteilt bekommen und alle Gastgeber sind als Gäste verteilt worden. Aus den drei Gruppenlisten lassen sich nun wie in Abb.13 die *Tische* bilden.

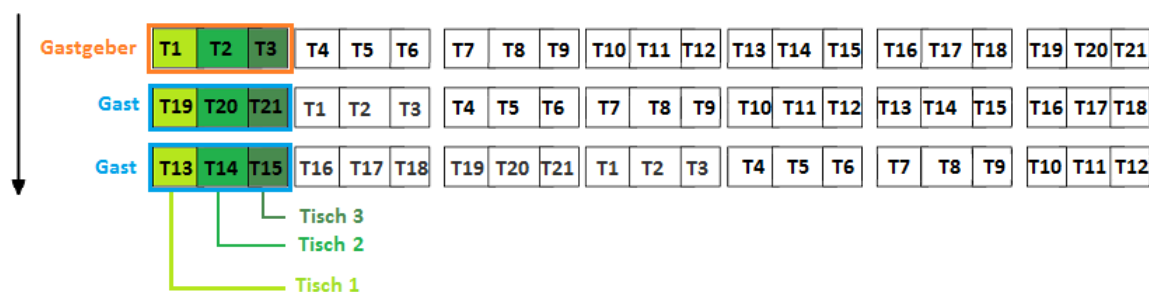


Abbildung 13: Bildung der *Tische* nach der Permutation.

Fasst man die Schritte dieses Prozesses zusammen, kommt man auf insgesamt vier Schritte, die von der Teilnehmer-Verteilung ausgeführt werden:

1. Schritt: Erzeuge eine Gruppenliste aus den angemeldeten Gastgebern.
2. Schritt: Führe die 1. Permutation durch.
3. Schritt: Führe die 2. Permutation durch.
4. Schritt: Bilde aus den Gruppen die *Tische*.

4.1.5 Prozess: Gerichte-Verteilung

Der Prozess Gerichte-Verteilung behandelt das Zuweisen der passenden Gerichte zu den entsprechenden Teilnehmern. Als solches müssen einem Gastgeber lediglich zwei Gerichte zugeordnet werden (da er selbst bereits ein Gericht vorbereitet) und einem Gast müssen drei Gerichte zugeordnet werden.

Auf die Gruppen und deren Position innerhalb der Gruppenliste wird kein Einfluss genommen. Stattdessen werden die Inhalte der Gruppen ebenfalls über eine Permutation manipuliert. Der Index 0 einer Gruppe entspricht dabei einer Vorspeise. Das bedeutet jeder Teilnehmer, der sich an der Position des Gruppenindex 0 befindet, erhält eine Vorspeise. Das gleiche gilt für den Hauptgang welcher dem Gruppenindex 1 zugeteilt ist und dem Dessert mit Gruppenindex 2. In Abb.14 wird dargestellt wie der Gastgeber *T1* in der 1. Permutation einem Hauptgang zugewiesen wird indem sein Index von 0 auf 1 erhöht wird.

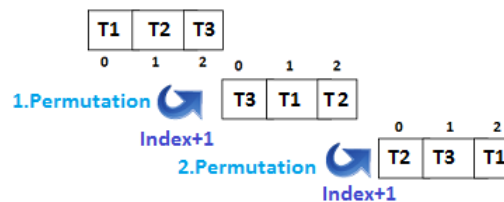


Abbildung 14: Permutation innerhalb der Gruppe.

Folgend auf die 2. Permutation wird der Gruppenindex von *T1* erneut um eins erhöht, um ihm ein Dessert zuzuweisen. Die Permutation betrifft alle Gastgeber der Gruppe gleichzeitig, dadurch werden die Gastgeber immer einem passendem Gericht zugeteilt.

Sowohl der Prozess der Teilnehmer-Verteilung wie auch der Prozess der Gerichte-Verteilung sind unabhängig voneinander und kein Prozess validiert das Ergebnis des Anderen. Somit kann gewährleistet werden, dass die Anforderungen sich nicht gegenseitig einschränken oder behindern. Lediglich der Zeitpunkt, wann die Gerichte-Verteilung ihre Permutation durchführen muss, ist durch die Teilnehmer-Verteilung vorgegeben.

Führt die Zuordnung beide Prozesse aus, wird eine vollständige und den Anforderungen FA12 und FA13 entsprechende Gästeliste gebildet (Abb.15). Betrachtet man nun in der Abb.15 den Teilnehmer *T1* und alle Tische in denen *T1* vorkommt, sind alle Teilnehmer die er trifft für ihn unbekannt und er speist alle Gerichte.

Vorspeise: $Tisch1 = \{T1, T21, T14\}$
 Hauptgang: $Tisch5 = \{T5, T1, T18\}$
 Dessert: $Tisch12 = \{T12, T8, T1\}$

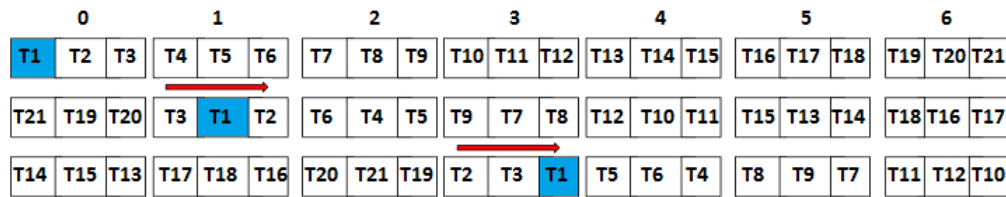


Abbildung 15: Komplett rotierte Gruppenliste.

Dieses Resultat deckt sich auch mit der *Schlussfolgerung 3* des Kapitel 6.1.2 und gilt für alle Teilnehmer gleichermaßen. Die Schritte der Gerichte-Verteilung teilen sich lediglich in die zwei Permutationen der Gruppeninhalte auf. Da diese Schritte aber dieselbe Funktionalität aufweisen ist es möglich, diese über eine einfache Schleife in der Implementierungsphase zu implementieren.

1. Schritt: Führe die 1. Permutation der Gruppeninhalte durch.
2. Schritt: Führe die 2. Permutation der Gruppeninhalte durch.

4.1.6 Prozess: Gäste hinzufügen

Ein weiterer Vorteil bei der Gruppenliste und der Benutzung von Gruppen, die einem vordefinierte Verteilungsschema folgen ist der, dass die Teilnehmer die als Gäste am Staffeessen teilnehmen sehr komfortable in die Teilnehmer-Zuordnung integriert werden können. Entsprechend hält sich der Implementierungsaufwand für die kommenden Phasen in Grenzen.

Der Aufbau der Gruppe erlaubt es einen Gast einfach aufzunehmen. Ein Gast wird hierfür in alle Positionen der Gruppe eingetragen, also in die Gruppenindizes 0, 1 und 2 (Abb.16). Infolgedessen erhält ein Gast alle drei Gerichte. Wenn man durch die Gruppenliste iteriert, ist man in der Lage dieses Schema auf alle Gruppen zu übertragen. Schließlich lässt sich für jede vorhandene Gruppe in der Gruppenliste ein Gast eintragen (in diesem Beispiel wären das sieben Gäste).

Dieses Vorgehen ist effizienter als einen Gast in der gesamten Gruppenliste zu verteilen, da hier unnötige Abfragen bzw. Vergleiche überflüssig sind. Es ist ein stabiles Verfahren mit einer guten Performance und vor allem einem unabhängigen Prozess, der als Baustein des Use Cases jederzeit ausgetauscht, modifiziert oder entfernt werden kann.

Als Anmerkung ist hier zu erwähnen, dass ein *Tisch* nicht mehr als vier Teilnehmer aufnehmen darf. Da ein Teilnehmer ebenfalls als Paar auftreten kann, wären dies im Worst-Case-Szenario genau acht Personen. Diese Personenzahl darf laut Kundendefinition nicht überschritten werden.

0	1	2	3	4	5	6
T1 T2 T3	T4 T5 T6	T7 T8 T9	T10 T11 T12	T13 T14 T15	T16 T17 T18	T19 T20 T21
T21 T19 T20	T3 T1 T2	T6 T4 T5	T9 T7 T8	T12 T10 T11	T15 T13 T14	T18 T16 T17
T14 T15 T13	T17 T18 T16	T20 T21 T19	T2 T3 T1	T5 T6 T4	T8 T9 T7	T11 T12 T10
G1 G1 G1	G2 G2 G2	G3 G3 G3	G4 G4 G4	G5 G5 G5	G6 G6 G6	G7 G7 G7

Abbildung 16: Gäste werden in die Gästeliste eintragen.

4.1.7 Prozess: Vegetarier-Verteilung

Der Prozess der Vegetarier-Verteilung hat die Aufgabe, die in der Gastgeberliste auftretenden Vegetarier zu verteilen und dafür zu sorgen, dass alle Vegetarier einem passendem veg. Gericht zugeteilt werden. Zum besseren Verständnis wird der Prozess in die verschiedenen Teilaufgaben aufgeteilt. Anhand der Teilaufgaben wird die Komplexität des gesamten Prozesses verdeutlicht.

Schritt 1: Sortieren der Vegetarier

Das Problem bei der Vegetarier-Verteilung ist die Notwendigkeit der Abfrage, ob ein Gastgeber Vegetarier ist oder ob ein Gastgeber ein vegetarisches Gericht zubereitet. Wie bereits für den Prozess der Teilnehmer-Verteilung vermerkt wurde, ist es erforderlich den Use Case der Teilnehmer-Zuordnung mit so wenig Abfragen wie möglich zu implementieren, da die Einschränkungen ansonsten den Ablauf der Teilnehmer-Zuordnung zu stark behindern würden. Bei der Vegetarier-Verteilung lässt sich dies jedoch nicht komplett vermeiden. Als solches gilt es eine Wahrheitstabelle aufzustellen, die angibt, welche Zustände des Gastgebers beachtet werden müssen.

Tabelle 1: Wahrheitstabelle der Zustände eines Gastgebers der Vorspeise.

Zustände	Gericht	Vegetarier	veg. Gericht
Zustand 1	Vorspeise	0	0
Zustand 2	Vorspeise	0	1
Zustand 3	Vorspeise	1	0
Zustand 4	Vorspeise	1	1

Aus den Tabellen.1 bis 3 lässt sich ableiten, dass zehn verschiedene Kombinationen des Gastgebers bei der Abfrage nach Vegetarier und veg. Gericht auftreten können. Für die Vegetarier-Verteilung ist es wichtig, dass jeder Vegetarier einem veg. Gericht zugeordnet wird. Umgekehrt spielt es jedoch keine Rolle, ob ein veg. Gericht ausschließlich aus Vegetariern besteht oder nicht. Deswegen müssen bei der Vegetarier-Verteilung lediglich die in den Tabellen markierten Kombinationen überwacht werden, die verbleibenden Zustände können ignoriert werden. Die Gastgeber müssen nun nach ihren jeweiligen Zuständen

Tabelle 2: Wahrheitstabelle der Zustände eines Gastgebers der Hauptgangs.

Zustände	Gericht	Vegetarier	veg. Gericht
Zustand 5	Hauptgang	0	0
Zustand 6	Hauptgang	0	1
Zustand 7	Hauptgang	1	0
Zustand 8	Hauptgang	1	1

Tabelle 3: Wahrheitstabelle der Zustände eines Gastgebers des Desserts.

Zustände	Gericht	Vegetarier
Zustand 9	Dessert	0
Zustand 10	Dessert	1

sortiert werden.

Schritt 2: Selektieren eines Vegetariers

Aus den sortierten Listen wird in diesem Schritt nun ein Vegetarier ausgewählt. Die Auswahl findet priorisiert statt. Das bedeutet, dass immer nach dem schwierigsten zuzuordnenden Vegetarier verteilt werden. Die Vegetarier, die zusätzlich veg. Gerichte anbieten, zählen zu dieser Kategorie.

Diese Vegetarier werden immer zuerst von der Vegetarier-Verteilung gruppiert. Bei der Vegetarier-Verteilung gilt es zu beachten, dass nur die Gastgeber der Vorspeise und des Hauptganges ein veg. Gericht zubereiten können, da es laut der Kundendefinition kein explizites vegetarisches Dessert gibt. Also werden meistens zuerst die Vorspeisen sortiert, anschließend die Hauptgänge und erst zum Schluss die Desserts. Die Verteilung orientiert sich anhand der Vegetarier und nicht anhand der zu Verfügung stehenden veg. Gerichte. Somit wird die Liste mit den Vegetariern iteriert und für jeden Vegetarier wird ein passendes Gericht gesucht.

Schritt 3: Suche innerhalb der Gruppenliste

Anhand des selektierten Vegetariers wird nun ein passendes veg. Gericht innerhalb der Gruppenliste gesucht. Als „passendes“ Gericht wird hier ein Gericht bezeichnet, in das der Vegetarier einsortiert werden kann. Die unterschiedlichen Gerichte, welche die Vegetarier zubereiten, sorgen für eine Aufteilung in drei mögliche Fälle.

Damit gilt für die Vegetarier der einzelnen Gerichte:

1. **Vegetarier der Vorspeise:** Handelt es sich um einen Vegetarier der eine Vorspeise zubereitet, so muss für diesen Vegetarier ein Gastgeber gefunden

werden, welcher einen veg. Hauptgang ausrichtet. Für das Dessert muss keine Abfrage durchgeführt werden.

2. **Fall Vegetarier des Hauptganges:** Handelt es sich um einen Vegetarier der einen Hauptgang zubereitet, so muss für diesen Vegetarier ein Gastgeber gefunden werden, welcher eine veg. Vorspeise ausrichtet. Für das Dessert muss keine Abfrage durchgeführt werden.
3. **Fall Vegetarier des Desserts:** Handelt es sich um einen Vegetarier der ein Dessert zubereitet, so muss für diesen Vegetarier jeweils eine veg. Vorspeise und ein veg. Hauptgang ermittelt werden.

Nun kann es bei der Suche zu zwei unterschiedlichen Folgeschritten kommen, je nachdem ob ein passendes Gericht vorhanden ist oder nicht. Sollte in der Gruppenliste kein passendes veg. Gericht für den selektierten Vegetarier gefunden worden sein, so folgt Schritt 4.1. Wird ein entsprechendes Gericht gefunden, so wird Schritt 4.2 an den Schritt 3 angehängt.

Schritt 4.1: Eintragen des Vegetariers ohne vorhandenes Gericht in der Liste

Als Grundlage für diesen Prozess dient wieder die Gruppenliste. Dank dem Verteilungsschema, das in Kapitel 4.1.4 beschrieben wird, lässt sich *voraussagen* wo ein Gastgeber zugeordnet wird. Dank dieser Voraussage ist die Teilnehmer-Zuordnung in der Lage, ein veg. Gericht an die Position zu setzen, in die ein Vegetarier rotiert wird. Es werden also unvollständige Gruppen mit jeweils einzelnen Teilnehmern erstellt und an die entsprechenden Positionen gesetzt. Erst nach diesem Prozess werden die Gruppen und die Gruppenliste vervollständigt. Somit steht fest, dass die Vegetarier-Verteilung noch vor der Teilnehmer-Verteilung durchgeführt werden muss.

Gibt es in der Gruppenliste kein passendes veg. Gericht, so wird der Vegetarier an die erstbeste Position platziert (Abb. 17). Dafür wird der Abstand ermittelt, den der Vegetarier zum veg. Gericht einnehmen muss. Aus der Liste der zu Verfügung stehenden veg. Gerichte wird nun ein passendes Gericht selektiert und an die berechnete Position gesetzt (Abb. 18).

Handelt es sich bei dem eingetragenen veg. Gericht ebenfalls um einen Vegetarier, so wie in Abb. 18, wird für diesen Vegetarier der Schritt 3 der Vegetarier-Verteilung nochmals ausgeführt, um für ihn ebenfalls ein passendes Gericht zu finden.

Schritt 4.2: Eintragen des Vegetariers mit passendem Gericht in der Liste

Wird ein veg. Gericht gefunden so wird berechnet an welche Position der Vegetarier gesetzt werden muss um dieses veg. Gericht zugeteilt zu bekommen. Der Vegetarier wird dann an die berechnete Position platziert falls diese noch frei ist. Ansonsten wird für den Vegetarier der Schritt 3 erneut ausgeführt und es wird ein anderes passendes Gericht gesucht. Damit wäre die erste Iteration des Prozesses abgeschlossen.

Anmerkungen zum Prozess Vegetarier-Verteilung

Der Prozess der Vegetarier-Verteilung folgt dem strikten Ablaufplan, welcher durch die einzelnen Schritte erklärt wurde. Als solches ähnelt dieser Prozess dem der Teilnehmer-

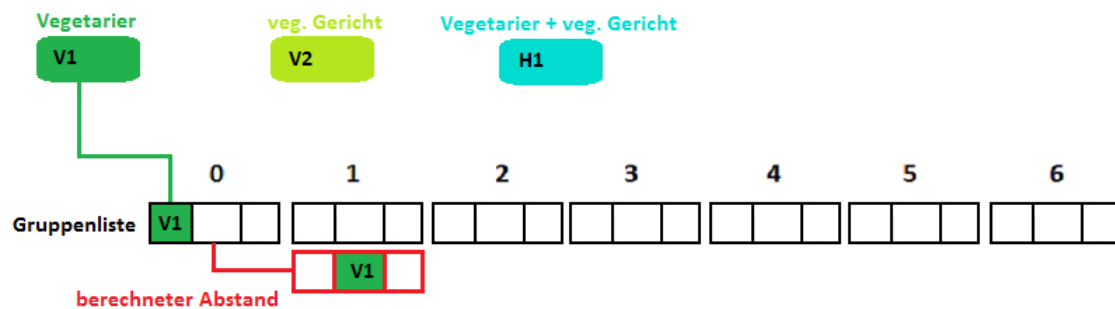


Abbildung 17: Vegetarier wird in die Leere Gruppenliste eingetragen.

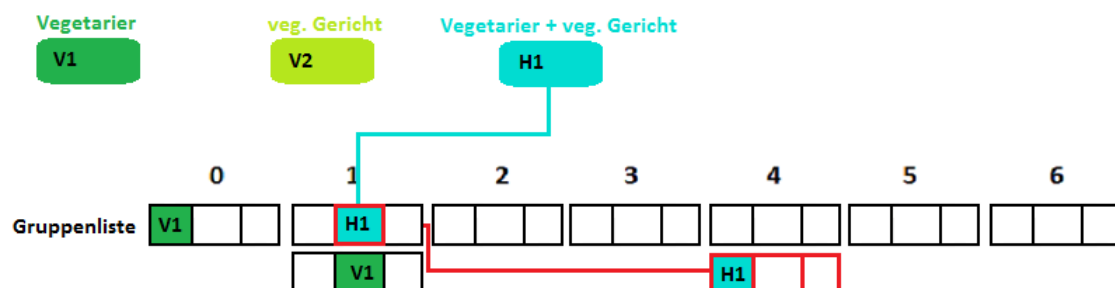


Abbildung 18: Ein passendes veg. Gericht wird in die Leere Gruppenliste eingetragen.

Verteilung in der Hinsicht, dass auch hier der Ablauf nicht von Abfragen beeinflusst werden darf. Um die Einflüsse von Abfragen zu minimieren wird die Sortierung aller Vegetarier in ihre unterschiedlichen Zustände innerhalb einer Teilnehmer-Zuordnung nur beim ersten mal ausgeführt.

Sollte bei der Verteilung der Vegetarier kein plausibles Ergebnis berechnet werden, ist es die Aufgabe des Organisators, die Teilnehmer so zu editieren, das die Vegetarier entsprechend verteilt werden können. Dies darf nämlich nicht eigenmächtig von der Anwendung entschieden werden.

Um den komplizierten Prozessablauf der Vegetarier-Verteilung besser nachvollziehen zu können, wird dieser in Abb.19 grafisch dargestellt.

4.1.8 Prozess: Teilnehmer-Zeitüberwachung

Anhand der Teilnehmer-Zeitüberwachung wird ermittelt, wie viele Zeitüberschreitungen innerhalb der Gästeliste auftreten. Das Resultat der Zeitüberwachung hat aber keinen direkten Einfluss auf die Gästeliste, da der Prozess nicht befugt ist, die Gästeliste zu optimieren. Stattdessen wird ein anderer Ansatz umgesetzt.

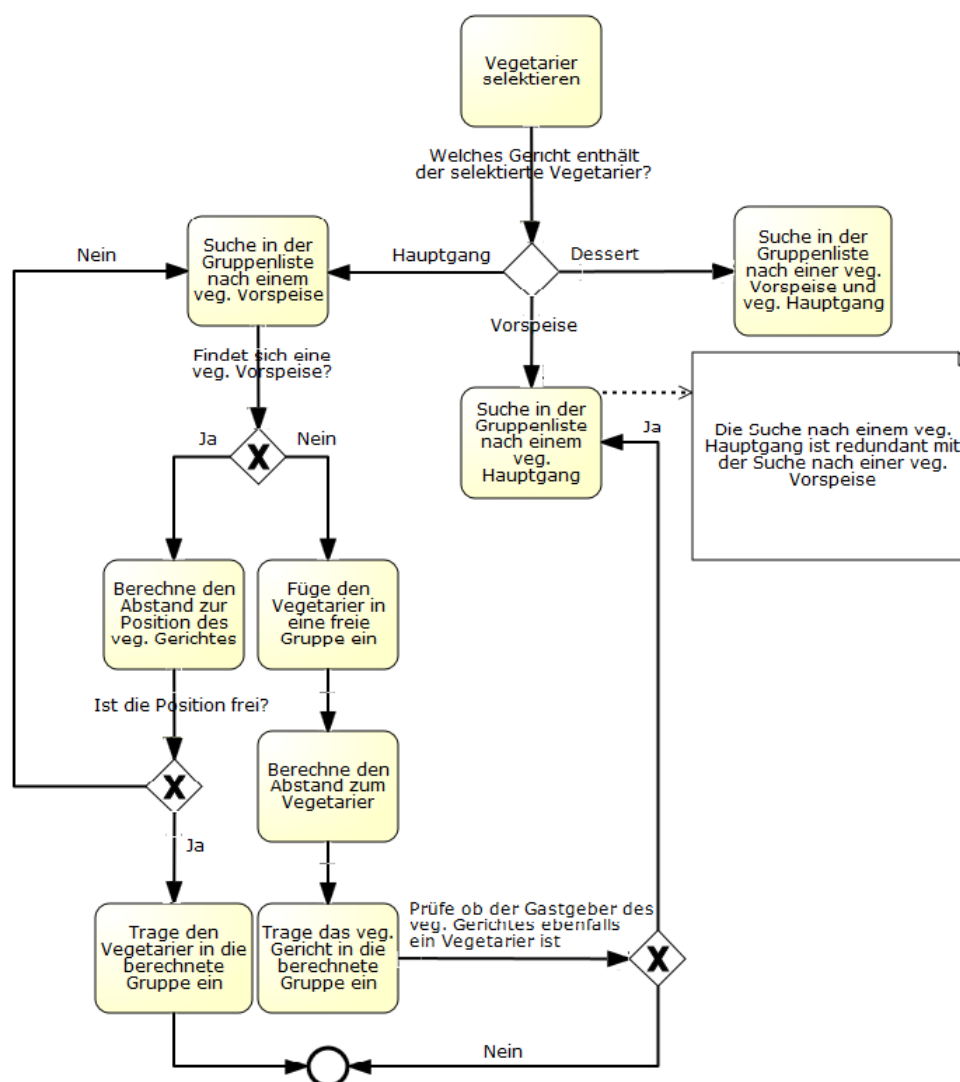


Abbildung 19: Der Ablauf des Prozesses Vegetarier-Verteilung.

Anstatt nur eine Gästeliste zu generieren und diese zu validieren, werden einfach mehrere verschiedene Gästelisten, von mehrfach ausgeführten Teilnehmer-Zuordnungen, berechnet. Die Zeitüberwachung wird dabei als Prozess von jeder Teilnehmer-Zuordnung selbstständig durchgeführt und die ermittelten Zeitüberschreitungen werden in einer Liste gespeichert. Aus diesen Resultaten wird dann die Gästeliste selektiert, welche die niedrigste Anzahl an Zeitüberschreitungen aufweist.

Um nun diesen Ansatz zu realisieren, übernimmt ein Manager die Steuerung über die Teilnehmer-Zuordnung. Der Manager hat die Aufgaben die Teilnehmer-Zuordnungen zu starten, die Gästelisten per Zufallsverfahren umzusortieren und diese als Inputs den Zuordnungen zu übergeben. Zusätzlich werden die generierten Gästelisten vom Manager evaluiert und die optimale Gästeliste als Resultat an die Anwendung übermittelt. Der

Manager wird in Abb.20 anhand eines einfachen Modelaufbaus erklärt.

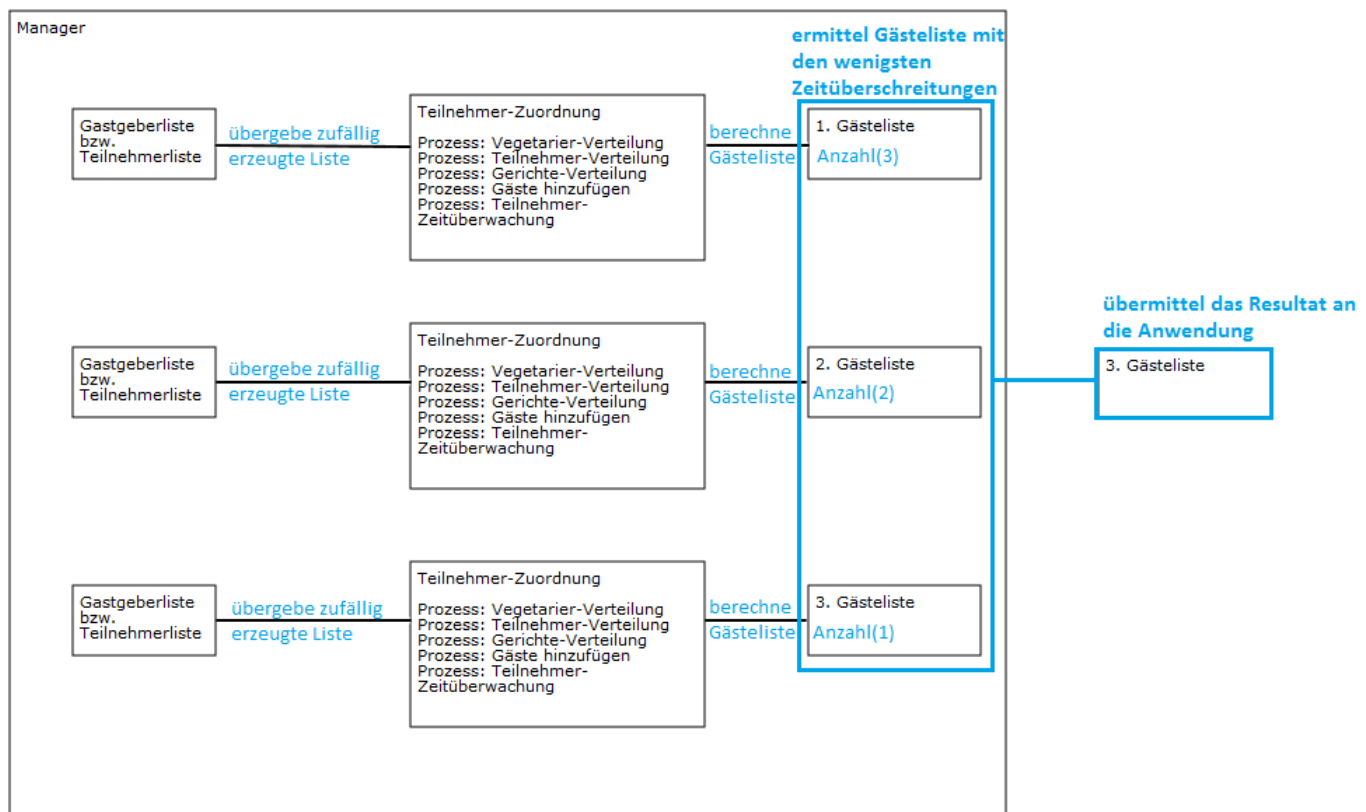


Abbildung 20: Model des Managers.

Der Manager wird an dieser Stelle erwähnt, um den Prozess der Zeitüberwachung besser zu verstehen. Der Manager führt die Zeitüberwachung durch, da es sonst nicht möglich wäre, die Anforderung FA15 in diesem Use Case sinnvoll zu integrieren. In den folgenden Kapiteln wird noch genauer auf den Manager und seine Funktionsweise eingegangen.

4.2 Beschreibung der Teilnehmer-Anmeldung

Die Teilnehmer-Anmeldung hat die Aufgabe die Teilnehmer für die Anwendung zu registrieren und alle für die Anwendung erforderlichen Daten vom Benutzer anzufragen oder zu ermitteln. Somit besitzt die Anmeldung folgende Prozesse, die abgearbeitet werden müssen. Als Erstes müssen die Teilnehmer-Informationen erfasst und validiert werden. Danach müssen die Geodaten für die Anwendung aus der angegebenen Adresse des Teilnehmers berechnet werden. Zum Schluss müssen alle Informationen in die Datenbank eingetragen werden. Diese Prozesse werden in den folgenden Kapiteln genauer erklärt.

4.2.1 Prozess: Teilnehmer-Informationen erfassen

Die Daten des Teilnehmers werden von einem einfachen Formular erfasst und validiert. Das Formular muss folgende Angaben vom Teilnehmer berücksichtigen können.

Angaben des Teilnehmers:

- Name des Teilnehmers
- Name des Partner/in
- Adresse
- E-Mail
- Handynummer
- Ist Teilnehmer ein Vegetarier
- Gastgeber Optionen
 - Wahl des Gangs.
 - Vegetarisch/Normal.
 - Anzahl Gäste

Die Angaben zum Partner, sowie die Gastgeber-Abfrage bzw. Vegetarier-Abfrage sind laut Anforderungen (FA1-FA5) variabel. Hier muss mit einem dynamischen Formular gearbeitet werden, welches seine Eingabefelder anhand der Auswahl des Benutzers anpasst. Die Teilnehmer-Informationen teilen sich in zwei Kategorien auf.

Kategorie 1: Persönliche Daten

Die erste Kategorie enthält die persönlichen Daten des Teilnehmers. Dazu zählen sein Name, der Name des Partners(falls vorhanden), die Adresse des Teilnehmers und die Kontaktdaten des Teilnehmers (also Handy und E-Mail). Diese Informationen wirken sich nicht direkt auf die Anwendung aus (bis auf die Adresse) und lassen sich deswegen zusammenfassen.

Kategorie 2: Anwendungsrelevante Daten

Die zweite Kategorie enthält all diejenigen Daten, welche für die Anwendung relevant sind, um den Teilnehmer sachgemäß einzuordnen. Darunter zählen die Angaben ob der Teilnehmer ein Vegetarier ist sowie die Gastgeber-Optionen. Das Formular mit den entsprechenden Kategorien wird in Abb.21 dargestellt.

Validierung der Eingabefelder

Nachdem der Benutzer seine Eingaben durchgeführt hat und über den *speicher*-Button (Abb.21) bestätigt, werden seine Angaben clientseitig validiert. Die clientseitige Validierung hat die Vorteile, dass sie Bandbreite und Zeit spart. Außerdem ermöglicht sie die explizite Darstellung eines einzelnen fehlerhaften Feldes oder einer fehlerhaften Eingabe. Für dieses Formular wird ein fehlerhaftes Feld mit einer roten Rand versehen. Die Validierung betrifft einzig die Angaben zu den persönlichen Daten des Benutzers und nicht die anwendungsrelevanten Daten. Das liegt daran das die anwendungsrelevanten Daten,

Teilnehmerregistrierung

Persönliche Daten

Name

Vorname

Partner hinzufügen

Straße

Nr

PLZ

Ort

@ test@web.de

Handynummer

Anwendungsrelevante Daten

Sind Sie Vegetarier

☐ ja

☒ nein

Können Sie Leute bei sich bewirten?

Ja Nein

Welchen Gang wollen sie ausrichten?

☐ Vorspeise

☒ Hauptgang

☐ Nachtsch

Ist Ihr Gericht vegetarisch?

☐ ja ☒ nein

Maximale Gästeanzahl: 1 ▼ bedenken sie das sie sich selber dazuzählen.

speichern

Abbildung 21: Das Anmeldeformular mit den persönlichen Daten und den anwendungsrelevanten Daten.

wie in Abb.21 zu erkennen ist, alle bereits vorselektiert wurden. Dadurch wird eine Validierung dieser Daten überflüssig. Sollten alle Eingaben des Benutzers fehlerfrei sein, so werden die Daten an den nächsten Prozess weitergeleitet.

4.2.2 Prozess: Geodaten-Ermittlung

Der Prozess der Geodaten-Ermittlung berechnet anhand der Adresse des Teilnehmers die Geodaten, also die Höhen- und Breitengrade des Wohnortes des Teilnehmers. Nur mit diesen Koordinaten ist die Anwendung in der Lage, die Fahrdistanz zwischen zwei Teilnehmern zu berechnen.

Das Ermitteln der Geodaten wird von der Google Geocoding API umgesetzt. Die Bedingungen zur Nutzung der API findet man auf der Google Maps Website⁶.

Die Google Geocoding API bietet eine direkte Möglichkeit, über eine HTTP-Anfrage auf einen Geocoder zuzugreifen, welcher eine Konvertierung von Adresse in Höhen- und Breitengraden vornimmt. Dabei wird die Adresse (in spezieller Form) als Variable an die Anfrage übergeben. Als Antwort erhält man eine JSON-Ausgabe (Javascript Object Notation) welche die Geodaten enthält (Abb.23). Diese JSON-Ausgabe wird anschließend in ein PHP-Objekt formatiert. Über dieses Objekt können nun die Geodaten angesprochen werden.

Da sich die Geodaten auf den Teilnehmer beziehen, werden sie zusammen mit den anderen Informationen des Teilnehmers in die Datenbank gespeichert. In den folgenden Prozessen kann nun über die Datenbank auf die Geodaten des Teilnehmers zugegriffen werden.

4.2.3 Prozess: Fahrdistanz-Berechnung

Der Prozess Fahrdistanz-Berechnung hat die Aufgabe die Distanz zwischen zwei Teilnehmern zu berechnen. Die Entfernung wird dabei von der Mapquest API berechnet. Die Lizenzbedingungen findet man auf der Mapquest Website⁷.

Die Berechnung wird ähnlich wie bei der Geodaten-Ermittlung über eine Anfrage an die Route Matrix von Mapquest gestartet. Diese Anfrage berechnet die Fahrzeit und die Distanz zwischen beliebig vielen angegebenen *Punkten*. Als *Punkte* sind die aus dem vorherigen Prozess ermittelten Geodaten zu verstehen. Aus der Datenbank werden nun die Geodaten der aktuellen Anmeldung eines Teilnehmers als Startpunkt der Anfrage genutzt. Als Zielpunkt werden die Geodaten eines anderen Teilnehmers verwendet, zu welchem die Distanz berechnet werden soll.

Als Antwort erhält die Anwendung, genau wie bei der Geocoding Anfrage, eine JSON-Ausgabe, die sowohl die Fahrdistanz in Meilen, wie auch die Fahrzeit in Sekunden beinhaltet. Diese Ausgabe wird erneut in ein Objekt umgewandelt und es wird auf die Fahrzeit zugegriffen. Die Fahrzeit wird im Anschluss in die Datenbank gespeichert.

Dieser Prozess muss für alle in der Teilnehmerliste vorkommenden Teilnehmer wieder-

⁶<https://developers.google.com/maps/licensing>

⁷<http://developer.mapquest.com/web/products/open/directions-service>

The screenshot shows a web browser window with the URL `maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&sensor=false`. The page displays the JSON response from the Google Maps API. The address is highlighted in blue at the top right. The JSON structure is as follows:

```

{
  "results" : [
    {
      "address_components" : [
        {
          "long_name" : "1600",
          "short_name" : "1600",
          "types" : [ "street_number" ]
        },
        {
          "long_name" : "Amphitheatre Parkway",
          "short_name" : "Amphitheatre Pkwy",
          "types" : [ "route" ]
        },
        {
          "long_name" : "Mountain View",
          "short_name" : "Mountain View",
          "types" : [ "locality", "political" ]
        },
        {
          "long_name" : "Santa Clara County",
          "short_name" : "Santa Clara County",
          "types" : [ "administrative_area_level_2", "political" ]
        },
        {
          "long_name" : "Kalifornien",
          "short_name" : "CA",
          "types" : [ "administrative_area_level_1", "political" ]
        },
        {
          "long_name" : "USA",
          "short_name" : "US",
          "types" : [ "country", "political" ]
        },
        {
          "long_name" : "94043",
          "short_name" : "94043",
          "types" : [ "postal_code" ]
        }
      ],
      "formatted_address" : "1600 Amphitheatre Parkway, Mountain View, Kalifornien 94043, USA",
      "geometry" : {
        "location" : {
          "lat" : 37.4219998,
          "lng" : -122.0839596
        },
        "location_type" : "ROOFTOP",
        "viewport" : {
          "northeast" : {
            "lat" : 37.4233487802915,
            "lng" : -122.0826106197085
          },
          "southwest" : {
            "lat" : 37.4206508197085,
            "lng" : -122.0853085802915
          }
        }
      },
      "types" : [ "street_address" ]
    }
  ]
}

```

Annotations in the image include:

- A blue box around the address in the URL bar.
- The word "Adresse" in blue text above the address components.
- A blue box around the "location" object in the "geometry" field.
- The text "Geodaten: Höhen- und Breitengrad" in blue text next to the "location" object.

Abbildung 22: Die JSON-Ausgabe der Geocoding Anfrage für die Bsp. Adresse „1600 Amphitheatre Parkway, Mountain View, CA“.

holt werden. Denn für einen sich neu anmeldenden Teilnehmer müssen alle Fahrdistanzen zu den bisherig angemeldeten Teilnehmern ermittelt werden. Der Startpunkt, welcher sich aus den Geodaten des aktuellen Teilnehmers ergibt, bleibt beim mehrfachen ausführen dieses Prozesses unverändert. Während aus der Teilnehmerliste bei jedem Iterationsschritt des Prozesses neue Geodaten als Zielpunkt angefordert werden müssen. Falls ein Teilnehmer zwischen sich und allen anderen Teilnehmern eine Fahrzeit berechnet hat, ist die Anmeldung abgeschlossen.

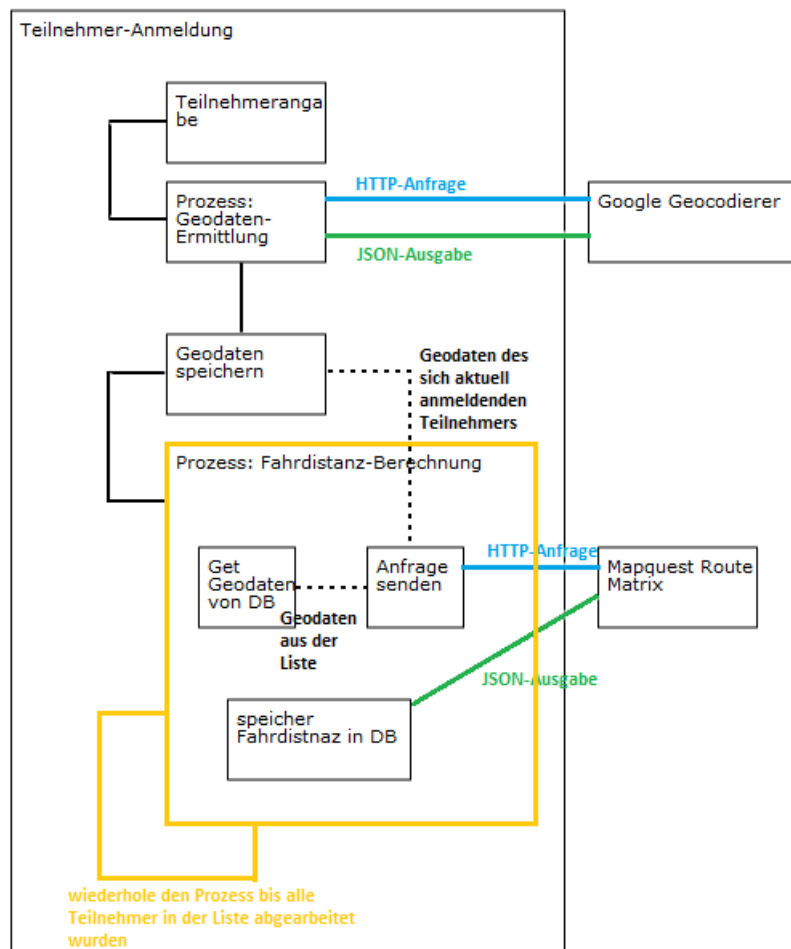


Abbildung 23: Model eines Ablauf der Teilnehmer-Anmeldung.

4.2.4 Anmerkungen zur Teilnehmer-Anmeldung

Die Prozesse Geodaten-Ermittlung und Fahrdistanz-Berechnung machen aus der einfachen Teilnehmer-Anmeldung einen zeitintensiven Ablauf komplizierter Teilschritte. Hinzu kommt, dass diese Prozesse für die Anmeldung überhaupt keine Rolle spielen, denn

die Angaben, welche an dieser Stelle berechnet werden, kommen erst bei der Teilnehmer-Zuordnung zum Einsatz. Trotz dieser negativen Aspekte lohnt es sich für die Anwendung, diese Prozesse bei der Teilnehmer-Anmeldung auszuführen.

Vorteil 1: Reduzierung unnötiger HTTP-Anfragen

Die zeitaufwendigen HTTP-Anfragen müssen nicht während der Teilnehmer-Zuordnung ausgeführt werden. Die Resultate liegen bereits gespeichert in der Datenbank vor und können dort abgefragt werden. Somit läuft die Anwendung zum einen schneller, da ein Datenbank-Aufruf performanter ist, als eine HTTP-Anfrage. Zum anderen läuft die Anwendung stabiler, da es passieren kann, dass die Google Server oder Mapquest Server überlastet bzw. nicht erreichbar sind. Dies würde sich bei der Anmeldung nicht ganz so negativ auswirken wie bei der Zuordnung.

Vorteil 2: Höhere Effizienz bei der Fahrdistanz-Berechnung

Da nicht vorhergesagt werden kann welche Teilnehmer bei der Zuordnung zusammengesetzt werden, ist es sinnvoll, alle möglichen Kombinationen zu berücksichtigen. Bei einer Teilnehmeranzahl von n Teilnehmern müssen $n - 1$ Fahrzeiten berechnet und gespeichert werden, da der Abstand eines Teilnehmers zu sich selbst nicht berechnet wird.

Somit steigt bei wachsender Teilnehmerzahl auch die Anmeldedauer, da immer mehr Fahrzeiten zwischen den wachsenden Teilnehmer-Kombinationen berechnet werden müssen. Durch eine Parallelisierung des Prozesses lässt sich die Anmeldedauer jedoch rapide senken.

Vorteil 3: Bessere Verteilung der HTTP-Anfragen

Die HTTP-Anfragen werden an die Anmeldung gekoppelt. Da es sehr unwahrscheinlich ist, dass sich alle Teilnehmer zum selben Zeitpunkt anmelden, kann man davon ausgehen, dass nicht zu viele Anfragen auf einmal an den Geocodierer oder an die Route Matrix gesendet werden. Dadurch wird verhindert, dass die Anfragen als Spam kategorisiert werden und die Google Geocoding API keine Antworten zurück sendet.

Anstelle die Teilnehmer-Zuordnung mit den Berechnungen der Geodaten und Fahrdistanzen weiter zu belasten werden diese Prozesse jedem Teilnehmer angehängt. Somit wird die zeitintensive Belastung der Prozesse auf alle Teilnehmer mehr oder weniger verteilt. Die ersten Teilnehmer die sich anmelden haben also eine geringere Wartezeit als jene, die sich später anmelden.

4.3 Beschreibung Instruktionen drucken

Kein Teilnehmer soll **vor** dem Beginn des Staffelessens im Bilde darüber sein, wo er als Gast eingeladen ist oder welche Gäste ihm selber zugeteilt wurden. Diese Informationen erhält der Teilnehmer während des Verlaufs des Staffelessens über die ihm zugeteilten *Instruktionen*.

Die Instruktionen werden aus der Gästeliste generiert, welche in der Datenbank gespeichert wurde und werden vom Organisator an die einzelnen Teilnehmer verteilt. Die Instruktionen teilen sich dabei in Gastgeber-Instruktionen und Gäste-Instruktionen auf. Somit gibt es eine Instruktion für den Gast und eine separate Instruktion für den Gastgeber.

Gastgeber-Instruktionen

Diese Instruktionen enthalten Informationen die ein Gastgeber benötigt um sein Gericht auszurichten. Die Informationen beziehen sich auf die Gäste und deren Eigenschaften. So wird dem Gastgeber z.B bekannt gegeben wie viele Gäste er zu erwarten hat und ob es sich bei seinen Gästen um Vegetarier handelt oder nicht. Zusätzlich enthalten die Instruktionen die Kontaktdaten des Organisators falls Rückfragen bestehen.

Gäste-Instruktionen

Die Instruktionen für die Gäste geben Aufschluss über deren **nächsten** Gastgeber. Die Gäste-Instruktionen werden der Gastgeber-Instruktion beigelegt sodass die Gäste, nachdem sie ihr Gericht beim Gastgeber eingenommen haben erfahren, zu welchem Gastgeber sie als nächstes eingeladen sind. Sollte der Gast im folgendem Gang selber ein Gastgeber sein erhält dieser Gast keine Instruktion, da dies überflüssig ist.

Tabelle 4: Inhalt der einzelnen Instruktionen.

	Gastgeber-Instruktion	Gäste-Instruktion
Information	Name seiner Gäste	Name seines Gastgebers
	Anzahl der Gäste	Adresse des Gastgebers
	Vegetarier werden markiert	Kontaktdaten des Gastgebers
	Kontaktdaten des Organisators	Kontaktdaten des Organisators

4.3.1 Aufbau der Instruktionen

Der Inhalt der Instruktionen wurde nach den bestehenden Kenntnissen früherer Staffelessen beschlossen. Zum Teil wurden die Instruktionen aber auch Optimiert, da es bei früheren Events häufiger zu Problemen kam. In den Aufbau der Instruktionen flossen ebenfalls die Erfahrungen und Kenntnisse des Kunden mit ein.

Die Verbesserungen beziehen sich hauptsächlich auf das mitteilen der Kontaktdaten des Gastgebers und des Organisators. Häufig haben sich die Gäste nämlich verfahren oder

haben das richtige Haus einfach nicht gefunden. Deswegen war es erforderlich, die Kommunikation der Teilnehmer untereinander zu verbessern.

4.3.2 Anmerkungen zum Use Case Instruktionen drucken

Es wäre zeitlich nicht möglich gewesen sich in die PDF-Generierung in PHP einzuarbeiten. Deswegen wurde die Implementierung des Skriptes, welches für diese Funktionalität verantwortlich ist, an andere Mitarbeiter innerhalb der nova GmbH vergeben. Die Mitarbeiter waren mit der Aufgabe betraut, ein einfach zu integrierendes Script zu realisieren, welches die Funktionalität des Use Cases erfüllt und der Beschreibungen des Konzept entsprechen sollte. Anschließend sollte das Script lediglich vom Entwickler des Projektes während der Implementierungsphase eingebunden werden.

Da die Implementierung dieses Features nicht selbständig durchgeführt wurde, wird sie in den folgenden Kapiteln ebenfalls nicht erwähnt. Aus diesem Grund wird diese Anmerkung auch in diesem Kapitel vermerkt.

4.4 Teilnehmer-Editierung

Die Teilnehmer-Editierung soll es dem Organisator ermöglichen, die Teilnehmerliste und die Angaben der Teilnehmer zu bearbeiten oder sie komplett zu löschen. Hierbei soll der Organisator so unterstützt werden, dass er Änderungen schnell und effizient durchführen kann. Die Editierung bezieht sich dabei immer auf die Datenbank. Da die Informationen von der Datenbank geliefert werden und die Änderungen direkt die Datenbank verändern, ist es wichtig, dass die Anzeige der Daten sowie die Skripte, welche die Editierung letzten Endes durchführen, konsistent mit der Datenbank sind.

In diesem Kapitel wird nicht auf das Layout oder auf die Teilnehmerinformationen, welche geändert werden, eingegangen. Vielmehr soll das Kapitel beschreiben, welche Funktionalität die Teilnehmer-Editierung bereitstellen sollte und welche Risiken zu beachten sind. Das Layout wird im Kapitel Screendesign behandelt.

4.4.1 Darstellung der Teilnehmer-Editierung

Auf einer eigenen HTML-Seite wird die komplette Teilnehmerliste mit allen nötigen Informationen des Teilnehmers dargestellt. Betätigt der Organisator nun den *edit*-Button (Abb.24) eines Teilnehmers, wird dieser auf eine weitere HTML-Seite geleitet, welche das Formular zur Änderung dieses spezifischen Teilnehmers enthält.

Das Formular entspricht dem der Teilnehmer-Anmeldung mit dem Unterschied, dass die Form-Felder bereits mit den vorhandenen Daten des Teilnehmers aus der Datenbank gefüllt sind. Dadurch lassen sich kleinere Änderungen effizienter durchführen und bei größeren Änderungen verliert der Organisator nichts von seiner Effektivität.

Betätigt der Organisator den *delete*-Button, wird der Teilnehmer in der gesamten Datenbank gelöscht. Da diese Informationen unwiderruflich entfernt werden, muss das Löschen

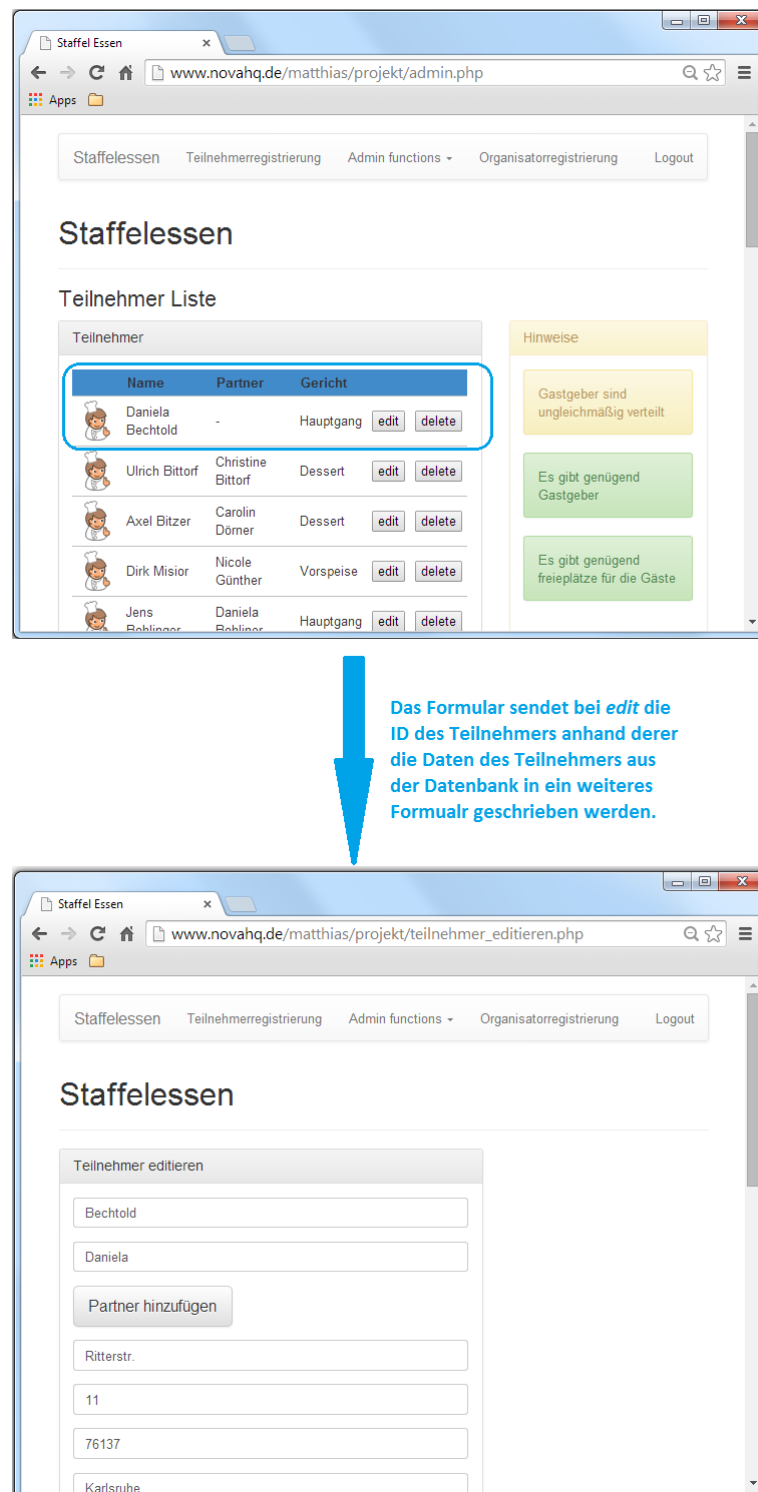


Abbildung 24: Die Darstellung der Teilnehmerliste und das Formular.

eines Teilnehmers über ein weiteres Dialog-Fenster bestätigt werden. Nachdem der Dialog bestätigt (oder abgebrochen) wurde, wird die Seite mit der Teilnehmerliste aktualisiert.

Somit wird die Tabelle anhand der editierten Datenbank neu dargestellt. Bei den beiden dargestellten Buttons handelt es sich um Elemente zweier Formulare. Diese Formulare enthalten jeweils das selbe *hidden*-Field, welches beim aufbauen der Teilnehmerliste mit der ID des jeweiligen Teilnehmers überschrieben wird. Anhand der ID werden nun beim abschicken der Formulare die Informationen des entsprechenden Teilnehmers bearbeitet oder gelöscht.

4.4.2 Manipulation der Datenbank

Die Datenbank muss beim editieren eines Teilnehmers unter Umständen neuen Inhalt hinzufügen(*INSERT*), alten Inhalt verändern(*UPDATE*) oder sogar Inhalt aus der Datenbank löschen(*DELETE*). In diesem Kapitel sollen die dadurch entstehenden und zu beachtenden Szenarien beschrieben werden.

Partner hinzufügen bzw. entfernen

Es könnte nötig sein, einem editierten Teilnehmer der zum Anmeldezeitpunkt ein Single war, einen Partner zuzuweisen. In diesem speziellen Fall müsste ein neuer Teilnehmer in die Datenbank eingefügt werden und dieser müsste mit dem editierten Teilnehmer verknüpft werden. Da sich Paare ihre Kontaktdaten teilen, wäre es einfach möglich, die Informationen des editierten Teilnehmers abzufragen und beim einfügen des Partners zu verwenden.

Bei der Verknüpfung des editierten Teilnehmers zu seinem Partner werden die jeweiligen IDs der Teilnehmer in Relation zueinander gesetzt. Somit erhält der editierte Teilnehmer die ID des neuen Teilnehmers als Partner ID, während der neue Teilnehmer die entsprechende ID des editierten Teilnehmers bereits als Partner ID besitzt, bevor dieser in die Datenbank gespeichert wird.

Beim entfernen eines Partners wird dessen Datensatz komplett aus der Datenbank gelöscht. Anschließend wird die Partner ID des editierten Teilnehmers auf 0 gesetzt um seine Verknüpfung mit dem Partner aufzulösen.

Gastgeber in einen Gast umschreiben bzw. Umgekehrt

Falls der Organisator vorhat einen Gastgeber in einen Gast umzuschreiben, muss darauf geachtet werden, dass innerhalb der Datenbank zwei separate Datenbanktabellen manipuliert werden müssen. Zum einen wird in der Teilnehmer-Tabelle das Attribut eines editierten Teilnehmers und seines Partners, welches symbolisiert ob der Teilnehmer Gast oder Gastgeber ist, entsprechend gesetzt(entweder auf 0 für Gast oder auf 1 für Gastgeber).

Zum anderen wird in der Gerichte-Tabelle, welche die Gerichte den einzelnen Gastgebern zuweist, ein *DELETE* durchgeführt, um den Datensatz des editierten Teilnehmers in dieser Tabelle zu löschen. Dies wird deswegen getan, um die Anzahl der Gastgeber zu reduzieren, die sich aus der Gerichtsanzahl ergibt. Da ein Gast logischerweise nicht als Gastgeber zählt, ist es also gerechtfertigt, den kompletten Datensatz der Gerichte-Tabelle zu entfernen.

Umgekehrt gilt die selbe Reihenfolge. in der Teilnehmer-Tabelle wird das Attribut für den Gastgeber mit einem *UPDATE* entsprechend gesetzt. Anschließend wird in die Gerichte-Tabelle ein neuer Datensatz eingefügt, welcher die Angaben des Formulars enthält, das abgesendet wurde.

Adresse des Teilnehmers manipulieren

Da die Geodaten (Höhen- und Breitengrade) sowie die Fahrzeiten eines Teilnehmers für die Anwendung essentiell sind, muss für jeden editierten Teilnehmer geprüft werden, ob sich die Geodaten und die daraus ergebenden Fahrzeiten verändern. Deshalb werden die Geodaten aus der übergebenen Adresse des editierten Teilnehmers neu berechnet. Nun können die neuen Geodaten mit den alten Geodaten verglichen werden. Weichen diese voneinander ab, so müssen die alten Fahrzeiten des editierten Teilnehmers aus der Datenbank gelöscht werden. Es werden neue Fahrzeiten aus den neuen Geodaten berechnet. Diese werden anschließend in die Datenbank eingetragen.

Da die Berechnung der Fahrzeiten ein zeitintensiver Prozess ist, wird die Teilnehmer-Editierung um eine extra Seite erweitert, welche einen Ladebalken anzeigen soll (diese kann einfach von der Anmeldeseite übernommen werden). Die Berechnung der Fahrzeiten sowie das eintragen in die Datenbank wird dabei vom selben Script ausgeführt welches auch in der Teilnehmer-Anmeldung zum Einsatz kommt.

Verbleibende Daten anpassen

Die verbleibenden Daten, welche editiert werden, benötigen keinerlei spezielles vorgehen. Die Daten wie z.B der Name oder die Wahl des Gerichtes werden immer automatisch mit einem *UPDATE* innerhalb der Datenbank mit den gesendeten Formularinformationen überschrieben. Selbst wenn keine Änderungen vorgenommen wurden.

Es würde sich nicht lohnen die Daten auf Veränderungen zu prüfen, da dies nur weitere Datenbank abfragen und somit mehr Aufwand für die Datenbank bedeuten würde. Um zu verhindern das Invalide Daten in die Datenbank eingefügt werden oder die Daten innerhalb der Datenbank falsch verändert werden, wird die Validierung vor dem Absenden des Formulars auf Validität überprüft. Hierfür können wieder die selben Funktionen wie bei der Teilnehmer-Anmeldung genutzt werden.

4.4.3 Änderungen der Datenbank festhalten und Ausgeben

Da sich die Änderungen auf die Datenbank beziehen wird ebenfalls die gespeicherte Gästeliste, welche sich aus den Informationen der Datenbank ergibt, abgeändert. Somit ist es erforderlich dem Organisator mitzuteilen welcher Teilnehmer editiert wurde, um die Änderungen auch bei der in der Datenbank vorhandenen Gästeliste besser zu visualisieren.

Die Visualisierung der Gästeliste soll dabei so modifiziert werden, dass ein editierter Teilnehmer einen roten Hintergrund in der Tabellendarstellung erhält. Zusätzlich wird ein Warnhinweis eingeblendet, welcher den Organisator darauf hinweist, dass es Änderungen innerhalb der vorhandenen Gästeliste gegeben hat. Es wird dann dem Organisator emp-

fohlen, eine neue Zuordnung zu starten.

Um diesen Vorgang zu realisieren wird die Datenbank erweitert. Zusätzlich zu den Änderungen der Datenbank wird nun auch die ID in einer erweiterten Tabelle der Datenbank eingefügt, um beim prüfen der Gästeliste auf diese IDs zugreifen zu können.

5 Screendesign

Im Screendesign bzw. der Entwurfsphase wird erarbeitet wie die einzelnen Unterseiten der Webanwendung zu gestalten sind. Für die Gestaltung der Benutzerschnittstelle (User Interface UI) werden nun die im Konzept erarbeiteten Elemente der Anwendung in einer für den Benutzer optimalen und intuitiv verständlichen Art und Weise dargestellt und auf Unterseiten der Anwendung verteilt. Gleichzeitig ergibt sich in diesem Prozess die Sitemap.

5.1 Allgemeines Seiten-Layout

Mit Hilfe von Bootstrap werden die einzelnen Elemente nun in ein konsistentes Design gebracht. Die Einheitlichkeit innerhalb der Anwendung und damit auch die Orientierung für den Anwender wird so sichergestellt. Dadurch ist es nicht zwingend notwendig mit einem individuellen Designentwurf zu arbeiten, da das Erscheinungsbild durch den bloßen semantisch sinnvollen Einsatz von Bootstrap professionell aussieht.

Da bei diesem Projekt, mit diesem Wissen im Hintergrund, kein Budget für das Screendesign vorgesehen ist, beschränkt sich das Design komplett auf die Nutzung des Standardtemplates von Bootstrap und es erfolgen keine weiteren Anpassungen. Es wird also Zeit gespart, da das individuelle Entwerfen von HTML-Elementen entfällt und ebenfalls ein *Standard* eingeführt wird, der die „groben“ Entwurfsfragen bereits beantwortet. So müssen die Entwickler nur noch über die Position der einzelnen Elemente entscheiden.

5.1.1 Strukturieren der Unterseiten

Als Layout-Rahmen wird jede Unterseite in die einzelnen Teile *header*, *content* und *footer* aufgeteilt. Jede Unterseite ist so aufgebaut, dass sie einen gemeinsamen Header-Rahmen und Footer-Rahmen teilen, während der Content-Rahmen von Unterseite zu Unterseite variiert. Während der Footer im momentanen Stand des Projektes noch keinerlei Funktionen enthält, außer seiner Darstellung, sind die anderen beiden Rahmen mit verschiedenen Operationen versehen. Der Header-Rahmen enthält die Navigationsleiste der Webanwendung. Über sie navigiert der Benutzer (momentan hauptsächlich der Organisator) innerhalb der Anwendung und erhält damit zugriff auf beinahe alle Unterseiten.

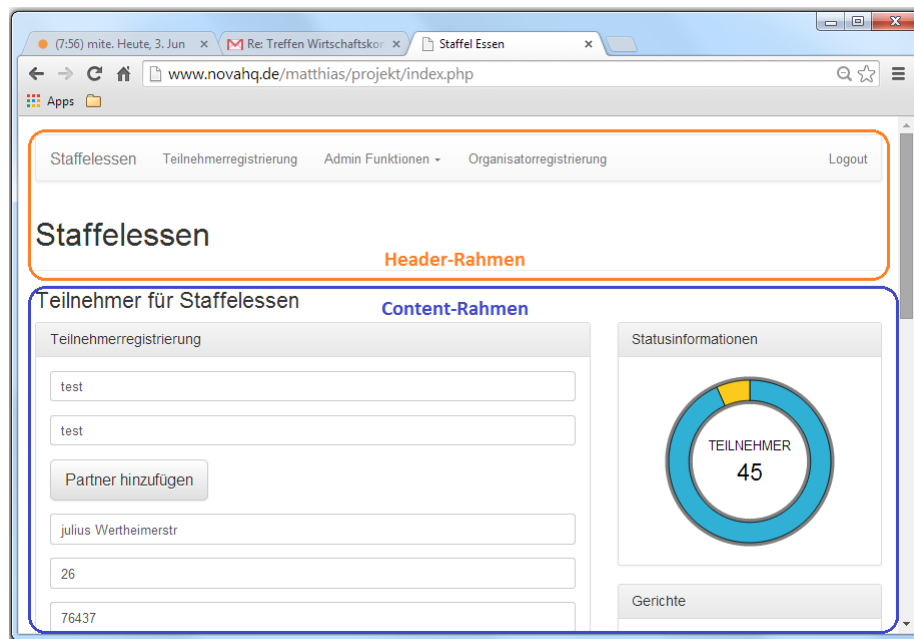


Abbildung 25: Die Darstellung der Header- bzw. Content-Rahmen.

Momentan lassen sich die Folgenden Unterseiten von der Navigationsleiste erreichen:

- Die Unterseite für die Teilnehmer-Anmeldung.
- Die Unterseite zum einloggen des Administrators.
- Die Unterseite für die Teilnehmer-Editierung.
- Die Unterseite zum erzeugen einer Gästeliste.
- Die Unterseite zum darstellen einer bereits gespeicherten Gästeliste.
- Die Unterseite zum generieren der Instruktionen.

Da sich die Navigationsleiste als Header-Rahmen auf jeder Unterseite befindet, ist es dem Organisator möglich, zwischen allen aufgeführten Unterseiten zu navigieren und keinerlei Reihenfolge beachten zu müssen. Neben der Navigationsleiste enthält der Header-Rahmen auch alle Meta-Daten die für die Unterseiten essenziell sind.

Im Gegensatz zum Header- und Footer-Rahmen wird der Content-Rahmen individualisiert gestaltet. Dadurch erhält jede Unterseite seine eigenen Operationen, die sich aus der Konzeptplanung ergeben. Der Content-Rahmen muss die Informationen, welche für seine Funktionalität von Bedeutung sind, aus der Datenbank anfragen und diese verarbeiten und dementsprechend visualisieren, um den Benutzer das gewünschte Feedback geben zu können. Die genaueren Angaben zur Funktionalität und den zugehörigen Unterseiten obliegt der Sitemap.

5.2 Die Sitemap

Die Sitemap oder auch Seitenübersicht zeigt die vollständige hierarchische Struktur einer Webpräsenz (Website oder Webanwendung) an. Die Unterseiten einer solchen Webpräsenz können dadurch verständlicher optimiert bzw. angepasst werden. Die in Abb.26 dargestellte Grafik zeigt die Sitemap für den Prototypen. Die Sitemap stellt alle Unterseiten und deren Beziehungen zueinander in abstrakter Form dar.

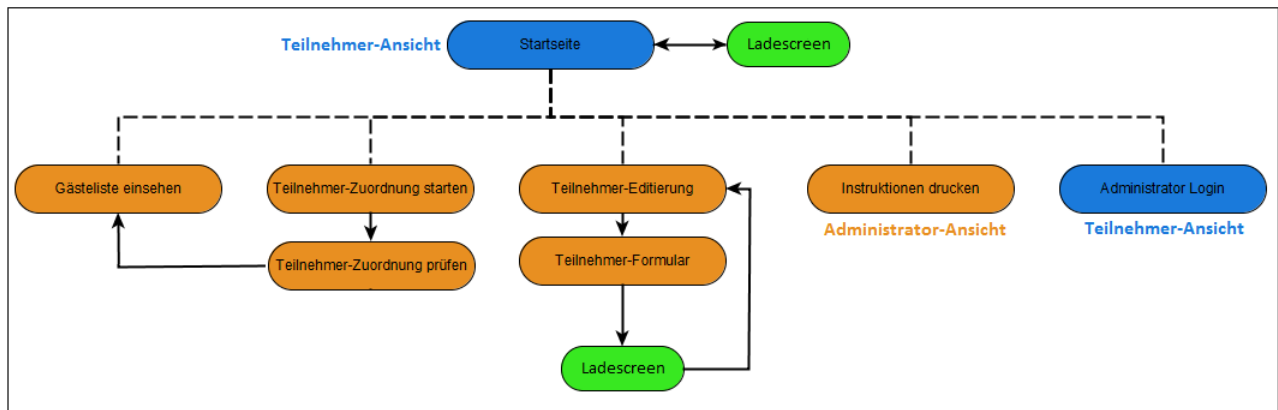


Abbildung 26: Die Sitemap der Webanwendung.

Die in Abb.26 aufgeführten Unterseiten *Startseite* und *Administrator Login* sind in der Darstellung blau markiert um zu symbolisieren, dass jeder Besucher bzw. Teilnehmer der Webanwendung diese Unterseiten betreten kann. Die mit Orange gekennzeichneten Unterseiten können lediglich vom Administrator eingesehen werden und deren Funktionen werden ebenfalls nur von ihm ausgeführt. Die gestrichelten Linien symbolisieren die Navigation der Webanwendung und die Pfeile stehen für das Erreichen einer Unterseite über das Ausführen einer bestimmten Funktion. Zum Schluss werden die Grün markierten Unterseiten erklärt. Bei ihnen handelt es sich um eine Unterseite, die bei längeren Wartezeit einen Fortschrittsbalken bzw. Ladebalken anzeigen soll.

Im Folgendem werden die Inhalte der einzelnen Unterseiten beschrieben. Dabei wird lediglich auf deren Arbeitsweise hingewiesen, da die Seitenfunktionalität bereits ausführlich in der Konzeptplanung besprochen wurde. Dieses Kapitel führt auf, wie die Webanwendungen mit grafischen Mitteln unterstützt werden soll. Aber zuerst wird noch in der Tabelle 5 verständlich dargelegt, welche Use Cases und Anforderungen den einzelnen Unterseiten der Sitemap zugeordnet wurden.

5.2.1 Use Cases und Anforderungen in der Sitemap

Tabelle 5: Sitemap Zurodnung der Use Cases und Anforderungen.

Sitemap	Use Case	Anforderung
Startseite	UC 1: Eintragen in die Teilnehmerliste	FA 1: Teilnehmer-Anmeldung FA 2: Partner hinzufügen FA 3: Teilnehmer Vegetarier FA 4: Gastgeber Option FA 5: Option Gericht
Administrator Login	UC 2: Administrator Login	FA 6: Admin Login
Teilnehmer Zuordnung starten	UC 3: Teilnehmer Zuordnung vorbereiten UC 4: Teilnehmer-Zuordnung starten	FA 11: Starte Zuordnung FA 12: Teilnehmer verteilen FA 13: Begegnungen prüfen FA 14: Zuordnung Vegetarier FA 15: Zuordnung Fahrdauer
Teilnehmer-Editierung	UC 5: Teilnehmer editieren	FA 7: selektiere Teilnehmer FA 8: editiere Teilnehmer FA 9: lösche Teilnehmer FA 10: speichere Änderungen FA 18: sortiere Liste
Gästeliste prüfen	UC 6: Gästeliste speichern	FA 16: Gästeliste speichern
Instruktionen drucken	UC 7: Instruktionen drucken	FA 17: Instruktionen drucken
Gästeliste ansehen		

5.2.2 Startseite

Die Startseite ist die erste Seite der Webanwendung die dem Benutzer dargestellt wird. Da es sich bei den Besuchern häufiger um Teilnehmer als um den Organisator handelt, ist es logisch, die Teilnehmer-Anmeldung auf dieser Seite zu verschieben. Die Unterseite besteht aus einem Formular für die Angaben des Teilnehmers und aus einer *Sidebar*. Die Sidebar enthält nun *Teilnehmerzahlen* oder die Aufteilung der Gerichte (Abb.27). Diese werden für den Teilnehmer anhand von *Progress-Bars*, welche durch das Bootstrap Framework zu Verfügung stehen, angezeigt.

5.2.3 Administrator Login

Die Unterseite welche für die Administrator Anmeldung verantwortlich ist wird über die Navigationsleiste unter *Login* erreicht. Die Seite soll lediglich über ein Formular verfügen, welches einen Benutzernamen und ein Passwort abfragt. Meldet sich der Teilnehmer korrekt an, so wird ein Session-Attribut gesetzt, welches den Teilnehmer nun als einen Organisator auszeichnet solange die Session läuft. Die Navigationsleiste eines Organisators erweitert sich dann um die *Admin Funktionen* und die *Organisatorregistrierung* (Abb.28).

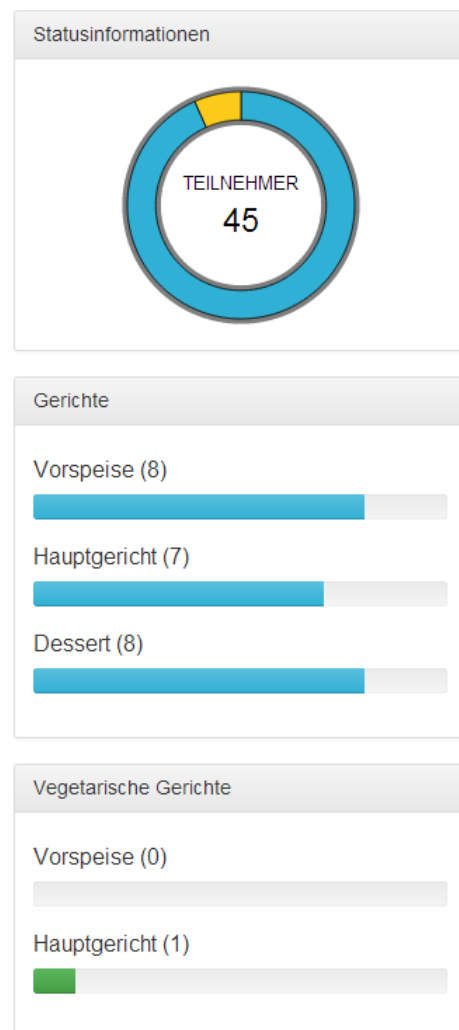


Abbildung 27: Die Anmeldungs Sidebar.

5.2.4 Admin Funktion 1: Gästeliste generieren

Die Unterseite Gästeliste generieren ist ein über das Dropdown-Menü *Admin Funktionen* erreichbarer Wizard, welcher den Komplexen und aufwändigen Prozess der Teilnehmer-Zuordnung übernimmt und daraus eine passende Gästeliste generiert. Wie bereits in der Konzeptplanung festgestellt wurde, ist die Teilnehmer-Zuordnung der Anwendung sehr vielschichtig. Deswegen wurde diese komplexe Aufgabe auf drei Unterseiten aufgeteilt.

Die erste Unterseite ist die *Teilnehmer-Zuordnung starten*. Diese Unterseite soll die Generierung der Gästeliste vorbereiten, indem drei Listen mit den Gastgebern der unterschiedlichen Gänge aufgeführt werden. Diese sind mit einer Jquery Sortable-List realisiert, welche es dem Administrator erlaubt, per Drag and Drop, Gastgeber zwischen den Gängen zu

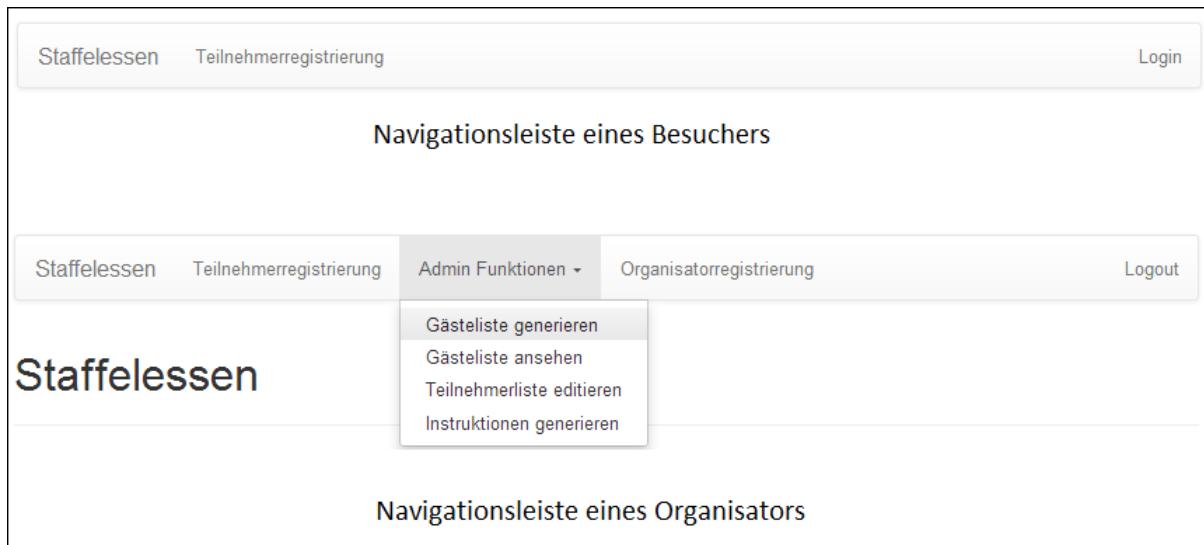


Abbildung 28: Die unterschiedlichen Navigationsleisten im direkten Vergleich.

verschieben (Abb.29). Dadurch kann der Organisator die Listen bei Ungleichmäßigkeiten wieder ausgleichen. Sind generell nicht genügend Gastgeber angemeldet oder ist es nicht möglich, in der Liste eine Gleichmäßigkeit zu erzeugen, so wird durch das Sperren des *zuordnen*-Buttons verhindert, dass eine Gästeliste generiert werden kann.

Auf der zweiten Unterseite des Wizards wird nun eine berechnete Gästeliste modelliert. Diese Berechnung kann einige Sekunden dauern. Deswegen wird über animierte *Ladezeichen* dem Organisator ein Feedback der Seite übermittelt. Die darauffolgende Gästeliste wird auf die selbe Art und Weise modelliert wie eine aus der Datenbank geladene Gästeliste. Der Organisator sollte diesen Unterschied auch nicht merken, da es für ihn nicht relevant ist, ob die dargestellte Gästeliste berechnet oder aus der Datenbank abgefragt wurde. Die Gästeliste kann auf dieser Unterseite noch einmal inspiziert werden, bevor sie im nächsten Schritt in die Datenbank gespeichert wird.

Wird auf dieser Unterseite nun der *speicher*-Button betätigt, wird die Gästeliste in die Datenbank transferiert und es wird automatisch auf die Unterseite *Gästeliste ansehen* weitergeleitet.

5.2.5 Admin Funktion 2: Gästeliste ansehen

Diese Unterseite stellt die gespeicherte Gästeliste über eine Schnittstelle mit der Datenbank dar. Die Sidebar auf dieser Unterseite weist den Organisator auf *Konflikte* hin. Ein Konflikt entsteht falls die Teilnehmer nach dem generieren der Gästeliste editiert werden (Abb.30). Zusätzlich kann der Organisator die Unterseite *Instruktionen drucken* aufrufen um dessen Funktion auszuführen.

Gastgeberlisten

Gastgeber für die Vorspeise	Gastgeber für den Hauptgang	Gastgeber für das Dessert
Stefan Koch	Kai Keune	Alexander Derler
Alexander Murawski	Ralf Klein	Adrian Hönig
Yvonne Rosen-König	Stefan Klocke	Thorsten Müller
Torsten Seeger	Yvonne Niederkrome	Nicole Munk
Martina Stoppanski	Tino Scraback	Martin Rosenberg
Arno Strecker	Matthias Misior	Claudia Schmitt
Nicole Tinsz	Vicky Rauh	Andreas Sütterlin
	Carola Schmidt	Volkmar Triebel
	Sandra Walzer	

Abbildung 29: Die Sortable-List der Unterseite *Teilnehmer-Zuordnung* starten.

Vorspeise

STOPPANSKI Martina						
KOCH Stefan						

Konflikte

Seit der letzten Zuordnungsberechnung haben Sie Teilnehmer editiert.

Das kann möglicherweise zu Konflikten führen. Die betroffenen Teilnehmer sind **rot** markiert.

[Neue Zuordnung starten](#)

Abbildung 30: Darstellung eines Konfliktes auf der Unterseite *Gästeliste ansehen*.

5.2.6 Admin Funktion 3: Teilnehmer-Editierung

Für die Teilnehmer-Editierung werden die Teilnehmer anhand einer Tabelle aufgeführt, die aus der Datenbank angefragt wird. Es werden nicht alle Informationen des Teilnehmers in dieser Tabelle aufgeführt, da diese sonst zu unübersichtlich wäre. Stattdessen werden die Teilnehmerinformationen auf zwei Unterseiten aufgeteilt.

Auf der Unterseite, die in der Sitemap als *Teilnehmer-Editierung* bezeichnet wird, werden die groben Angaben eines Teilnehmers wie beispielsweise sein Name oder sein Gericht in einer Liste dargestellt. Diese Liste enthält alle bisher angemeldeten Teilnehmer (Abb.24). Die zweite Unterseite mit dem Namen *Teilnehmer-Formular* kann als Detailseite angesehen werden, da sie alle Angaben des Teilnehmers als ausgefülltes Formular wiedergibt. Alle Änderungen, die den Teilnehmer betreffen, können ausschließlich in diesem Formular vorgenommen werden.

Um den Organisator beim Editieren zu unterstützen, wird in die *Teilnehmer-Editierung* eine Sidebar eingefügt, welche die Kriterien zur Erzeugung einer Gästeliste überwacht und damit dem Organisator Hinweise gibt, wie die Teilnehmerliste noch zu optimieren wäre (Abb.31). Ist eines der Kriterien nicht erfüllt, so wird explizit ein Warnhinweis in der Sidebar erzeugt.

Die Kriterien Lauten

- Die Gerichte müssen gleichmäßig verteilt sein.
- Es müssen min. 21 Gastgeber angemeldet sein.
- Die Anzahl der Gäste darf höchstens ein Drittel der gesamten Gastgeberanzahl entsprechen.



Abbildung 31: Sidebar zur Optimierung der Teilnehmer-Editierung.

5.3 Responsive-Design

Wie bereits im Abschnitt zu Bootstrap (Kapitel 2.2.5) erklärt wurde, wird über das Bootstrap Framework ein Responsive-Design verwirklicht. In diesem Kapitel soll nochmals im Detail auf die Vorteile und Eigenschaften von Responsive-Design eingegangen werden.

Beim Thema Responsive-Design handelt es sich nicht einfach um einen Trend beim Webdesign, sondern um ein komplettes Umdenken bei der Planung und Entwicklung eines Entwurfs für eine Website. Das Ziel ist die Gestaltung von Websites, die sich an alle beliebigen Displaygrößen anpassen können. Ein Responsive-Design, welches ein flexibles Layout erlaubt, wird bei steigenden Absatzzahlen von Smartphones oder Tablets immer wichtiger. Laut der ARD/ZDF-Onlinestudie vom Jahr 2013 gehen nämlich 45 Prozent aller Deutschen über das Smartphone ins Internet⁸. Dies führt dazu, dass das Webdesign nicht mehr nur für Desktop-PCs, sondern auch für andere Endgeräte eine Rolle spielt.

Tab. 3 Genutzter Internetzugang 2013 nach Geschlecht und Alter in %							
	Gesamt	Frauen	Männer	14-29 J.	30-49 J.	50-69 J.	ab 70 J.
Computer bzw. PC/Laptop (netto)	96	97	96	98	97	95	93
Computer bzw. PC über einen Laptop	70	65	76	65	71	74	76
Smartphone (netto)	45	41	48	69	48	24	9
iPhone	17	18	17	23	20	11	7
anderes Smartphone	31	26	35	50	32	16	3
„normales“ Handy	5	4	5	5	4	5	2
Spielekonsole	9	5	12	18	9	1	2
MP3-Player	5	3	6	8	5	1	3
Fernseher	12	9	15	10	13	10	15
Tablet PC (netto)	16	15	16	14	20	13	5
iPad	9	10	8	9	11	8	4
anderer Tablet PC	7	6	8	6	9	6	1
E-Book-Reader	5	6	5	4	6	5	4
Ø Anzahl genutzter Geräte	2,5	2,3	2,7	3,0	2,7	2,1	1,9

Basis: Deutschspr. Onlinenutzer ab 14 Jahren (n=1 389).
Quelle: ARD/ZDF-Onlinestudie 2013.

Abbildung 32: Die Onlinestudie des ARD/ZDF.

Aus diesen Gründen wird auch bei der nova GmbH immer häufiger das Responsive-Design zum Gestalten und Entwickeln einer Website genutzt. Es mag aufwändiger und komplizierter als die Vorgehensweise bei einem herkömmlichen Webdesign sein, aber der Zukünftiger Nutzen einer Responsive-Website ist diesen Aufwand auf jeden Fall Wert.

⁸http://www.media-perspektiven.de/uploads/tx_mppublications/0708-2013_Eimeren.pdf

5.3.1 Grundlagen des Responsive-Design

Um eine Responsive-Website zu erstellen müssen bestimmte Voraussetzungen, die die Website betreffen, erfüllt werden. Zum einen erfordert ein Responsive-Design ein Flüssiges-Seitenlayout. Das bedeutet, dass das Layout sich bei Unterschiedlichen *Viewports* prozentual anpasst. Der Viewport ist dabei der Bereich, der für die Darstellung der Website zu Verfügung steht [?].

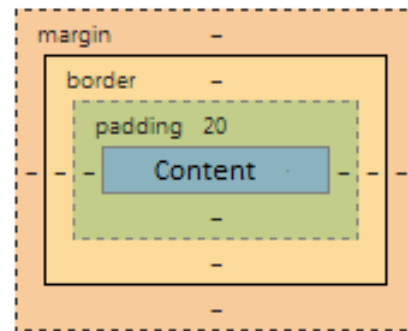


Abbildung 33: Das HTML Box Model.

Um das Layout fließend zu gestalten, müssen alle Angaben des Stylesheets die sich auf die Pixelgrößen der Elemente beziehen, in Prozent umgerechnet werden. Dabei muss darauf geachtet werden, dass die Breite eines Elementes sich bei Prozentwerten, genau wie auch bei Pixelwerten, aus dem *HTML Box Model* zusammenrechnet (Abb.33). Das Box Model besagt, dass sich die Breite eines HTML-Elementes, also dessen *Border*, aus der Breite des *Content* und der Breite des *Paddings* ergibt. Somit müssen also die Prozentwerte des *Contents* und die des *Paddings* addiert werden, um die vollständige Breite eines Elementes zu erhalten.

Das Fließende-Seitenlayout alleine macht aber noch kein Responsive-Webdesign möglich. Durch das Fließende-Layout werden zwar die Seiteninhalte der Displaygröße entsprechend angepasst, aber ab einem bestimmten Punkt kann der Inhalt einfach nicht sachgemäß wiedergegeben werden. Deswegen müssen für eine Responsive-Website auch die Seiteninhalte umstrukturiert werden, falls die Displaygröße zu klein wird.

Um dies zu bewerkstelligen werden sogenannte *Media Queries* eingesetzt. Media Queries erweitern das Stylesheet um alternative Anzeigen der HTML-Elemente. Der Media Query prüft, über welches Gerät (PC, Smartphone, Tablet etc.) die Website aufgerufen wird und über welche Breite der Viewport für dieses Gerät verfügt. Dadurch werden *Breakpoints* in das Design integriert. Unterschreitet der Viewport nun einen solchen Breakpoint, so wird das Layout nach den Angaben im Media Query neu strukturiert und der Viewport zeigt die optimierten Seiteninhalte an (Abb.34 und Abb.35).

Mit diesen beiden Techniken lässt sich nun eine Responsive-Website verwirklichen. Da die Media Queries und das Fließende-Layout bereits im Bootstrap Framework integriert sind, ist es nicht erforderlich, das Projekt wie oben beschrieben durchzuführen. Stattdessen werden diese Aspekte durch die Bootstrap-Klassen und durch die Bootstrap-Bibliothek abgedeckt.

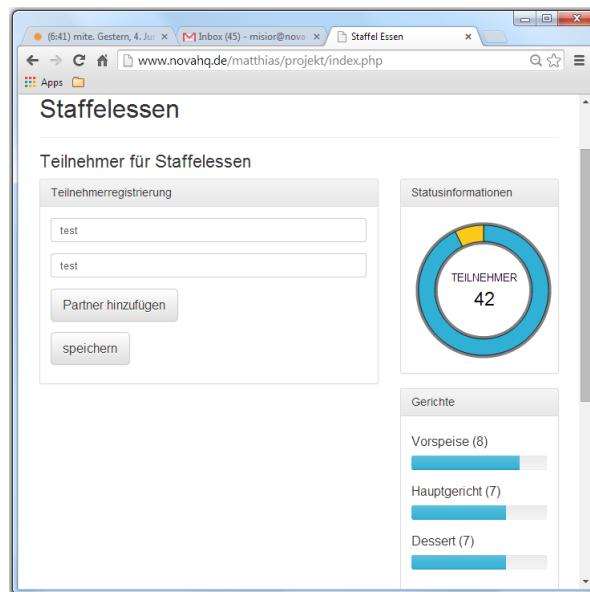


Abbildung 34: Viewport vor dem Unterschreiten des Breakpoints.

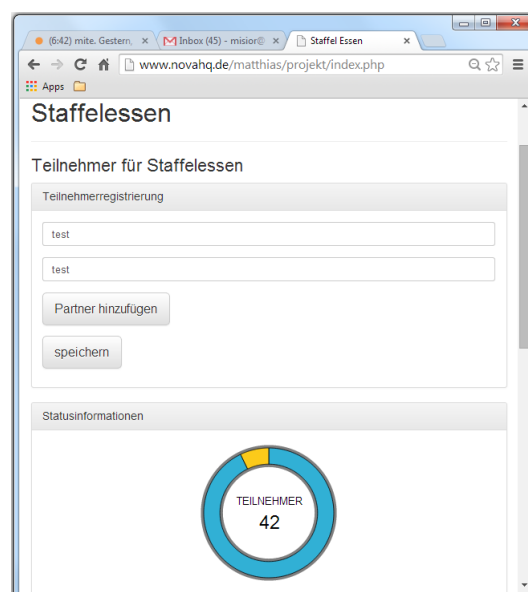


Abbildung 35: Viewport nachdem der Breakpoint unterschritten wird.

6 Implementierung

Die in der Konzeptplanung ermittelten Funktionsweisen und Features wurden in der Entwurfsphase (Screendesign) auf die unterschiedlichen Unterseiten der Anwendung mithilfe einer Sitemap aufgeteilt. Nun muss noch beschrieben werden, wie die Umsetzung des Konzeptes in Software erfolgte und wie die entstandenen Probleme behoben wurden. In diesem Kapitel wird dabei auf die verwendete Architektur der Webanwendung eingegangen sowie

auf die Klassen und Datenstrukturen der Datenbank. Es wird erläutert wie die Interaktion von Datenbank, Skripten und Klassen abläuft und weshalb diese Vorgehensweise im Bezug auf das Projekt von Vorteil ist.

6.1 System Architektur

Da es sich bei diesem Projekt um eine prototypische Webanwendung handelt, welche von einem Standardbrowser (bis einschließlich Internet Explorer Ver 8.0) ausgeführt wird, eignet sich das Model-View-Controller Prinzip (MVC) als Architekturmuster. Die Für dieses Muster eingeteilten Elemente lauten wie folgt. Als View gelten alle HTML-Seiten der Webanwendung, da diese mit dem Anwender interagieren und die Ergebnisse der Geschäftslogik grafisch wiedergeben. Die PHP-Skripte bilden das Model der Architektur ab. Sie führen die Berechnungen durch und sind meistens mit der Datenbank verbunden.

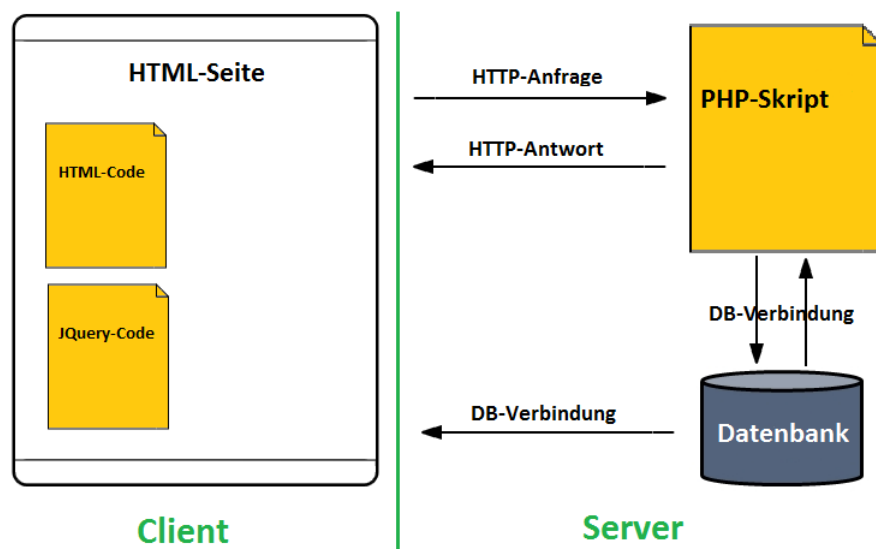


Abbildung 36: Die Architektur einer HTML-Unterseite der Webanwendung.

Die Architektur wird zum größten Teil von der Sitemap (Abb.26) der Webanwendung abgedeckt. Das bedeutet, dass die Anwendungsfunktionen entweder direkt von den Unterseiten selbst ausgeführt werden oder dass die HTML-Seiten Skript-Aufrufe durchführen, welche dann die gewünschten Operationen für die Unterseite ausführen. Durch diese Vorgehensweise wird eine übersichtliche Kapselung der Funktionen erreicht. Diese können dann in den einzelnen Scrum-Sprints einfacher realisiert werden. Jeder Sprint bezieht sich also auf eine Unterseite und die Umsetzung derer Funktion. Aber nicht nur die Sprints werden vereinfacht. Da sich die Anwendung im Web-Bereich befindet, wird für die Geschäftslogik hauptsächlich PHP verwendet. Ein PHP-Skript hat den Vorteil, dass es sich vom Browser über das HTTP-Protokoll ansprechen und ausführen lässt. Folgend ist es also möglich, ein PHP-Skript über eine einfache HTML-Seite auszuführen (Abb.38). Dieser Umstand macht ein MVP Vorgehen überflüssig, da eine Funktionsbasierende Implementierung sinnvoller als ein Schichtenmodell ist.

Die HTML-Seite die als UI (User Interface) benutzt wird, nimmt die Befehle des Benutzers entgegen und übermittelt diese per HTTP-Anfrage an sein entsprechendes PHP-Skript. Dieses führt nun die ihm zugewiesenen Operationen aus. Z.B sorgt das Skript der Teilnehmer-Anmeldung dafür, dass die Angaben des Teilnehmers in die Datenbank gespeichert werden. Das PHP-Skript ist also dafür zuständig mit der Datenbank zu kommunizieren. Datenbank Einträge (Inserts) oder Datenbank Änderungen (Updates) dürfen ausschließlich von einem PHP-Skript vorgenommen werden. Jedoch können sowohl das Skript als auch die HTML-Seite Daten aus der Datenbank anfordern (Select). Diese Daten werden z.B für die *Sidebar* der HTML-Seite genutzt oder für die Berechnungen, welche vom Skript durchgeführt werden müssen.

Ein PHP-Skript muss aber nicht unbedingt ausschließlich mit der Datenbank kommunizieren. Unter Umständen kann das Skript auch eine HTTP-Antwort (Response) an die HTML-Seite liefern. Dabei kann die Antwort für die HTML-Seite verschiedene Folgen haben. Zum einen kann es sich bei der Antwort um ein Ergebnis der Berechnung handeln die vom PHP-Skript berechnet wurde und nun von der HTML-Seite dem Benutzer angezeigt werden soll. Ein Beispiel hierfür wäre die Teilnehmer-Zuordnung. Über die HTML-Seite startet der Organisator die Berechnung der Gästeliste. Ist die Berechnung abgeschlossen liegt die Gästeliste als PHP-Objekt im Skript vor. Diese wird dann als Antwort an die HTML-Seite geliefert damit diese das PHP-Objekt darstellen kann.

Eine weitere Möglichkeit einer HTTP-Antwort vom PHP-Skript kann eine Weiterleitung auf eine neue HTML-Seite bedeuten. Dies kann z.B. der Fall sein, falls die Operation des Skriptes einen Zwischenschritt erfordert.

6.2 Implementierung des Konzeptes

Die bereits im Konzept (Kapitel.4) genannten Beschreibungen und Prozesse müssen nun in Software umgeschrieben werden. Hierfür werden die aufgeführten Prozesse durch Funktionen implementiert. Durch die Nutzung von PHP-Funktionen wird der Code übersichtlicher, da mehrere Befehle einen gemeinsamen Befehlsblock darstellen [?]. Zusätzlich lassen sich die Funktionen besser modifizieren oder im weiteren Verlauf des Projektes leichter austauschen.

Zusätzlich werden die komplexeren Konzepte wie z.B. die Teilnehmer-Zuordnung in verschiedene Klassen aufgeteilt. Die Funktionen werden dann entweder innerhalb der Klassen oder durch Objektreferenzen auf die entsprechenden Funktionen ausgeführt.

Da sich die Vorgehensweise bei der Implementierung ständig wiederholt werden nicht alle Konzeptumsetzungen beschrieben. Dies ist auch nicht nötig, da bereits viele Punkte, die sich auf die Implementierung beziehen, in den vorherigen Kapiteln ausführlich besprochen wurden. Stattdessen wird anhand der Teilnehmer-Zuordnung dargelegt, wie die Konzepte generell umgesetzt wurden.

6.2.1 Struktur des Skriptes

Die Teilnehmer-Zuordnung wurde Aufgrund der Komplexität in drei Klassen aufgeteilt. Die Klassen werden mit PHP implementiert und halten sich an die in PHP verwendeten Klassenkonventionen [?]. Klassen können anders als Skripte nicht direkt über eine HTTP-Anfrage angesprochen werden sondern brauchen ein Skript, welches als Umgebung für die Klasse dient. Damit ein PHP-Skript eine Klasse verwenden kann muss die Klasse dem Skript über einen *include*-Befehl bekannt gemacht werden (Abb.37).

```
//SKRIPT der TEILNEHMER-ZUORDNUNG

include("config_projekt.php");
include("Gruppe.php");
include("GruppenZuweiser.php");

//DB-Verbindung
$dbconnection = mysql_connect($dbhost, $dbuser, $dbpass) or die
("keine Verbindung möglich. Benutzernamen oder Passwort sind falsch");
$dbname= mysql_select_db("usrdb_novahdbs_matthias",$dbconnection) or die
("Die Datenbank existiert nicht.");

mysql_query("SET NAMES 'utf8'");
//-----
//Übergebene GET-PARAMETER
$zeitliche_distanz = $_GET['fahrt_dauer'];
$zeitliche_distanz = (int)$zeitliche_distanz*60;

//-----
//Alle TEILNEHMER die keine GASTGEBER sind
$i = 0;
$query = "SELECT * FROM stoffeessen_teilnehmer
        WHERE gastgeber = 0 AND (id < partner_id OR partner_id = 0)";
$result = mysql_query($query);
while($row = mysql_fetch_array($result,MYSQL_ASSOC)){
    $gaeste_liste[$i] = $row;
    $i++;
}

//-----
//Alle GASTGEBER
$i = 0;
$query = "SELECT * FROM $tabelle2";
$result = mysql_query($query);
while($row = mysql_fetch_array($result,MYSQL_ASSOC)){
    $gastgeber_liste[$i] = $row;
    $i++;
}
```

Abbildung 37: Der Anfang des Skriptes Teilnehmer-Zuordnung.

Das Skript welches die Klassen der Teilnehmer-Zuordnung aufruft und verwendet wird `action_zuordnung.php` genannt. Jedes PHP-Skript das über eine HTML-Seite aufgerufen wird, wird in diesem Projekt mit dem Präfix `action_` markiert. Das Skript wiederum wird von der Unterseite *Teilnehmer-Zuordnung starten* (Abb.26 S.61) aufgerufen.

Die Aufgabe des Skriptes ist es nun, die über die HTTP-Anfragen gesendeten *GET*-Parameter der HTML-Seite und die aus der Datenbank benötigten Daten vorzubereiten. Um die Daten aus der Datenbank anzufragen wird eine Datenbank-Verbindung aufgebaut. Aus der Datenbank werden alle Gastgeber und alle Gäste abgefragt und in Arrays gespeichert. Der in Abb.37 dargestellte PHP-Code zeigt wie die Datenbankabfrage die entsprechenden Daten liefert. Der *GET*-Parameter `$_GET['fahrt_dauer']` der von der HTML-Seite geliefert wurde enthält die vom Organisator bestimmte Fahrdauer die überwacht werden soll. Die Fahrdauer eines Teilnehmers zu einem Gastgeber darf diesen Wert nicht überschreiten. Tut er es doch, so wird ein Interner Zähler gesetzt und die Distanzüberschreitung wird erfasst. Da sich die angegebene Fahrdauer des Organisators auf Minuten bezieht aber in der Anwendung selber Sekunden überprüft werden, muss der Wert des Parameters noch umgerechnet werden.

```
$gruppenZuweiser = new GruppenZuweiser($zeitliche_distanz);  
$gruppenListe = $gruppenZuweiser->erzeugeGruppenListe($gastgeber_liste,$gaeste_liste);  
$anzahl_zeit_ueberschreitungen = 0;
```

Abbildung 38: Der Aufruf der Klasse `GruppenZuweiser` im Skript.

Da die Benötigten Daten nun vorliegen können diese der Klasse *GruppenZuweiser* übergeben werden. Dabei wird die `$zeitliche_distanz` dem Konstruktor der Klasse übergeben und die Arrays `$gastgeber_liste` und `$gaeste_liste` welche eine Liste aus Gastgebern- und Gäste-Informationen wiedergibt wird der Funktion `erzeugeGruppenliste(list1,list2)` übergeben. Diese Methode ist für die Generierung der Gästeliste verantwortlich. Die Variable `$gaeste_liste` ist nicht zu verwechseln mit der von der Anwendung zu berechnenden Gästeliste. Die Variable enthält lediglich die aus der Datenbank angeforderten Informationen aller Teilnehmer die keine Gastgeber sind.

6.2.2 Die Klassen `GruppenZuweiser`, `Gruppe` und `Gast`

Die wichtigste Klasse innerhalb dieses Skriptes ist die *GruppenZuweiser*-Klasse. Bereits in Kapitel 4.1.8 Prozess Teilnehmer-Zeitüberwachung wurde die Klasse als Manager erwähnt und als Model in Abbildung.20 modelliert. Als Klasse verfügt der *GruppenZuweiser* über Klassenvariablen und Klassenfunktionen, die sich wiederum aus den Prozessen des Teilnehmer-Zuordnungs Konzeptes ergeben. Der *GruppenZuweiser* hat wie der Name bereits vermuten lässt die Aufgabe, die Gruppenliste (Abb.16 S.41) zu erzeugen, zu befüllen und zu rotieren. Dies geschieht hauptsächlich in der Methode `erzeugeGruppenliste(liste1,list2)`. Diese Funktion besteht nun aus vielen weiteren Funktionen, welche die

Teilnehmer-Zuordnung Schritt für Schritt abarbeiten. In Abb.39 die Funktionen und deren Aufgaben verdeutlicht.

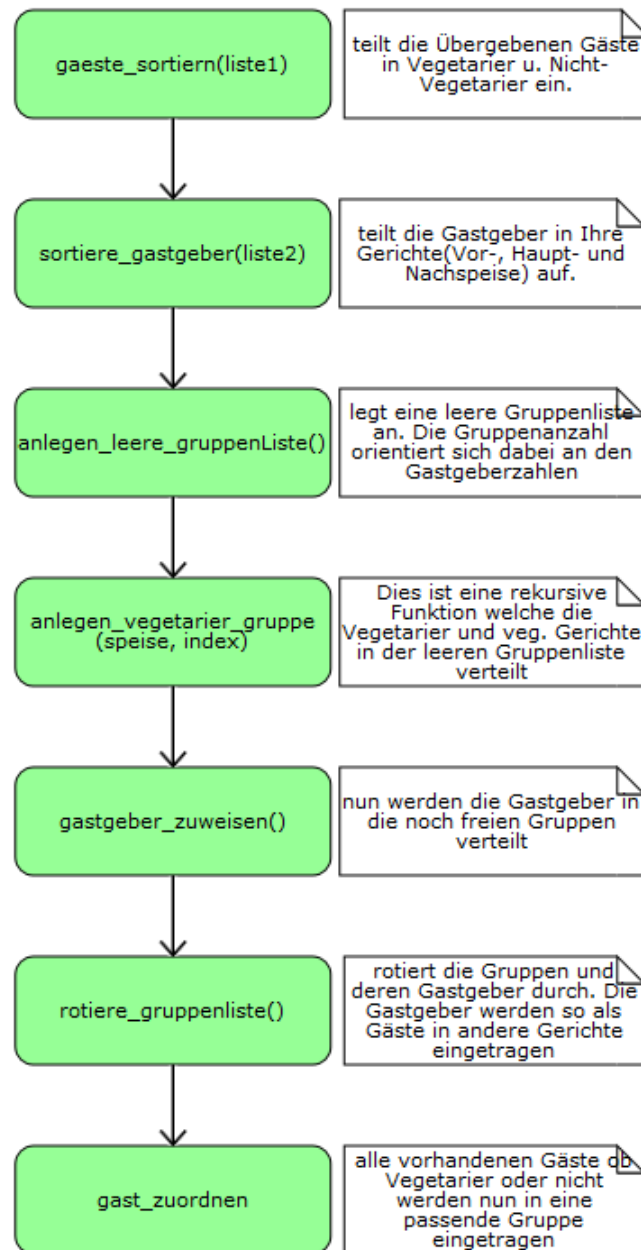


Abbildung 39: Die Prozesskette des GruppenZuweisers mit den entsprechenden Funktionen.

Über die Funktion `anlegen_leere_gruppenliste()` werden nun die *Gruppen*-Objekte initialisiert. Die Klasse *Gruppe* kann in dieser Anwendung als Container-Klasse gesehen

werden. Sie enthält die Informationen der ihr zugeteilten Teilnehmer. Eine Gruppe enthält immer die drei Gastgeber mit jeweils einem anderen Gericht und über jeweils ein Array pro Gastgeber in dem die Gäste des entsprechenden Gastgebers gespeichert werden. Diese Arrays modellieren die *Tische* welche im weiteren Verlauf der Anwendung dargestellt werden. Als Container für die Teilnehmer verfügt die Klasse über Funktionen um die Gastgeber bzw. deren Gäste anzulegen oder wieder aus der Klasse zu lesen. Sobald ein Teilnehmer mit der `add_gast(gast)` Funktion an einen Tisch zugeteilt wird, wird ein *Gast*-Objekt aus seinen Teilnehmer-Informationen initialisiert.

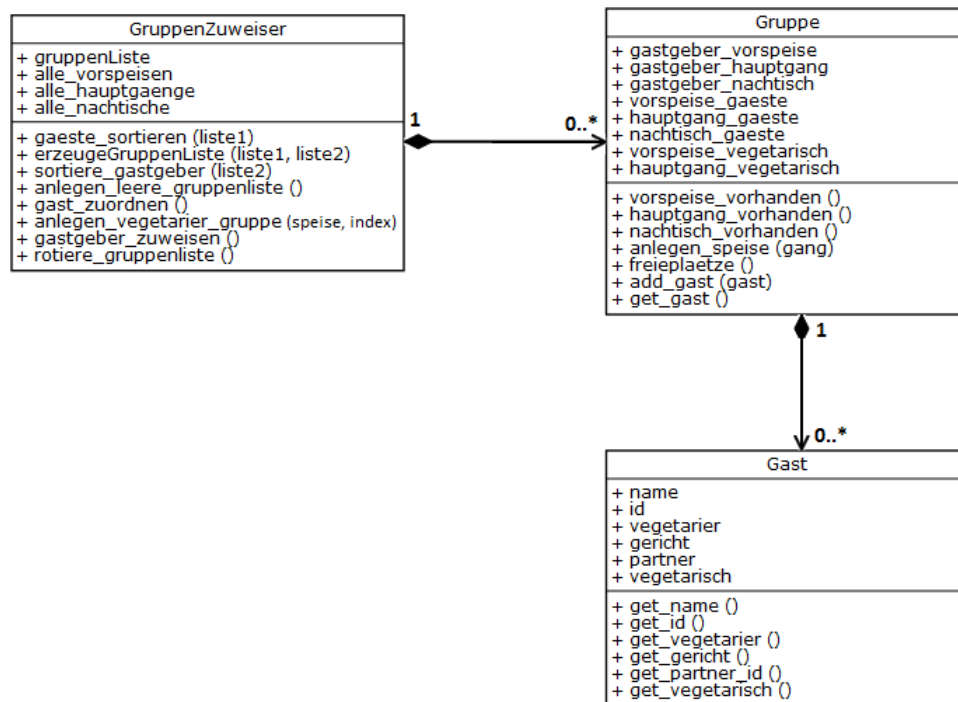


Abbildung 40: Das UML-Klassendiagramm für die Teilnehmer-Zuordnung.

6.3 Datenbank Struktur

Die Datenbank Struktur soll aufzeigen, wie die benötigten Daten innerhalb der Datenbank hinterlegt wurden. Es soll verständlich Begründet werden wieso die gewählte Struktur sinnvoll ist und für welche Teile der Webanwendung die Daten von Bedeutung sind. Durch diese Überlegungen lässt sich auf die Schnittstelle zwischen Datenbank und Anwendung schließen.

6.3.1 Datenbank Tabellen

Die Datenbank welche mit MySQL realisiert wurde besteht aus mehreren einzelnen Tabellen, welche für die Anwendung von essentieller Bedeutung sind. Da jedoch innerhalb der Datenbank mehrere Tabellen existieren die keinerlei Auswirkungen auf die Anwendung haben werden hier nur die wichtigsten Tabellen aufgeführt. Die Tabellen werden dabei so aufgeteilt das ihre Daten in logische Bereiche unterteilt werden.

Teilnehmer Tabelle

Die Teilnehmer Tabelle welche den Namen `staffelessen_teilnehmer` erhält speichert die persönlichen Daten eines Anwenders. Die Tabelle legt für jeden Anwender eine Identifikationsnummer (ID) an, die automatisch inkrementiert wird, falls eine neue Person in die Tabelle gespeichert wird. Ab jetzt kann der Anwender als Teilnehmer bezeichnet werden. Ein Teilnehmer darf nur einmal in der Teilnehmer Tabelle vorkommen, da es sich um eine reale Person handelt. Somit kann die ID des Teilnehmers als Primärschlüssel verwendet werden.

Neben den persönlichen Daten eines Teilnehmers wird gleichzeitig erfasst, welche der Teilnehmer Paare bilden und welche Teilnehmer als Single am Staffelessen teilnehmen. Um dies festzustellen wird die Teilnehmer-ID eines Partners in sein Attribut `partner_id` geschrieben. Dadurch wird eine Relation zwischen zwei Teilnehmern aufgebaut die nun als Paar gekennzeichnet sind. Bei einem Single-Teilnehmer wird die `partner_id` auf Null gesetzt um zu signalisieren das dieser Teilnehmer keinen Partner hat. Dies wird in Abb.41 dargestellt.

id	name	vornam	email	handyNumr	strasse	nr	plz	ort	veg	gastgeber	partner_id	latLong
26	Keune	Kai	test@web.de	178666555	Frühlings:	30	76275	Ettlinge Bruchh	0	1		27 48.92896,8.36521
					Relation							
27	Rupp	Johann	test@web.de	178666555	Frühlings:	30	76275	Ettlinge Bruchh	0	1	26	
					Paar							
28	Klein	Ralf	test@web.de	178666555	Wilhelms:	15	76137	Karlsru	0	1	29	49.0028505,8.4057
29	Klein	Debbie	test@web.de	178666555	Wilhelms:	15	76137	Karlsru	0	1	28	
30	Klocke	Stefan	test@web.de	178666555	Mozartstr	7	76133	Karlsru	0	1	0	49.0142817,8.3802
33	König	Dennis	test@web.de	178666555	Akademie	25	76131	Karlsru	0	Single	0	49.0112279,8.3968
34	Müller	Thorstei	test@web.de	178666555	Im Blumenwi	92	76227	Karlsru	0	1	0	48.9979986,8.4553

Abbildung 41: Ausschnitt aus der Teilnehmer Tabelle.

Über die Teilnehmer Tabelle erhält die Anwendung Zugriff auf die Benötigten Informationen des Teilnehmers wie z.B Höhen und Breitengrad über das Attribut **latLong** oder seine Kontaktdaten, welche für das drucken der Instruktionen von großer Wichtigkeit sind. Die Angaben kommen dabei von der Teilnehmer-Anmeldung welche auf der Startseite von einem Benutzer ausgefüllt werden. Zusätzlich kann der Organisator bei der Teilnehmer-Editierung alle angaben manipulieren. Obwohl die Teilnehmer Tabelle die meisten Eigenschaften besitzt dient sie lediglich der Datenspeicherung und ihr Aufbau ist daher recht simpel.

Gerichte Tabelle

Über die ID eines Teilnehmers wird eine Relation zwischen Teilnehmer und Gericht hergestellt. Die ID wird unter dem Attribut **t_id** als Fremdschlüssel abgelegt. Dadurch kann über die ID des Teilnehmers nun ebenfalls sein Gericht und alle Gerichtsrelevanten Informationen wie z.B die Art des Gerichtes (Vorspeise, Hauptgang, Dessert) oder die bevorzugte Gästeanzahl ermittelt werden (Abb.42).

id	t_id	vegetarier	gericht	gericht_alternativ	anzahlgaeste	vegetarischesGericht
13	26	0	1	NULL	5	0
14	28	0	1	NULL	5	0
15	30	0	1	NULL	6	1
17	34	0	2	NULL	6	0
18	35	0	2	NULL	7	0
19	37	0	0	NULL	6	0
20	39	0	1	NULL	5	0
22	43	0	1	NULL	6	0
23	45	0	2	NULL	5	0
24	47	0	0	NULL	6	0
25	49	0	1	NULL	6	0
26	51	0	2	NULL	6	0

Abbildung 42: Ausschnitt der Gerichte Tabelle.

Zwischen einem Gericht und einem Teilnehmer herrscht eine 1:1 Entitäten Beziehung, was bedeutet, das genau ein Teilnehmer mit höchstens einem Gericht in Relation steht. Ein Teilnehmer muss nicht automatisch ein Gericht austragen. Handelt es sich bei dem Teilnehmer um einen *Gast*, so wird für diesen Teilnehmer kein Eintrag in die Gerichte Tabelle angelegt. Umgekehrt gilt das jedoch nicht. Ein Gericht muss immer eine Relation zu einem Teilnehmer besitzen, da ein Gericht nicht alleine in der Datenbank existieren kann. Das würde nämlich bedeuten das es ein Gericht gibt das von niemandem zubereitet wird, was natürlich unmöglich ist.

Ein Eintrag in die Gerichte Tabelle erfolgt bei der Anmeldung des Teilnehmers falls dieser sich als Gastgeber für das Staffeessen meldet. Diese Tabelle kann ebenfalls wie die Teilnehmer Tabelle über die Editierung durch den Organisator geändert werden.

Geo Tabelle

In dieser Tabelle wird die Distanz zweier Teilnehmer zueinander erfasst. Dafür verwendet die Tabelle wieder die Fremdschlüssel der Teilnehmer ID. Bei dieser Tabelle werden die beiden Fremdschlüssel zusammengefasst. Somit sind beide Fremdschlüssel zur eindeutigen Identifikation der Fahrdauer erforderlich. In dem Attribut **zeit** wird die Distanz zwischen den beiden Teilnehmern in Sekunden festgehalten.

uid1	uid2	zeit
57	26	724
57	62	629
57	64	1410
57	61	859
57	59	1222
57	55	1233
57	51	813
57	49	999
57	47	1192
57	66	1193

Fremdschlüssel 2

Berechnete Fahrdistanz

Fremdschlüssel 1

Der Teilnehmer mit ID 57 muss ca. 724 sek. mit seinem Auto zum Teilnehmer mit der ID 26 fahren.

Abbildung 43: Ausschnitt der Geo Tabelle mit Erklärung.

In dieser Tabelle baut jeder Teilnehmer eine Beziehung mit einem anderem Teilnehmer auf. Dadurch werden alle vorkommenden Relationen der Teilnehmer untereinander erfasst und somit können alle auftretenden Fahrdistanzen abgedeckt werden. Dabei wird die Zeit nur in eine Richtung bemessen. Es gibt also keinen Unterschied ob Teilnehmer A zu Teilnehmer B oder ob Teilnehmer B zu Teilnehmer A fährt. Die Abweichungen der Fahrdistanzen, die durch einen unterschiedlichen Verkehrslauf herrühren könnten, werden also nicht beachtet (z.B. Einbahnstraßen), da sie nicht sonderlich ins Gewicht fallen.

Da eine Relation zwischen zwei expliziten Teilnehmern nur einmal vorkommt, könnte man vermuten das es sich bei dieser Tabelle um eine 1:1 Entitäten-Beziehung zwischen zwei Teilnehmern handelt. Es ist egal ob die ID des Teilnehmers im Attribut **uid1** oder **uid2** vorkommt, da jede Relation genau einmal vorkommt. Dies ist aber inkorrekt, da jeder Teilnehmer, verkörpert durch seine ID als Fremdschlüssel, mehrmals vorkommt und mit allen anderen Teilnehmern und deren Fremdschlüssel in Verbindung steht, wie in Abb.44 dargestellt wird. Somit handelt es sich bei dieser Tabelle um eine n:m Entitäten Beziehung zwischen zwei Teilnehmern und der Geo Tabelle.

Speise Tabelle der Vorspeise, Hauptgang und Dessert

Die drei Tabellen welche die Gastgeber und deren Gäste in Relation zueinander stellen sind sowohl von ihrer Funktion als auch von ihrem Aufbau identisch, weswegen sie hier

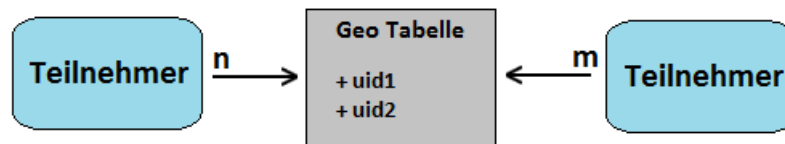


Abbildung 44: Geo Tabelle Entity Relationship Model (ERM).

zum besseren Verständnis als Speise Tabelle zusammengefasst werden können. Die Tabellen teilen sich allerdings in der Datenbank in Vorspeise Tabelle, Hauptgang Tabelle und Dessert Tabelle auf. Diese Tabellen speichern die in der Anwendung zugewiesenen Teilnehmer nach ihren Rollen in der Datenbank. Über die Fremdschlüssel der Teilnehmer ID werden die in diesem Gang vorkommenden Gastgeber mit den Fremdschlüssel der Gäste in Relation gebracht. Somit gibt es also wie bei der Geo Tabelle zwei Schlüssel mit denen die Relation aufgebaut wird (Abb.45).

id	gastgeber_id	gast_id	
1	57	57	Fremdschlüssel 1
2	57	58	Fremdschlüssel 2
3	57	35	
4	57	36	
5	57	30	
6	31	31	
7	31	32	
8	31	64	

Abbildung 45: Ausschnitt der Tabelle für die Vorspeise, den Hauptgang und das Dessert.

Laut der Definition der Anwendung zählt ein Gastgeber für sich selber ebenfalls als Gast. Deshalb kann es in dieser Tabelle häufiger zu doppelten Einträgen in beiden Attributen `gastgeber_id` und `gast_id` kommen. Normalerweise sollte man auf eine solche Verwendung des Fremdschlüssel verzichten, da dies zur Inkonsistenz der Datenbank führen könnte [?]. In dieser Form ist der Nutzen jedoch größer als der Schaden, da nun über die doppelt auftretenden IDs die Gastgeber beim durchlaufen der Tabelle unterschieden werden können. Wird also beim Durchlaufen der Tabelle sowohl in der `gastgeber_id` und der `gast_id` die selbe ID ermittelt so handelt es sich um einen neuen Gastgeber.

Bei dieser Speise Tabelle handelt es sich um eine 1:n Entitäten Beziehung zwischen einem Gastgeber und mehreren Gästen. Wie bereits in den vorherigen Kapiteln erwähnt sind Gastgeber und Gäste lediglich spezielle Rollen innerhalb der Anwendung. Diese Rollen werden ebenfalls bei der Modellierung der Datenbank verwendet. Dadurch handelt es sich also um eine Spezialisierung der Teilnehmer in Gastgeber und Gast wie in Abb.46 aufgeführt wird.

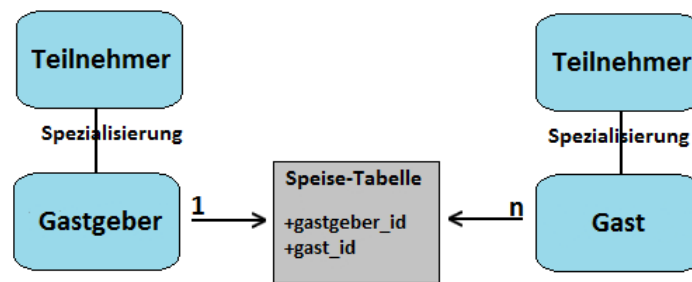


Abbildung 46: Ausschnitt der Tabelle für die Vorspeise, den Hauptgang und das Dessert.

7 Fazit

Bei dieser Bachelor Thesis ging es in erster Linie um die prototypische Implementierung einer Webanwendung, welche den organisatorischen Aufwand eines Staffeleessens reduzieren sollte. Im Rahmen dieser Arbeit wurde dabei ermittelt, welche Aspekte der Organisation aufwändiger sind und wie man diese Problemstellungen durch den Einsatz spezifischer Algorithmen lösen könnte. Zusätzlich sollten die verwendeten Technologien wie PHP oder jQuery beschrieben werden. Über die Technologien sollte auch ein Bezug auf die im Web-Bereich verwendeten Standards eingegangen werden.

Im Bezug auf die Thesis rückt dabei die Implementierung eher in den Hintergrund, da diese durch Frameworks und Standard-Skriptsprachen wie Bootstrap oder PHP vergleichsweise unproblematisch verlief. Dank Bootstrap konnte ein Großteil der Entwurfsphase entfallen, da die verwendeten Komponenten sich als ideales Design für die Webanwendung herausstellten. Durch die Verwendung von Skripten konnte eine überschaubare und sehr leicht zu modifizierende Architektur für die Webanwendung entwickelt werden. Diese Zeiterparnisse wurden nun genutzt um Lösungsansätze für die organisatorischen Aspekte zu entwickeln oder um diese für den Einsatz innerhalb der Webanwendung zu optimieren.

Bereits im frühem Stadium dieser Arbeit war bekannt, dass der Aspekt der Teilnehmer-Zuordnung einen erheblichen Implementierungsaufwand darstellte. Die Zuordnung von Gästen und Gastgebern in eine Gästeliste war bereits bei der von Hand durchgeführten Berechnung so zeitintensiv, dass eine annehmbare Lösung oftmals erst nach mehreren Stunden mit Versuchen und Fehlschlägen erstellt werden konnte. Trotzdem wurde die Zuordnung sehr unterschätzt, da angenommen wurde, dass lediglich ein passender Algorithmus für dieses Problem gefunden werden müsste. Das entwerfen eines geeigneten Algorithmus erwies sich aber als zu zeitintensiv, da die kombinatorische Komplexität der Teilnehmer-Zuordnung nicht klar definiert werden konnten.

Als Lösungsansatz entschied man sich, die Komplexität der Teilnehmer-Zuordnung aufzuteilen und jeweils für die so entstandenen Fragmente, Prozesse und Verfahren zu ent-

wickeln. Durch die Aufteilung der Zuordnung und den definieren von Grundbedingungen, welche in Zusammenarbeit mit dem Kunden aufgestellt wurden, konnte die Teilnehmer-Zuordnung dennoch realisiert werden. Die Anwendung benötigt unter Voraussetzung aller Grundbedingungen nun weniger als eine Minute um eine mögliche Gästeliste zu erstellen. Dies stellt einen massiven Zeitgewinn im Vergleich zur manuellen Verteilung dar. Ein weiterer Vorteil der Webanwendung ist die Umstrukturierung des Anmelde-Verfahrens. Musste bei der Organisation früher jeder Teilnehmer vom Organisator in die Teilnehmerliste eingetragen werden, so steht dem Teilnehmer dank der Anwendung nun ein schlichtes und selbst erklärendes Anmelde-Formular auf der Startseite der Webanwendung zur Verfügung. Durch das Implementieren der Teilnehmer-Zuordnung und Teilnehmer-Anmeldung wurden alle aufgestellten Hauptziele erreicht. Über den Browser lässt sich nun die Webanwendung aufrufen. Somit stehen den Anwendern alle Features der Webanwendung vollständig zur Verfügung.

8 Ausblick

In diesem Kapitel werden Ideen und Konzepte zur Weiterentwicklung der momentanen Webanwendung besprochen. Zu diesen Weiterentwicklungen zählen sowohl noch nicht vorhandene Features, welche zwar für das Projekt geplant, jedoch aus Zeitgründen nicht rechtzeitig umgesetzt werden konnten sowie Funktionen welche erst in der Testphase als zusätzliche Features wahrgenommen wurden. Aber nicht nur die Optimierung von Features sollen hier angesprochen werden sondern auch Optimierungen zum Aufbau oder der Struktur der Webanwendung sollen an dieser Stelle erwähnt werden.

8.1 Features

Die Teilnehmer Informationen werden momentan lediglich im System erfasst und nur der Organisator erhält eine Ansichtsseite und die Möglichkeit, Änderungen an diesen Informationen durchzuführen. Dadurch kann nicht einmal der Teilnehmer selbst seine eigenen Daten beeinflussen oder einsehen. Sollte der Organisator also genötigt sein einen der Teilnehmer zu editieren so muss der Organisator den Teilnehmer manuell darüber in Kenntnis.

Nun könnte die Teilnehmer-Editierung um ein Skript erweitert werden, welches den Teilnehmer per E-Mail darüber benachrichtigt, ob Änderungen an dessen Informationen durchgeführt wurden und welche Daten explizit vom Organisator editiert wurden. Über die `mail()`-Funktion welche in PHP bereits vorhanden ist wäre dieses Skript leicht zu implementieren.

Als weiteres könnte noch die Darstellung der generierten Gästeliste optimiert werden. Momentan wird die Darstellung über einfache table-Elemente verwirklicht. Durch diese HTML-Elemente ist die Darstellung stark eingeschränkt da sich diese Elemente nicht eignen um die komplexen Kombinationen zwischen Tisch, Gastgeber und Gast sachgemäß anzuzeigen. Als Lösung müsste hier ein eigenes HTML-Element entworfen werden, welches sich speziell um die Darstellung der Teilnehmer und des Tisches kümmert. Diese Darstellung könnte ebenfalls um ein Feature erweitert werden welche es dem Organisator

erlaubt, Gäste zwischen zwei verschiedenen Gastgebern via *Drag and Drop* zu verschieben und damit die Zuordnung der Teilnehmer manuell zu beeinflussen nachdem die Zuordnung durch die Anwendung erfolgte.

Dieses Feature könnte durch eine Teilnehmer-Suche unterstützt werden. Über ein Suchfeld könnte der Organisator den Namen eines Teilnehmers eingeben und würde diesen dann farblich in der gesamten Gästeliste angezeigt bekommen. Nun müsste man noch die Teilnehmer selektieren und an eine andere Position versetzen können. Dafür würde sich ein *Droppable Widget* oder *Sortable Widget* eignen. Diese Widgets werden vom JQuery Framework zu Verfügung gestellt und müssten lediglich für die Funktionsweise auf die entsprechenden Teilnehmer-Elemente angepasst werden.

8.2 Architektur

Die prototypische Webanwendung soll in den kommenden Monaten von den Wirtschafts-junioren Karlsruhe in betrieb genommen und dabei ausführlicher getestet werden. Sollte diese Testphase erfolgreich verlaufen wäre es denkbar die Anwendung einem breiteren Publikum zu präsentieren. Um die Webanwendung auch den übrigen Wirtschafts-junioren zu Verfügung zu stellen müsste jedoch das komplette System neu strukturiert werden, da die Anwendung derzeit nicht für multiple Staffeleessen ausgelegt ist. Um dies durchführen zu können muss also eine Systemumgebung geschaffen werden, welche die momentane Webanwendung als Subsystem nutzen würde. Somit würde über der momentanen Webanwendung eine weitere Schicht stehen welche ausschließlich für das anlegen und löschen eines Staffeleessen verantwortlich wäre. In diesem neuen System müsste es dann auch möglich sein mehrere Staffeleessen anzulegen und diese dann zu verwalten.

Man könnte, wie es bei der gegenwärtigen Webanwendung ebenfalls der Fall ist, den einzelnen Organisatoren Sonderrechte in diesem System zukommen lassen, die es einem Organisator ermöglichen, ein Staffeleessen anzulegen. Über Anmelde-Mechanismen könnte man einem Teilnehmer in diesem Fall lediglich zugriff auf ein Spezielles Staffeleessen gewähren. Der Organisator des entsprechenden Staffeleessen würde dann ein Passwort für die Anmeldung festlegen welche dann an die Teilnehmer entweder vom System oder manuell vom Organisator an die gewünschten Teilnehmer verteilt werden würde. Durch diese Herangehensweise wären also alle Wirtschafts-junioren in der Lage ihr eigenes Staffeleessen zu organisieren und zu verwalten und das alles komplett Online.

Tabellenverzeichnis

1	Wahrheitstabelle der Zustände eines Gastgebers der Vorspeise.	42
2	Wahrheitstabelle der Zustände eines Gastgebers der Hauptgangs.	43
3	Wahrheitstabelle der Zustände eines Gastgebers des Desserts.	43
4	Inhalt der einzelnen Instruktionen.	54
5	Sitemap Zurodnung der Use Cases und Anforderungen.	62

Abbildungsverzeichnis

1	Beispiel einer Gästeliste für alle Vorspeisen.	8
2	Die Elemente eines HTML-Dokumentes	13
3	Abstarktion des HTML-Codes	13
4	Beispiel eines Event-Handlers	17
5	Beispiel des Grid-Layouts	19
6	Die in Bootstrap möglichen Raster Optionen	19
7	Darstellung eines SCRUM-Prozesses	23
8	Use Case Diagramm.	30
9	Die grafische Darstellung der Teilnehmer-Zuordnung.	35
10	Die grafische Darstellung der Gruppenliste.	38
11	Die Gruppenliste nach der 1. Permutation.	38
12	Die Gruppenliste nach der 2. Permutation.	39
13	Bildung der <i>Tische</i> nach der Permutation.	39
14	Permutation innerhalb der Gruppe.	40
15	Komplett rotierte Gruppenliste.	41
16	Gäste werden in die Gästeliste eintragen.	42
17	Vegetarier wird in die Leere Gruppenliste eingetragen.	45
18	Ein passendes veg. Gericht wird in die Leere Gruppenliste eingetragen. . .	45
19	Der Ablauf des Prozesses Vegetarier-Verteilung.	46
20	Model des Managers.	47
21	Das Anmeldeformular mit den persönlichen Daten und den anwendungsrelevanten Daten.	49
22	Die JSON-Ausgabe der Geocoding Anfrage für die Bsp. Adresse „1600 Amphitheatre Parkway, Mountain View, CA“.	51
23	Model eines Ablauf der Teilnehmer-Anmeldung.	52
24	Die Darstellung der Teilnehmerliste und das Formular.	56
25	Die Darstellung der Header- bzw. Content-Rahmen.	60
26	Die Sitemap der Webanwendung.	61
27	Die Anmeldungs Sidebar.	63
28	Die unterschiedlichen Navigationsleisten im direkten Vergleich.	64
29	Die Sortable-List der Unterseite <i>Teilnehmer-Zuordnung starten</i>	65
30	Darstellung eines Konfliktes auf der Unterseite <i>Gästeliste ansehen</i>	65
31	Sidebar zur Optimierung der Teilnehmer-Editierung.	66
32	Die Onlinestudie des ARD/ZDF.	67
33	Das HTML Box Model.	68

34	Viewport vor dem Unterschreiten des Breakpoints.	69
35	Viewport nachdem der Breakpoint unterschritten wird.	69
36	Die Architektur einer HTML-Unterseite der Webanwendung.	70
37	Der Anfang des Skriptes Teilnehmer-Zuordnung.	72
38	Der Aufruf der Klasse GruppenZuweiser im Skript.	73
39	Die Prozesskette des GruppenZuweisers mit den entsprechenden Funktionen.	74
40	Das UML-Klassendiagramm für die Teilnehmer-Zuordnung.	75
41	Ausschnitt aus der Teilnehmer Tabelle.	76
42	Ausschnitt der Gerichte Tabelle.	77
43	Ausschnitt der Geo Tabelle mit Erklärung.	78
44	Geo Tabelle Entity Relationship Model (ERM).	79
45	Ausschnitt der Tabelle für die Vorspeise, den Hauptgang und das Dessert.	79
46	Ausschnitt der Tabelle für die Vorspeise, den Hauptgang und das Dessert.	80

Literatur

- [1] Stefan Zink. Generic Open Source Gamification Framework. pages 50–59, 2010.
- [2] Ivo Blohm and Jan Marco Leimeister. Gamification: Design of IT-based enhancing services for motivational support and behavioral change. *Business and Information Systems Engineering*, 5(4):275–278, 2013.