



Dies ist eine vorläufige Version der Master-Thesis.

Assistenzplaner

Serverbasierte Software zur Koordinierung der Assistenten für behinderte Menschen

Masterarbeit

im Studiengang Software Engineering und Informationstechnik



TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

vorgelegt von: Ulrich Belitz

Fakultät: Elektrotechnik Feinwerktechnik Informationstechnik

Matrikelnummer: 2464776

Erstgutachter: Prof. Dr. Thomas Mahr

Zweitgutachter: Prof. Dr. Ralph Lano



Eidesstattliche Erklärung

Ich, Ulrich Belitz, Matrikel-Nr. 2464776, versichere hiermit, dass ich meine Masterarbeit mit dem Thema

Assistenzplaner - Serverbasierte Software zur Koordinierung der Assistenten für behinderte Menschen

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Mir ist bekannt, dass ich meine Masterarbeit zusammen mit dieser Erklärung fristgemäß nach Vergabe des Themas in zweifacher Ausfertigung und gebunden im Studienbüro der Technischen Hochschule Nürnberg Georg Simon Ohm abzugeben oder spätestens mit dem Poststempel des Tages, an dem die Frist abläuft, zu senden habe.

Taunusstein, 13. September 2014

ULRICH BELITZ



Abstract

Hier wird noch die Zusammenfassung auf Englisch ergänzt.



Kurzfassung

Der [Assistenzplaner](#)¹ wurde als serverbasierte Anwendung geschaffen, um körperlich behinderte Menschen (Klienten) und ihre Assistenz-Teams bei der Planung zu unterstützen.

Ein Klient ist auf persönliche Assistenz angewiesen. Die Dienstzeiten können von wenigen Stunden am Tag bis zu einem ganzen Tag variieren.

Mit dem Assistenzplaner hat der Klient die Möglichkeit sein Assistenz-Team zu verwalten. Dabei kann der Klient neben den Kontaktdaten auch Priorisierungswerte und Stundenkontigente der Assistenten definieren.

Der Klient kann mit dem Monats-Plan die Dienstzeiten für einen Monat festlegen. Seine Assistenten werden per Knopfdruck benachrichtigt und gebeten ihre Termine in einen Kalender einzutragen.

Ein Algorithmus erstellt automatisch einen Dienstplanvorschlag, der alle Vorlieben des Klienten und die möglichst gleichmäßige Ausschöpfung der Stundenkontigente berücksichtigt. Der Klient kann sich mehrere Dienstplanvorschläge erstellen lassen. Weiterhin ist es möglich den Dienstplan manuell anzupassen.

Eine Aufgaben-Verwaltung rundet den Assistenzplaner ab.

In der vorliegenden Ausarbeitung werden nach einem einleitenden Grundlagenteil die Anforderungen dargestellt. Bei einer Marktanalyse wird verifiziert, dass es noch keine bestehende Lösung gibt, die alle Anforderungen erfüllt. Die Kapitel Software-Design und Implementierung zeigen, wie der Autor die Anforderungen umgesetzt hat. Die Arbeit schließt mit einem Fazit ab, in dem einige Verbesserungs- und Erweiterungsmöglichkeiten aufgezeigt werden.

Bei Abgabe dieser vorliegenden Ausarbeitung wurde ein [Code-Freeze des Assistenzplaners](#)² durchgeführt.

Der Autor hat den Assistenzplaner als Open-Source-Projekt bei [GitHub](#)³ veröffentlicht und wird zukünftig weiter daran arbeiten.

¹<http://assistenzplaner.de/>

²<http://assistenzplaner.de/master>

³<https://github.com/Ulle84/AssistancePlaner>



Danksagung

Ich danke Herrn Prof. Dr. Mahr für die Betreuung und Begutachtung dieser Masterarbeit. Weiterhin danke ich Herrn Prof. Dr. Lano für das Zweitgutachten.

Herzlicher Dank geht an Patrick! Ohne ihn wäre diese Arbeit nicht entstanden. Er hat maßgeblich die Anforderungen gestellt und bereitwillig mit seinem Team die Software getestet.

Ich danke Frederik sehr für die vielen Skype-Gespräche, das gegenseitige Motivieren und die zahlreichen Korrekturanregungen.

Melanie danke ich vielmals für die vielen Korrekturanregungen und die Gastfreundschaft während der Präsenz-Phasen des Studiums.

Meinen Eltern danke ich sehr herzlich für alles, was sie mir ermöglicht und auf den Weg gegeben haben!

Unendlich dankbar bin ich meiner Frau Marielle! Sie hat mich während des gesamten Studiums verständnisvoll unterstützt und mir den Rücken frei gehalten. Weiterhin hat sie wertvolle Anregungen und Korrekturvorschläge beigetragen.



Inhaltsverzeichnis

Eidesstattliche Erklärung	2
Abstract	3
Kurzfassung	4
Danksagung	5
1. Motivation	8
2. Grundlagen	9
2.1. Netzwerke und Internet	9
2.2. Web-Technologien	12
2.3. Modellierungssprachen	17
3. Anforderungsermittlung	19
3.1. Begrifflichkeiten	19
3.2. Ist-Prozess	20
3.3. Soll-Prozess	21
3.4. Benutzergeschichten	21
3.5. Abgeleitete Anforderungen	28
4. Marktanalyse	30
4.1. Dienstplan in Excel	30
4.2. Offline-Anwendungen	30
4.3. Online-Anwendungen	31
5. Software-Design	33
5.1. SysML Anforderungsdiagramm	33
5.2. UML Komponentendiagramm	35
5.3. UML Klassendiagramme	36
6. Implementierung	43
6.1. Wahl der Programmiersprachen	43
6.2. Einheitliches Layout	44
6.3. Persistenz der Daten	44

*Inhaltsverzeichnis*

6.4. Benutzer-Verwaltung	45
6.5. Willkommen-Seite	46
6.6. Navigation	47
6.7. Monatsnavigation	48
6.8. Anlegen eines neuen Klienten-Zugangs	48
6.9. Übersicht	50
6.10. Einstellungen	50
6.11. Standard-Dienstzeiten	51
6.12. Team-Verwaltung	52
6.13. Monatsplan	54
6.14. Kalender	55
6.15. Dienstplan	56
6.16. Entwicklung des Algorithmus zur Dienstplanerstellung	60
6.17. Aufgaben-Verwaltung	71
6.18. Feedback	73
6.19. Dokumentation	74
6.20. Impressum	75
7. Überführung in den Produktiv-Betrieb	76
8. Ergebnis	77
9. Fazit und Ausblick	78
9.1. Einsatz bei mehreren Teams	78
9.2. Mögliche Verbesserungen	79
9.3. Mögliche Erweiterungen	83
10. Literaturverzeichnis	86
A. Tabellenverzeichnis	88
B. Abbildungsverzeichnis	89
C. Verzeichnis der Listings	91
D. Abkürzungsverzeichnis	92
E. Inhalt der beiliegenden CD	93
F. Verwendete Software	94



1. Motivation

1. Motivation

Ein Bekannter des Autors ist ein körperlich behinderter Mensch (im folgenden „Klient“ genannt), der auf persönliche Assistenz angewiesen ist. Der Klient hat sieben Assistenten. Bisher erstellte der Klient den Dienstplan jeden Monat manuell. Der Klient hat gleichzeitig die sieben E-Mails seiner Assistenten überblickt (in denen mögliche Arbeits-Termine enthalten waren) und einen möglichst ausgewogenen Dienstplan erstellt. Der Dienstplan sollte die Verfügbarkeiten und die Stundenkontingente der Assistenten möglichst gut berücksichtigen. Das manuelle Erstellen des Dienstplans dauerte ca. vier Stunden pro Monat.

Um den Klienten zu entlasten und technisch besser zu unterstützen, wurde die Idee des Assistenzplaners ins Leben gerufen. Der Assistenzplaner soll als webbasierte Anwendung den Klienten und sein Team bei der Dienstplanerstellung unterstützen. Der Klient und alle Assistenten erhalten einen personalisierten Zugang zum Assistenzplaner und können alle nötigen Informationen austauschen. Ein Algorithmus erstellt den Dienstplan unter Berücksichtigung aller Vorlieben des Klienten und der Stundenkontingente der Assistenten. Der Zeitaufwand für die Dienstplanerstellung sollte auf wenige Minuten sinken.



2. Grundlagen

In diesem Kapitel werden die technischen Grundlagen für das Verständnis der vorliegenden Arbeit gelegt. Wer schon selbst Webprogrammierung mit PHP, HTML, CSS und JavaScript gemacht hat, kann dieses Kapitel überspringen.

2.1. Netzwerke und Internet

Client/Server-Paradigma

„Netzwerkanwendungen arbeiten in einer Kommunikationsform, die man Client/Server-Paradigma nennt. Die Serveranwendung wartet passiv auf Kontaktaufnahme, während die Clientanwendung die Kommunikation aktiv einleitet.“ [Comer 2002, S. 405]

Clientsoftware

Nach [Comer 2002, S. 406] weist die Clientsoftware unter anderem folgende Merkmale auf:

- Sie „ist ein Anwendungsprogramm, die vorübergehend zum Client wird, wenn entfernter Zugriff notwendig ist, das lokal aber auch andere Aufgaben ausführt.“
- Sie „läuft lokal auf dem Personalcomputer eines Benutzers“
- Sie „leitet den Kontakt mit einem Server aktiv ein.“
- Sie „benötigt keine spezielle Hardware oder ein besonderes Betriebssystem.“



Serversoftware

Nach [Comer 2002, S. 406] weist die Serversoftware unter anderem folgende Merkmale auf:

- Sie „ist ein dediziertes Programm mit dem speziellen Zweck, einen Dienst bereitzustellen, kann aber auch gleichzeitig mehrere entfernte Clients bedienen.“



2. Grundlagen

- Sie „läuft auf einem gemeinsam genutzten Computer.“
- Sie „wartet passiv auf die Verbindungsaufnahme durch einen entfernten Client.“
- Sie „nimmt den Kommunikationswunsch von beliebigen Clients entgegen, bietet aber nur einen Dienst.“
- Sie „setzt leistungsstarke Hardware und ein gehobenes Betriebssystem voraus.“

URL

Uniform Resource Locator (URL) ist ein Syntaxformat, das alle Informationen kodiert, die zum Erreichen einer Webseite notwendig sind. Die URL hat folgendes Format:

protokoll://domainNameComputer:port/dokumentName

Die Angabe eines Ports ist optional.

Ein Beispiel für eine URL ist die Startseite des Assitenzplaners:

<http://assistenzplaner.de/index.html>

Browser

„Ein Browser ist ein interaktives Programm, mit dem ein Benutzer Informationen aus dem Web lesen kann. Die Informationen enthalten wählbare Elemente, über die der Benutzer weitere Informationen anzeigen lassen kann.“ [Comer 2002, S. 489]

Link

Link (manchmal auch Hyperlink) ist der umgangssprachliche Begriff für eine Hypertext-Referenz. „Im Browser ist eine Hypertext-Referenz in einem angezeigten Dokument zunächst ein wählbares Element. Wählt der Benutzer das Element, folgt der Browser der Referenz, holt das betreffende Dokument und ersetzt die momentane Anzeige durch das neue Dokument.“ [Comer 2002, S. 496]



HTTP

„Die Interaktion zwischen einem Browser und einem Webserver erfolgt über das *HyperText Transport Protocol (HTTP)*. HTTP erlaubt es dem Browser, ein bestimmtes Element anzufordern, das der Server ausgibt. In der Praxis ist HTTP aber komplexer, weil ein Server zusätzliche Statusinformationen mit jedem Transfer sendet und das Protokoll es einem Browser gestattet, Informationen zu senden und anzufordern.“

[Comer 2002, S. 497]



„HTTP unterstützt vier Basisoperationen, die ein Browser in einer Anfrage spezifizieren kann:

- *GET* verlangt ein spezifisches Element vom Server. Der Server gibt einen Kopf (HEAD) zurück, der Statusinformation, gefolgt von einer Leerzeile und dem Element, enhält.
- *HEAD* verlangt Statusinformationen über ein Element. Der Server gibt den Status zurück, jedoch ohne Kopie des Elements selbst.
- *POST* sendet Daten an den Server. Der Server hängt die Daten an ein spezifisches Element an (z. B. wird eine Nachricht an eine Nachrichtenliste angehängt).
- *PUT* sendet Daten an den Server. Der Server benutzt die Daten, um ein spezifisches Element zu ersetzen.



Ein Browser erzeugt eine HTTP-Anfrage, wenn ein Benutzer eine URL eingibt oder einen Link wählt. In beiden Fällen sendet der Browser eine GET-Anfrage, die ein Element spezifiziert, und der Server gibt ihm das angefragte Element zurück.“ [Comer 2002, S. 497]

Webdokumente

Es gibt drei verschiedene Typen von Webdokumenten [Comer 2002, S. 508]:

- Statisch: Das Dokument ist fest in einer Datei definiert. Der Autor bestimmt einmalig bei Erstellung den Inhalt. Jede Anfrage an das statische Dokument führt zeitlich unabhängig zur identischen Ausgabe. Das statische Verhalten kann technisch mit starren HTML-Seiten realisiert werden.
- Dynamisch: Das Dokument ist nicht fest definiert. Sobald eine Anfrage an den Webserver gestellt wird, wird das dynamische Webdokument zusammengestellt. Der Inhalt des dynamischen Dokumentes kann sich von Anfrage zu



2. Grundlagen

Anfrage ändern. Das dynamische Verhalten kann mit Server-Script-Sprachen wie z. B. PHP realisiert werden.

- Aktiv: Das aktive Dokument wird vom Client erstellt. Der Server liefert lediglich das Programm zur Erstellung des Dokumentes aus. Die konkrete Struktur des Dokumentes wird im Client zusammengebaut. Die benötigten Inhalte werden vom Client beim Server angefragt. Das aktive Verhalten kann mit Client-Script-Sprachen wie z. B. JavaScript realisiert werden.

Cookie

„Wenn ein Server umfangreiche Statusinformationen speichern muss, benötigt er die Informationen auf einer lokalen Platte. Statusinformationen werden einem Browser in Form eines *Cookies* weitergegeben. Jedes Cookie besteht aus zwei Zeichenketten, die man als *Name/Wert-Paar* bezeichnet. Im Teil *Name* befindet sich der Name der Website und im Teil *Wert* eine kleine Zeichenkette, die der Browser speichert. Wenn der Browser die Website wieder kontaktiert, fügt er das Cookie in die Anfrage ein. Aus Sicht eines Servers hat es also den Anschein, dass der Browser Statusinformationen speichern und zurückgeben kann.“ [Comer 2002, S. 514f]

2.2. Web-Technologien

XML

Extensible Markup Language (XML) ist eine Auszeichnungssprache. Mit XML lassen sich hierarchisch strukturierter en darstellen. Mit den textbasierten XML-Dateien ist es leicht möglich Daten zwischen Systemen auszutauschen. Die XML-Dateien sind für den Menschen gut lesbar. Die Syntax ist einfach gehalten. Es gibt Tags und tribute. Die Tags können ineinander geschachtelt werden. Ein beispielhaftes XML Dokument findet sich in Listing 2.1. Hier gibt es einen Wurzelknoten **AssitanceTeam** mit zwei Kinderknoten **Assistant**, die jeweils das Attribut **isAvailable** mit unterschiedlichen Werten beinhalten. Die Kinderknoten (**Assistant**) haben wiederum eigene Kinderknoten (**Name** und **eMailAddress**). Die Schachtelung der Knoten kann beliebig tief erfolgen.



Listing 2.1: Beispiel für ein XML Dokument

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <AssitanceTeam>
3   <Assistant isAvailable="true">
4     <Name>Max Mustermann</Name>

```



2. Grundlagen

```

5      <eMailAddress>max@mustermann.de</eMailAddress>
6  </Assistant>
7  <Assistant isAvailable="false">
8      <Name>Markus Mustermann</Name>
9      <eMailAddress>markus@mustermann.de</eMailAddress>
10 </Assistant>
11 </AssitanceTeam>
```

HTML

HyperText Markup Language (HTML) ist eine Auszeichnungssprache und hat „die Aufgabe, die logische Grob- und Feinstruktur einer Webseite zu beschreiben. [...] HTML ist die Sprache zur Strukturierung von Texten, wobei aber auch die Möglichkeit besteht, Grafiken und multimediale Inhalte in Form einer Referenz einzubinden und in den Text zu integrieren“ [Münz und Gull 2012, S. 21].

Es ist unter anderem möglich, vergleiche [Münz und Gull 2012, S. 21],

- das Dokument in Bereiche wie Kopf, Inhalt, Fuß, Navigation und Artikel einzuteilen 
- dem Dokument Überschriften, Textabsätze, Listen und Tabellen hinzuzufügen
- Links (siehe Abschnitt 2.1) auf beliebige andere Webseiten oder Datenquellen im Internet einzubinden
- Formulare in das Dokument zu integrieren
- Erweiterungssprachen wie CSS oder JavaScript einzubinden 

Ein Beispiel für ein HTML Dokument findet sich in Listing 2.2. Die Verwandtschaft von XML und HTML spiegelt sich in der gemeinsamen Struktur der Dokumente wider.

Listing 2.2: Beispiel für ein HTML-Dokument

```

1 <!DOCTYPE html>
2 <meta charset="utf-8">
3 <html>
4 <head>
5   <title>Assistenzplaner</title>
6   <link rel="stylesheet" type="text/css" href="../../CSS/example.css" media="all"/>
7   <script language="JavaScript" src="../../JavaScript/example.js"></script>
8 </head>
9 <body onload="init()">
```



2. Grundlagen

```

10 <h1>Aktuelle Zeit</h1>
11 <span class="time" id="createContentWithJavaScript"></span>
12 </body>
13 </html>
```

CSS

Bei *Cascading Stylesheets (CSS)* „handelt [es] sich um eine beschreibende Ergänzungssprache für HTML. Sie klinkt sich nahtlos in HTML ein und hat zwei Aufgaben: das Formatieren von Inhalten und das Gestalten von Webseitenlayouts.“ [Münz und Gull 2012, S. 23]

„CSS erlaubt es, zentrale Formate zu definieren, beispielsweise für alle Überschriften erster Ordnung oder für alle Textabsätze mit einem bestimmten Klassennamen oder für hervorgehobenen Text, der innerhalb einer Tabellenzelle vorkommt. Die zentralen Formate können in eine externe Style-Datei ausgelagert werden [...] So ermöglicht CSS seitenübergreifend einheitliche Layouts und Formatierungen.“ [Münz und Gull 2012, S. 24]

Listing 2.3: Beispiel für ein CSS-Dokument

```

1 /* set global font for all elements */
2 html * {
3     font-family: Verdana;
4 }
5
6 /* change font color of all headings */
7 h1, h2, h3, h4, h5, h6 {
8     color: #0000ff;
9 }
10
11 /* define attributes of all elements with class "time" */
12 .time {
13     border: 1px solid black;
14     padding: 10px;
15 }
```

JavaScript

JavaScript ist eine Script-Sprache, die meistens im Browser (also clientseitig) ausgeführt wird. Mögliche Einsatzgebiete von JavaScript sind unter anderem:

- Plausibilitätsprüfung (Validierung) von Formulareingaben vor dem Absenden



2. Grundlagen

- Dynamische Manipulation von Webseiten mit Hilfe von DOM
- Senden und Empfangen von Daten, ohne dass der Browser die Seite neu laden muss mit AJAX (siehe Abschnitt 2.2)

Ein Beispiel für ein JavaScript-Dokument findet sich in Listing 2.4.

Listing 2.4: Beispiel für ein JavaScript-Dokument

```

1 function init() {
2     var date = new Date();
3
4     var element = window.document.getElementById("createContentWithJavaScript");
5     element.textContent = date.toLocaleString();
6 }
```

Bei JavaScript sind einige Objekte jederzeit verfügbar (zum Beispiel `RegExp`, `Date`, `window`). Diese Objekte lassen sich ur Laufzeit noch um Funktionalität erweitern. Dies geschieht über den Prototyping-Mechanismus. Ein Beispiel findet sich in Listing 2.5. Hier wird das bestehende `Date`-Objekt um die Funktion `addYears(number)` erweitert.

Listing 2.5: Beispiel für die Erweiterung eines bestehenden Objekts in JavaScript

```

1 Date.prototype.addYears = function (numberOfYears) {
2     var year = this.getFullYear();
3     year += parseInt(numberOfYears);
4     this.setFullYear(year);
5 }
```

Zusammenarbeit von HTML, CSS und JavaScript

HTML, CSS und JavaScript ergänzen sich gegenseitig. Wenn man der Idee von [Teague 2011, S. 8] folgt, dann ergeben die drei Webtechnologien eine vollständige Sprache: HTML stellt die Substantive zur Verfügung, CSS die Adjektive und Adverbien und JavaScript fügt die Verben hinzu.

Wenn man sich den HTML-Code Listing 2.2 im Browser anut (siehe Abbildung 2.1), dann ist zu erkennen, wie die Zusammenarbeit der drei Webtechnologien sich in der Praxis auswirkt. Im HTML wird die Struktur des Dokumentes definiert, mit dem verwendeten CSS-Dokument (Listing 2.3) wird das Aussehen angepasst und mit JavaScript (Listing 2.4) wird dynamisch Inhalt erzeugt.



Aktuelle Zeit

17.8.2014 13:35:09

Abbildung 2.1.: Das Ergebnis des Zusammenspiels von HTML, CSS und JavaScript im Browser dargestellt

DOM

Document Object Model (DOM) ist eine Programmierschnittstelle für den Zugriff auf HTML- oder XML-Dokumente. „HTML-Dokumente besitzen eine hierarchische Struktur, die im DOM als Baumstruktur repräsentiert wird. Die Knoten des Baums stellen die verschiedenen Arten von Inhalt in einem Dokument dar.“ [Flanagan 2002, S. 310] DOM stellt unter anderem einen Satz von Methoden zur Verfügung, der es möglich macht, Elemente des Dokumentes aufzufinden. Zum Beispiel kann man mit Hilfe von

```
var captions = window.document.getElementsByTagName("h1");
```

alle Überschriften erster Ordnung finden.

Weiterhin stellt DOM Methoden zur Verfügung um Inhalte des Dokument-Baums zu verändern. Mit dem Code

```
captions[0].setAttribute("id", "firstCaption");
```

wird der *Identifikator (ID)* der ersten Überschrift (erster Ordnung) gesetzt.

AJAX

Bei *Asynchronous JavaScript and XML (AJAX)* handelt es sich nicht direkt um eine Webtechnologie, sondern eher um eine spezielle Denkweise, die Kombination von HTML bzw. XML, CSS und JavaScript einzusetzen.

„Mit Ajax können einzelne Informationen nach Bedarf vom Server geholt und in die bestehende Seite integriert werden. Im besten Fall entfallen die Wartezeiten, und die Kommunikation mit dem Server tritt aus der Sicht des Anwenders in den Hintergrund. Ajax-Applikationen ermöglichen so eine Bedienung, die eher an eine Desktop-Anwendung als an eine Webapplikation erinnert.“ [Koch 2011, S. 309]

Ein spezielles JavaScript-Objekt namens **XMLHttpRequest** ermöglicht die asynchrone Kommunikation mit dem Server. Durch den asynchronen Aufruf kann der Benutzer direkt weiterarbeiten und wird nicht durch die Kommunikation mit dem Server



2. Grundlagen

blockiert. Ist die Antwort des Servers beim Client angekommen, wird sie sofort verarbeitet und gegebenenfalls dem Nutzer dargestellt.

Mit Hilfe von AJAX ist es möglich aktive Dokumente (siehe Abschnitt 2.1) zu erstellen.

PHP

PHP Hypertext Preprocessor (PHP) ist eine Programmiersprache, die meistens auf dem Server ausgeführt wird. Mit ihr ist es möglich Internetanwendungen zu erstellen. Dabei erzeugt PHP die HTML-Seiten, die im Browser angezeigt werden. Mit PHP kann man objektorientiert programmieren. Ein einfaches PHP-Script ist in Listing 2.6 gezeigt.

Listing 2.6: Beispiel für ein PHP-Dokument

```

1 <?php session_start(); ?>
2 <!DOCTYPE html>
3 <meta charset="utf-8">
4 <html>
5 <head>
6   <title>Assistenzplaner</title>
7   <link rel="stylesheet" type="text/css" href="../CSS/global.css" media=
8     all"/>
9 </head>
10 <body>
11 <?php include('navigation.php'); ?>
12 <? echo '<h1>Willkommen beim Assistenzplaner</h1>'; ?>
13 </body>
14 </html>
```

2.3. Modellierungssprachen

Modellierungssprachen dienen der Modellierung von Software. Mit Hilfe von Modellierungssprachen kann man komplexe Systeme konstruieren. Durch mehrere Abstraktionsebenen behält man die Übersicht. Modellierungssprachen geben einem Software-Architekten die Möglichkeit sich sehr konkret auszudrücken, ohne Code schreiben zu müssen.



UML

Die *Unified Modeling Language (UML)* „dient zur Modellierung, Dokumentation, Spezifizierung und Visualisierung komplexer Systeme, unabhängig von deren Fach- und Realisierungsgebiet. Sie liefert die Notationselemente gleichermaßen für die statischen und dynamischen Modelle von Analyse, Design und Architektur und unterstützt insbesondere objektorientierte Vorgehensweisen.“ [Rupp u. a. 2012, S. 4]

SysML

System Modeling Language (SysML) ist eine grafische Sprache, die sich zur Modellierung von komplexen Systemen einsetzen lässt. Einige Diagrammtypen aus der UML wurden übernommen und weitere Diagrammtypen hinzugefügt. Mit SysML lassen sich Systeme modellieren, die aus Hard- und Software bestehen.



3. Anforderungsermittlung

In diesem Kapitel werden die Anforderungen dargestellt. Neben den konkreten Benutzergeschichten, die von den zukünftigen Benutzern formuliert wurden, werden auch nicht explizit erwähnte Anforderungen festgehalten.

3.1. Begrifflichkeiten

In diesem Abschnitt werden einige Begriffe erläutert, die im Folgenden häufig verwendet werden.

Klient

Der Klient ist ein körperlich behinderter Mensch, der im alltäglichen Leben permanent auf Hilfe angewiesen ist. Im Rahmen der individuellen Schwerstbehindertenbetreuung bekommt er Assistenten an die Seite gestellt. Mit Hilfe seiner Assistenten kann er ein selbstbestimmtes Leben führen.

Assistent

Der Assistent ist ein Helfer, der den Klienten pflegt und ihn im alltäglichen Leben unterstützt.

Team



Der Klient muss rund um die Uhr mit Assistenz unterstützt werden. Dafür bedarf es eines Teams. Das Team eines Klienten besteht aus mehreren Assistenten. In dem konkreten Fall besteht das Team aus sieben Assistenten.



3. Anforderungsermittlung

Dienst

Wenn ein Assistent Dienst hat, dann ist er permanent in der Nähe des Klienten und steht ihm helfend zur Verfügung. Für den Leistungsnachweis ist eine Berechnung der Dienststunden notwendig. Wenn ein Assistent einen Dienst von 13:00 Uhr eines Tages bis 8:00 Uhr des Folgetages wahrnimmt, dann sind das rechnerisch 19 Stunden, die er gearbeitet hat. Die Nacht wird mit 6 Stunden berechnet und abgezogen. So bekommt der Assistent eine Netto-Arbeitszeit von 13 Stunden vergütet.

Bereitschaft



Wenn ein Assistent Bereitschaft hat, dann muss er in der Lage sein den Dienst zu übernehmen und zu einer festgelegten Zeit (zum Beispiel von 10:00 bis 11:00 Uhr) telefonisch erreichbar sein. Falls der diensthabende Assistent erkrankt, springt der Bereitschaftshabende ein und übernimmt den Dienst. Bei Diensten, deren Netto-Arbeitszeit größer als 13 Stunden ist, wird eine zweite Bereitschaftsstunde (zum Beispiel von 18:00 bis 19:00 Uhr) vereinbart, falls der diensthabende Assistent während der Arbeitsausführung erkrankt. Jede Bereitschaftsstunde wird mit dem Faktor 0.5 vergütet.

Dienstplan

Im Dienstplan sind die Dienst- und Bereitschaftszeiten der Assistenten festgehalten. Für jeden Tag gibt es einen Assistenten, der Dienst hat, und einen Assistenten, der Bereitschaft hat.

Stundenkontingent



Die Assistenten arbeiten unterschiedlich viel. Die Monatsstundenzahl variiert zwischen 30 und 130. Bei der Dienstplanerstellung ist darauf zu achten, dass alle Assistenten möglichst gleichmäßig ihr Stundenkontingent ausgeschöpft bekommen.

3.2. Ist-Prozess

Vor Beginn der Arbeiten des Autors war der Prozess zur Dienstplanerstellung komplett manuell. Der Klient schickte für die Dienstplanerstellung eine E-Mail mit Hinweisen für den Monat an sein Team und bat um Rückmeldung bezüglich der Verfügbarkeiten. Die Assistenten beantworteten diese E-Mail mit Zeitangaben für mögliche



3. Anforderungsermittlung

Dienste. Der Klient erstellte manuell den Dienstplan und musste dabei ständig zwischen seiner Übersichts-Tabelle und allen Antworten der sieben Team-Mitglieder wechseln. Nach der Fertigstellung des Dienstplans versandte der Klient eine E-Mail mit dem Dienstplan an sein Team. Die Dienstplanerstellung war zeitaufwändig (ca. vier Stunden im Monat) und wurde vom Klienten als umständlich wahrgenommen.

3.3. Soll-Prozess

Ein möglicher Soll-Prozess könnte folgendermaßen aussehen: Der Klient gibt auf einer Webseite die Dienstzeiten für den kommenden Monat ein. Per Knopfdruck wird das Team benachrichtigt und darum gebeten, die freien Tage auf einer Webseite (Link ist in der Nachricht enthalten) einzugeben. Sobald alle Assistenten ihre Eingaben getätigt haben können dem Klienten mit Hilfe eines Algorithmus mehrere Dienstplan-Vorschläge unterbreitet werden. Der Dienstplan kann vom Klienten manuell editiert werden und nach den eigenen Vorlieben (die bei der automatischen Erstellung schon größtenteils berücksichtigt wurden) angepasst werden. Der Klient gibt den Dienstplan frei und benachrichtigt sein Team, dass ein neuer Dienstplan verfügbar ist, der online eingesehen werden kann. Der Zeitaufwand für die Assistenten bleibt unverändert. Der Zeitaufwand für den Klienten sinkt auf wenige Minuten.

Hier wird ein Subjekt-Interaktions-Diagramm ergänzt.

3.4. Benutzergeschichten

„Benutzergeschichten beschreiben Anforderungen aus der Sicht des Endanwenders. Sie bestehen aus einem Namen, aus einer kurzen textuellen Beschreibung der Anforderung und einer Reihe von Akzeptanzkriterien. Letztere halten fest, welche Kriterien erfüllt sein müssen, damit die Anforderung als erfolgreich realisiert gilt.“ [Pichler 2008, S. 46]

Wichtig ist, dass die Anforderung einen „konkreten und sichtbaren Mehrwert für den Kunden“ besitzt. [Wirdemann 2011, S. 52]

Benutzergeschichten haben ein festes Muster. Es hat sich für die Beschreibung folgendes Muster bewährt [Wirdemann 2011, S. 59]:

Als <Benutzerrolle> will ich <das Ziel> [,so dass <Grund für das Ziel>]

Aus den Gesprächen mit dem Klienten und der Formulierung des Soll-Prozesses sind einige Benutzergeschichten abgeleitet worden:



3. Anforderungsermittlung

Standard-Dienstzeiten

Als Klient möchte ich Standard-Dienstzeiten angeben können.

Akzeptanzkriterien:

- Der Klient kann für jeden Tag der Woche einen Standard-Dienstbeginn und ein Standard-Dienstende festlegen.
- Die Standard-Dienstzeiten können gespeichert und wieder geladen werden.
- Bei der Erstellung eines Monatsplans werden initial die Standard-Dienstzeiten eingetragen.

Angabe Dienstzeiten

Als Klient möchte ich für den nächsten Monat vorausplanen können, wie die Dienstzeiten sind.

Akzeptanzkriterien:

- Es ist möglich einen Monat auszuwählen.
- Initial werden die Standard-Dienstzeiten im Monats-Plan eingetragen.
- Es ist möglich die Dienstzeiten für den nächsten Monat festzulegen.
- Die Dienstzeiten für den nächsten Monat können gespeichert und wieder geladen werden.

Öffentliche Bemerkungen (tagesbezogen)

Als Klient möchte ich Bemerkungen zum Tag erstellen können, die für jeden (Klient und Assistenten) sichtbar sind.

Akzeptanzkriterien:

- Es ist möglich öffentliche Bemerkungen pro Tag zu erstellen.
- Die öffentlichen Bemerkungen können gespeichert und wieder geladen werden.
- Die öffentlichen Bemerkungen des Klienten werden den Assistenten bei der Eingabe möglicher Termine angezeigt, weil die Bemerkungen Einfluss auf die Planung haben könnten.
- Die öffentlichen Bemerkungen werden im Dienstplan angezeigt.



3. Anforderungsermittlung

Private Bemerkungen (tagesbezogen)

Als Klient möchte ich Bemerkungen zum Tag erstellen können, die nur für mich sichtbar sind.

Akzeptanzkriterien:

- Es ist möglich private Bemerkungen pro Tag zu erstellen, die nur für den Klienten sichtbar sind.
- Die privaten Bemerkungen können gespeichert und wieder geladen werden.
- Die privaten Bemerkungen werden dem Klienten bei der Dienstplanerstellung angezeigt.

Öffentliche Bemerkungen (monatsbezogen)

Als Klient möchte ich monatsbezogene Bemerkungen zum Dienstplan eingeben können, die für jeden (Klient und Assistenten) sichtbar sind.

Akzeptanzkriterien:

- Der Klient hat eine Möglichkeit monatsbezogene Bemerkungen einzugeben.
- Die Bemerkungen können gespeichert und wieder geladen werden.
- Die Bemerkungen werden den Assistenten bei der Eingabe möglicher Termine angezeigt.

Verwaltung des Teams

Als Klient möchte ich mein Team verwalten.

Akzeptanzkriterien:

- Es gibt eine Übersicht der Team-Mitglieder.
- Es ist möglich Team-Mitglieder hinzuzufügen.
- Es ist möglich die  Freundschaften der Team-Mitglieder zu editieren.
- Es ist möglich Team-Mitglieder aus dem Team zu entfernen.
- Es ist möglich das Team abzuspeichern und zu einem späteren Zeitpunkt wieder zu laden.



3. Anforderungsermittlung

Bewertung der Assistenten

Als Klient möchte ich die Assistenten bewerten, um bestimmte Assistenten bei der Dienstplanerstellung zu favorisieren.

Akzeptanzkriterien:

- Es ist möglich jedem Assistenten in der Team-Verwaltung einen Priorisierungswert zuzuweisen.
- Der Algorithmus für die Dienstplanerstellung greift auf den Priorisierungswert zu und berücksichtigt diesen.

Berücksichtigung der Wochentage

Als Klient möchte ich für jeden Assistenten die Möglichkeit haben, Vorlieben für bestimmte Wochentage eingeben können.

Akzeptanzkriterien:



- Es ist möglich jedem Assistenten in der Team-Verwaltung favorisierte Wochentage zuzuweisen.
- Der Algorithmus für die Dienstplanerstellung greift auf die favorisierten Wochentage zu und berücksichtigt diese.

Team benachrichtigen

Als Klient möchte ich eine Nachricht an mein Team verschicken können mit der Bitte, freie Termine einzutragen.

Akzeptanzkriterien:

- In der Team-Verwaltung gibt es eine Möglichkeit Kontaktdaten für die Assistenten zu hinterlegen.
- Der Klient hat die Möglichkeit per Knopfdruck sein Team zu benachrichtigen. In dieser Nachricht ist unter anderem ein automatisch generierter Text mit der Bitte, die möglichen Termine einzutragen, enthalten.



3. Anforderungsermittlung

Dienstplan-Vorschläge

Als Klient möchte ich mehrere Vorschläge für einen Dienstplan bekommen, die ich manuell anpassen kann.

Akzeptanzkriterien:

- Es gibt einen Algorithmus, der automatisch unter Berücksichtigung der Verfügbarkeiten der Assistenten und der Vorlieben des Klienten einen Dienstplan erstellt, der die Stundenkontingente berücksichtigt.
- Der vom Algorithmus erstellte Dienstplan wird dem Klienten übersichtlich in einer Tabelle angezeigt.
- Wenn der Klient erneut einen Dienstplan anfordert, läuft der Algorithmus erneut und ein neuer Dienstplanvorschlag (der vom alten abweicht) wird generiert und dargestellt.

Manuelle Anpassung des Dienstplans

Als Klient möchte ich die Möglichkeit haben, den Dienstplan nach meinen Wünschen anzupassen.

Akzeptanzkriterium: Es ist möglich den, vom Algorithmus erstellten, Dienstplan manuell zu verändern.

Stundenverteilung bei Dienstplanerstellung

Als Klient möchte ich bei der manuellen Nachbearbeitung des Dienstplanes die Stundenverteilung sehen können.

Akzeptanzkriterien:

- Es ist möglich in einer Tabelle zu sehen, wie die Arbeitsstunden des Monats auf die Assistenten verteilt werden.
- Es wird dargestellt, wieviel Stunden ein Assistent arbeiten müsste. Diese Daten kommen aus der Team-Verwaltung.
- Die Differenz zwischen Soll- und Ist-Arbeitsstunden pro Assistent wird berechnet.
- Es wird grafisch in 3 Kategorien (grün, gelb, rot) dargestellt, wie gut das Stundenkontingent eines Assistenten ausgeschöpft ist.



3. Anforderungsermittlung

Eingabe möglicher Termine

Als Assistent möchte ich angeben können, wann ich Zeit habe zu arbeiten.

Akzeptanzkriterien:



- Der Assistent kann in einem Kalender einen Monat auswählen.
- Der Assistent kann mögliche Termine für einen Monat eintragen.
- Die Eingaben lassen sich speichern und zu einem späteren Zeitpunkt editieren.

Termine favorisieren

Als Assistent möchte ich eine Möglichkeit haben Termine zu favorisieren.

Akzeptanzkriterien:



- Der Assistent kann neben „Dienst möglich“ und „Dienst unmöglich“ auch ein „Dienst zur Not möglich“ auswählen.
- Der Algorithmus zur Dienstplanerstellung berücksichtigt diese Abstufung der Eingaben.

Bemerkungen der Assistenten

Als Assistent möchte ich bei der Angabe von Terminen auch Bemerkungen machen können, um weitere Informationen zu transportieren.

Akzeptanzkriterien:

- Neben den Termineingaben kann der Assistent auch Bemerkungen machen.
- Der Klient bekommt die Bemerkungen bei der Dienstplanerstellung angezeigt.

Ausschöpfung Stundenkontingent

Als Assistent möchte ich mein Stundenkontingent möglichst gut ausgeschöpft bekommen.

Akzeptanzkriterien:

- Der Algorithmus zur Dienstplanerstellung berücksichtigt die Stundenkontingente und erstellt einen Dienstplan, bei dem alle Assistenten ihre Stundenkontingente möglichst gleichmäßig ausgeschöpft bekommen.





3. Anforderungsermittlung

- In einer Tabelle kann man sehen, wie gleichmäßig die Stundenkontigente der Assistenten ausgeschöpft sind.



Ansicht Dienst- und Bereitschaftszeiten

Als Assistent möchte ich sehen, wann ich Dienst oder Bereitschaft habe.

Akzeptanzkriterien:

- Der Assistent kann den gesamten Dienstplan aller Assistenten eines Monats einsehen.
- Im Dienstplan sind die Dienste und Bereitschaften des angemeldeten Assistenten farblich hinterlegt.
- In einer gesonderten Übersicht sieht der Assistent nur seine Dienst- und Bereitschaftszeiten.

Letzte Änderung Dienstplan

Als Anwender (Klient oder Assistent) möchte ich sehen, wann der Dienstplan zum letzten Mal geändert wurde, um zu überprüfen, ob es eine Änderung gegeben hat.

Akzeptanzkriterien:

- Wenn der Dienstplan gespeichert wird, wird das Datum und die Uhrzeit protokolliert.
- Das Datum und die Uhrzeit der letzten Änderung wird beim Dienstplan angezeigt.



Dokumentation

Als Anwender möchte ich eine Dokumentation des Systems zur Verfügung haben, um mich gut in das System einarbeiten zu können.

Akzeptanzkriterien:

- Es steht eine Dokumentation zur Verfügung.
- Die Dokumentation wird abhängig vom Typ des angemeldeten Benutzers (Klient/Assistent) erzeugt.
- Es wird nur die Dokumentation der tatsächlich verwendeten Funktionen angezeigt.



3. Anforderungsermittlung

Feedback

Als Anwender möchte ich Feedback an den Entwickler senden können.

Akzeptanzkriterien:

- Der Anwender kann Feedback eingeben und senden.
- Der Anwender kann eine E-Mail-Adresse für eine Antwort eingeben.
- Das Feld für die Antwortadresse wird nach Möglichkeit automatisch ausgefüllt. Die benötigen Daten werden aus den Einstellungen (für den Klienten) oder der Team-Verwaltung (für die Assistenten) ausgelesen.

3.5. Abgeleitete Anforderungen

Aus den Benutzergeschichten ergeben sich einige abgeleitete Anforderungen, die zwar nicht explizit erwähnt wurden, aber für eine technische Umsetzung zwingend notwendig sind.

Plattformunabhängigkeit

Die Software-Lösung soll sich auf unterschiedlichen Gerät-Kategorien (zum Beispiel Handy, Tablet, Desktop) bedienen lassen können. Weiterhin sollen die größten Betriebssysteme (Windows, Linux, MacOS X, Android, iOS) unterstützt werden.

Serverbasierte Anwendung



Klient und Assistenten möchten an verschiedenen Orten zu verschiedenen Zeiten unabhängig voneinander das System bedienen. Um diese Bedingungen zu erfüllen, bietet sich eine serverbasierte Anwendung an.

Benutzer-Verwaltung



Der Klient und die Assistent haben unterschiedliche Interessen am Assistenzplaner. Der Klient möchte unter anderem den Dienstplan erstellen und sein Team verwalten, die Assistenten hingegen möchten mögliche Termine eintragen und den fertigen Dienstplan einsehen können.

Jeder Benutzer möchte seine Daten individuell eingeben können. Die hinterlegten Daten sollten nicht für jedermann einsehbar sein. Daher bekommt jeder Benutzer



3. Anforderungsermittlung

eigene Zugangsdaten, bestehend aus ID und individuellem Passwort. Bei der Anmeldung am System wird registriert, ob es sich um einen Klienten oder einen Assistenten handelt. Dementsprechend variiert der angezeigte Inhalt.

Aufgaben-Verwaltung



Eine Ergänzung zu der Dienstplanerstellung wäre eine Aufgaben-Verwaltung. Der Klient kann Aufgaben mit Fälligkeitsdatum und eventuellen Wiederholungen definieren. Klient und Assistenzen können die Aufgaben einsehen und als erledigt markieren.



4. Marktanalyse

In diesem Kapitel wird eine Marktanalyse durchgeführt. Ziel ist es herauszufinden, ob auf dem Markt eine Lösung verfügbar ist, die alle Anforderungen (siehe Kapitel 3) berücksichtigen.

Die meisten Anwendungen zur Dienstplanerstellung sind für die Verwendung in Krankenhäusern, Einzelhandel, Restaurant oder Clubs optimiert.



4.1. Dienstplan in Excel

Im Internet kann man Menge Vorlagen für die Erstellung von Dienstplänen mit Hilfe der Tabellenkalkulation **Excel**⁴ finden. Die Zusammenarbeit Klient und Assistent über das Internet gestaltet sich schwierig. Der Klient kann Dokumente freigeben und damit den Assistenten die Möglichkeit geben, ihre Termine einzutragen. Allerdings haben sie damit auch Einsicht und Zugriff auf alles, was sie nicht sehen sollten. Es gibt nicht die Möglichkeit einzelne Bereiche des Dokumentes freizugeben.

4.2. Offline-Anwendungen

Es gibt Programme zur Dienstplanerstellung, die ohne Internet-Anbindung arbeiten. Hier müsste der Klient die Eingabe der möglichen Termine der Assistenten selbstständig vornehmen.

Ein Beispiel für solch eine Anwendung ist **QTime**⁵. Aber schon der Blick auf die Screenshots Beispiel: Abbildung 4.1) lässt erahnen, dass die Software sehr komplex ist und für jemanden, der sie nur einmal im Monat verwenden möchte, zuviel Einarbeitung abverlangt.

⁴ <http://office.microsoft.com/de-de/excel/> - zuletzt abgerufen am 23.08.2014

⁵ <http://www.qtime.de/kostenlos/dienstplan-erstellung.html> - zuletzt abgerufen am 23.08.2014

ASSISTENZPLANER

Serverbasierte Software zur Koordinierung der Assistenten für behinderte Menschen



4. Marktanalyse

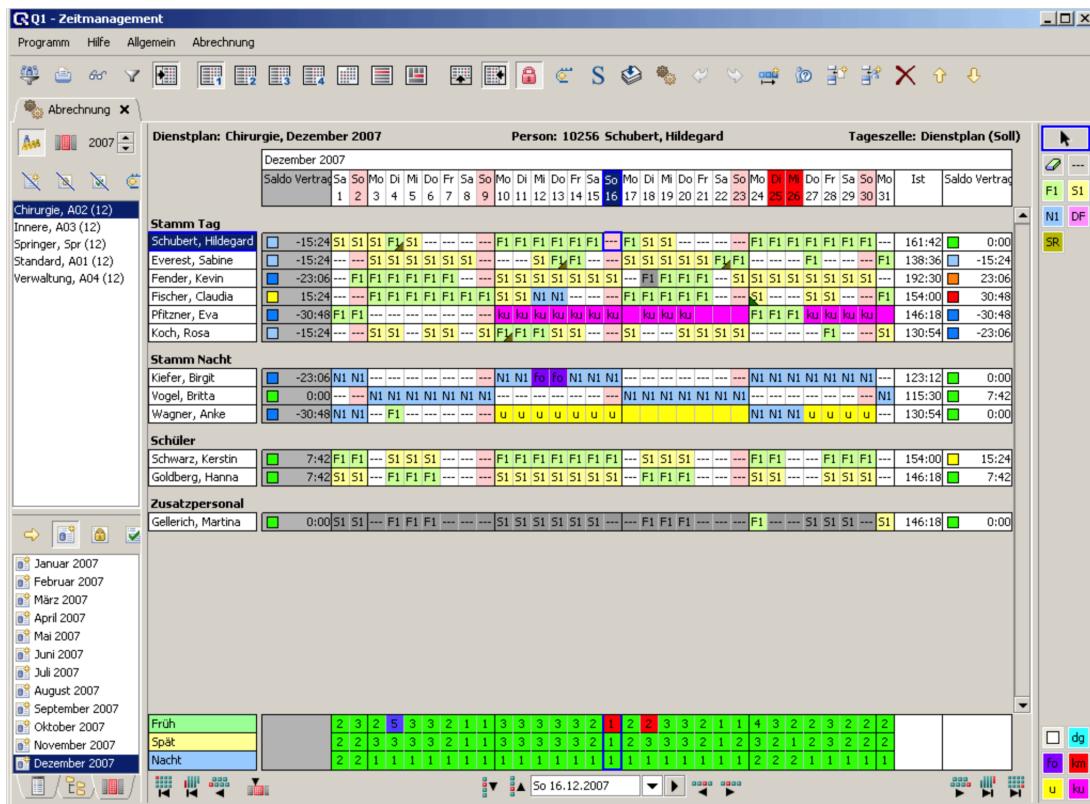


Abbildung 4.1.: Screenshot von QTime: Komplexe Software

4.3. Online-Anwendungen

Es gibt einige Anwendungen im Internet, die es ermöglichen, Dienstpläne zu erstellen.

Beispiele hierfür sind:

- Kostenloser Dienstplan⁶
- Schichtplaner⁷
- Papershift⁸
- EasyPep⁹
- ShiftJuggler¹⁰

⁶ <http://kostenloser-dienstplan.de> - zuletzt abgerufen am 23.08.2014

⁷ <http://schichtplaner-online.de/index.php/de/> - zuletzt abgerufen am 23.08.2014

⁸ <http://www.papershift.com> - zuletzt abgerufen am 23.08.2014

⁹ <https://easypep.de> - zuletzt abgerufen am 23.08.2014

¹⁰ <https://www.shiftjuggler.com/de/> - zuletzt abgerufen am 23.08.2014



4. Marktanalyse

Keine dieser Anwendungen unterstützt das Bereitschaftskonzept. Einige sind in der Lage Stundenkontingente zu berücksichtigen. Bei allen wird der Dienstplan manuell erstellt. Bei keiner können persönliche Vorlieben oder allgemein Regeln für die Dienstplanerstellung festgelegt werden. Alle waren auf die reine Erstellung eines Dienstplans ausgelegt, Erweiterungen wie Aufgaben-Verwaltung waren nicht vorgesehen.

Zusammenfassend lässt sich festhalten, dass es keine bestehende Lösung gibt, die alle formulierten Anforderungen erfüllen kann. Es gibt auch keine Anknüpfungsmöglichkeiten oder Wiederverwendungsmöglichkeiten, da die meisten Anbieter ein finanzielles Interesse haben und den Code ihrer Lösungen nicht offen gelegt haben.



5. Software-Design

Alle Grafiken dieses Kapitels sind vorläufig und werden noch überarbeitet.

In diesem Kapitel werden die Anforderungen aus dem vorletzten Kapitel in einem SysML Anforderungsdiagramm dargestellt. Daraufhin wird ein Software-Design in UML entworfen, das den Anforderungen gerecht wird.

5.1. SysML Anforderungsdiagramm

Mit SysML lassen sich Systeme, bestehend aus Hard- und Software, modellieren. Da in der vorliegenden Arbeit eine Lösung entstanden ist, die ausschließlich aus Software besteht, wurde nur das Anforderungsdiagramm der SysML verwendet. Alles andere wurde in UML modelliert.

Die Anforderung „beschreibt ein oder mehrere Eigenschaften oder Verhaltensweisen eines Systems, die stets erfüllt sein müssen.“ [Weilkiens 2014, S. 315] Im Anforderungsdiagramm wird jede Anforderung als Box dargestellt, in deren oberen Bereich der Stereotyp «requirement» steht. Die Anforderungen können Beziehungen untereinander haben. Nachfolgend sind die verwendeten Beziehungen dargestellt.

Enthältbeziehung

Die Enthältbeziehung „beschreibt, dass eine Anforderung in einer anderen enthalten ist.“ [Weilkiens 2014, S. 318] Dargestellt wird diese Beziehung durch eine Verbindung zweier Anforderungen, wobei an einem Ende der Verbindung ein Kreis mit einem Kreuz ist. In Abbildung 5.1 ist ein Beispiel dargestellt. Die Anforderungen „Aufgaben als erledigt markieren“ und „Aufgaben erstellen“ sind in der Anforderung „Aufgabenverwaltung“ enthalten.

Erfüllungsbeziehung

Die Erfüllungsbeziehung „beschreibt, dass ein Element eine Anforderung erfüllt.“ [Weilkiens 2014, S. 319] Dargestellt wird diese Beziehung durch einen Pfeil, der mit



5. Software-Design

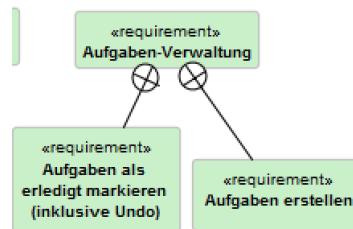


Abbildung 5.1.: SysML Anforderungsdiagramm - Enthältbeziehung

dem Stereotyp «satisfy» beschriftet ist. In Abbildung 5.2 ist ein Beispiel dargestellt. Die Anforderung „Kalender“ erfüllt die Anforderung „Angabe möglicher Termine“.



Abbildung 5.2.: SysML Anforderungsdiagramm - Erfüllungsbeziehung

Verfeinerungsbeziehung

Die Verfeinerungsbeziehung „beschreibt, dass ein Modellelement die Eigenschaften einer Anforderung detaillierter darstellt.“ [Weilkiens 2014, S. 325] Dargestellt wird diese Beziehung durch einen Pfeil, der mit dem Stereotyp «refine» beschriftet ist. In Abbildung 5.3 ist ein Beispiel dargestellt. Die Anforderung „Wiederkehrende Aufgaben erstellen“ verfeinert die Anforderung „Aufgabe erstellen“.

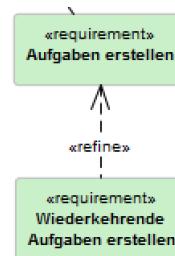


Abbildung 5.3.: SysML Anforderungsdiagramm - Verfeinerungsbeziehung

Anforderungsdiagramm

In Abbilung 5.4 ist das Anforderungsdiagramm mit allen Anforderungen und den Beziehungen untereinander zu sehen.

Es ist deutlich eine Clusterung zu erkennen. Die Anforderungen „Erstellung Dienstplan“, „Erstellung Montsplan“, „Team-Verwaltung“, „Benutzerverwaltung“ und „Aufgabenverwaltung“ sind die zentralen Anforderungen, die weitere Anforderungen enthalten.



5. Software-Design

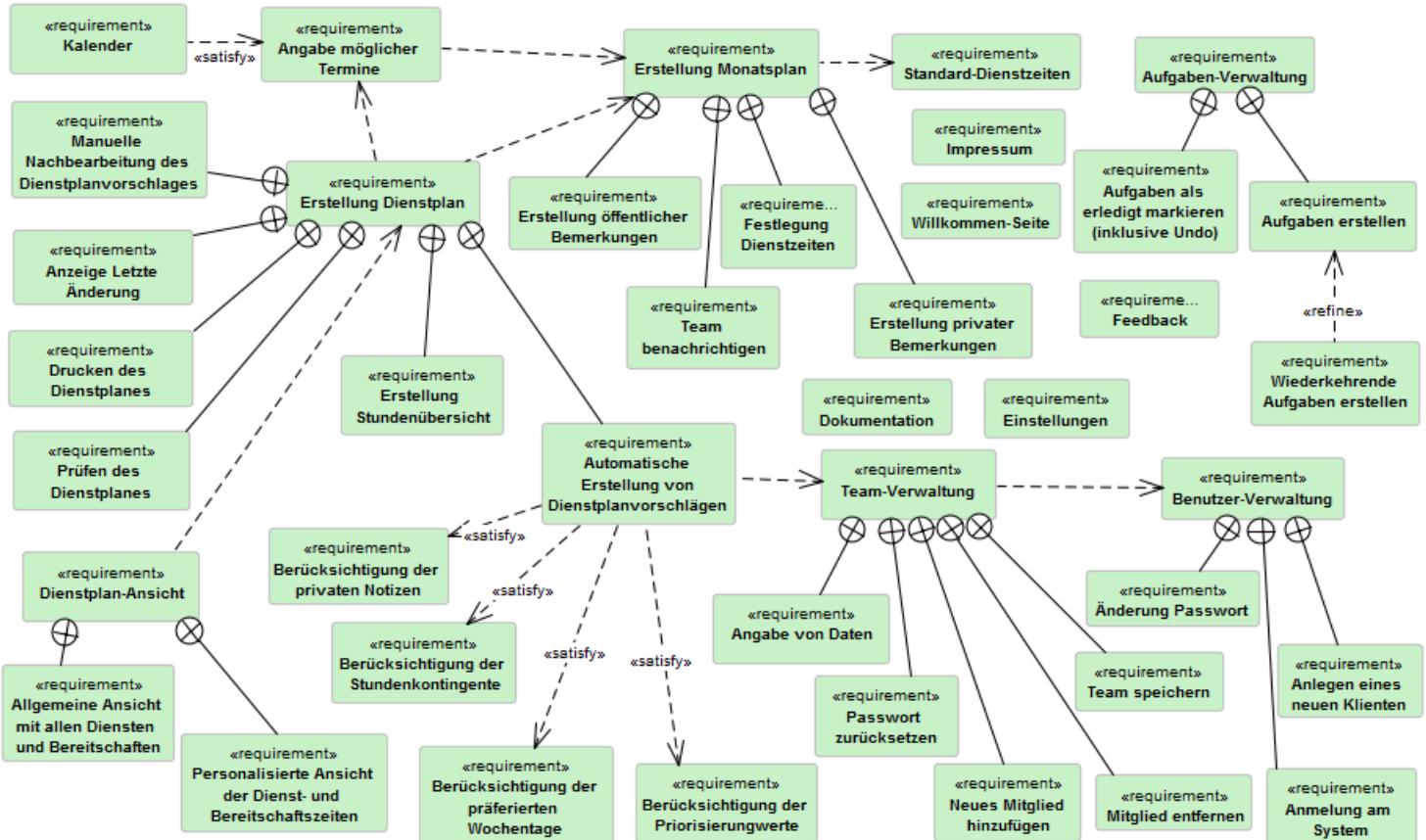


Abbildung 5.4.: SysML Anforderungsdiagramm

5.2. UML Komponentendiagramm

Mit dem Komponentendiagramm der UML kann man eine Grobarchitektur der Software darstellen. Die Diagrammform ist hilfreich um sich einen Überblick zu verschaffen. Nutzt eine Komponente Funktionalität einer anderen Komponente, so kann man dies durch Interfaces¹¹ darstellen

Das Anforderungsdiagramm ist in Deutsch gehalten, da es noch unabhängig von einer Programmiersprache ist. Software schreibt der Autor ausschließlich englischsprachig. Nachfolgend sind die zentralen Anforderungen und die daraus abgeleiteten Paketnamen dargestellt:

- Erstellung Dienstplan ⇒ Roster
- Erstellung Monatsplan ⇒ MonthOrganisation
- Team-Verwaltung und Benutzer-Verwaltung ⇒ TeamOrganisation

¹¹Der Autor bevorzugt den englischen Begriff Interface (deutsch: Zwischenstück), weil er ihn treffender findet als das häufig verwendete Wort „Schnittstelle“.



5. Software-Design

- Aufgaben-Verwaltung ⇒ ToDoManager

Das Komponentendiagramm ist in Abbildung 5.5 zu sehen.

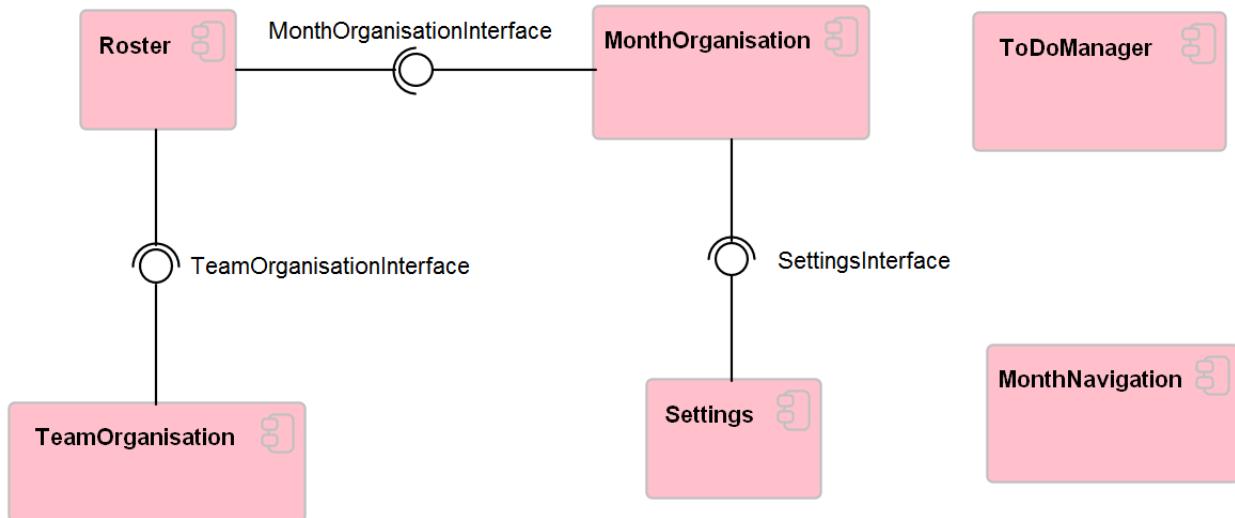


Abbildung 5.5.: UML Komponentendiagramm

Ziel der Modellierung war es, die Komponenten voneinander so unabhängig wie möglich zu machen. Zwei Komponenten (**ToDoManager** und **MonthNavigation**) sind völlig autark. Die restlichen vier Komponenten sind durch schlanke Interfaces verbunden. Beim **MonthOrganisationInterface** und **SettingsInterface** ist nur eine Methode zu implementieren. Beim **TeamOrganisationInterface** sind fünf Methoden zu implementieren.

5.3. UML Klassendiagramme

In diesem Abschnitt werden die Komponenten aus Abbildung 5.5 näher beleuchtet und die darin liegenden Klassen gezeigt.

Komponente MonthNavigation

Die Komponente **MonthNavigation** (siehe Abbildung 5.6) enthält nur die Klasse **MonthNavigation**.

5. Software-Design

Abbildung 5.6.: Klassendiagramm der Komponente **MonthNavigation**

Die Klasse **MonthNavigation** ist für die Erstellung der Monats-Navigation (Implementierung siehe Abschnitt 6.7) zuständig.

Komponente **MonthOrganisation**

Die Komponente **MonthOrganisation** (siehe Abbildung 5.7) enthält die Klassen **MonthPlan**, **AssistanceInput**, **WorkingTimes** und **Day**.



5. Software-Design

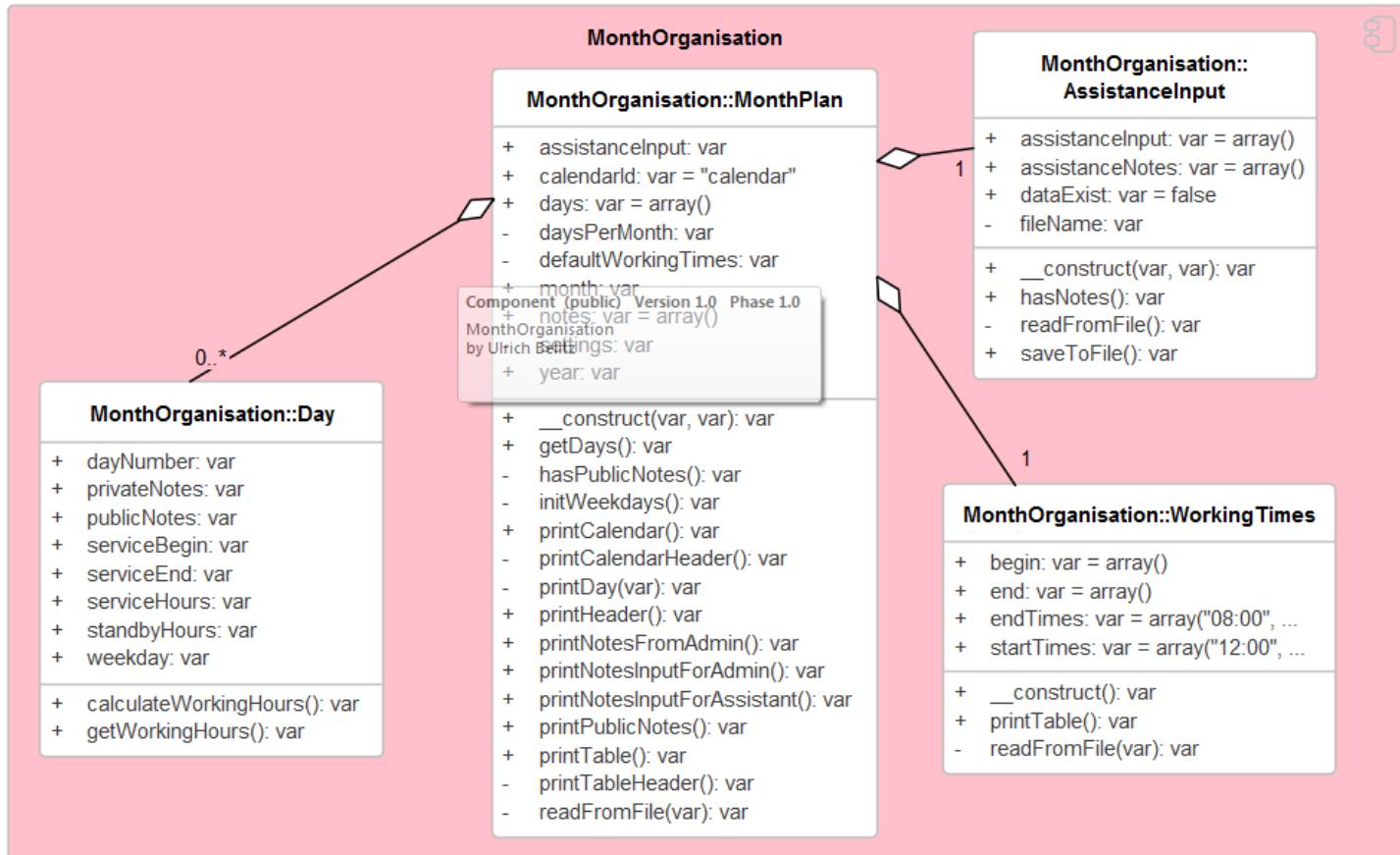


Abbildung 5.7.: Klassendiagramm der Komponente MonthOrganisation

Zentrale Klasse ist **MonthPlan**, die Instanzen von **Day** weiteren Klassen hat. Mit der Klasse **MonthPlan** werden alle Daten des Monatplans (Implementierung siehe Abschnitt 6.13) verwaltet. Die Kalendereingaben (Implementierung siehe Abschnitt 6.14) der Assistenten werden in der Klasse **AssistanceInput** hinterlegt. Die Standardarbeitszeiten (Implementierung siehe Abschnitt 6.11) werden von der Klasse **WorkingTimes** bereitgestellt. In der Klasse **Day** werden alle tagesbezogenen Daten gebündelt.

Komponente Roster

Die Komponente **Roster** (siehe Abbildung 5.8) enthält nur die Klasse **Roster**.



5. Software-Design



Abbildung 5.8.: Klassendiagramm der Komponente Roster

Die Klasse **Roster** wird bei der Dienstplanerstellung (Implementierung siehe Abschnitt 6.15) verwendet.

Komponente Settings

Die Komponente **Settings** (siehe Abbildung 5.9) enthält die Klassen **Passwords** und **Settings**.



5. Software-Design

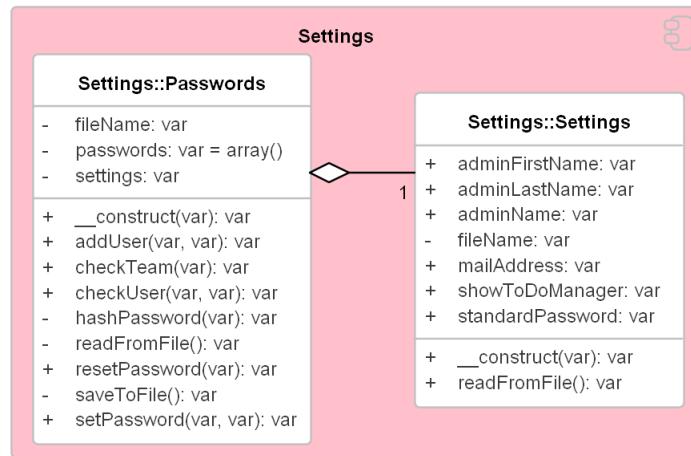


Abbildung 5.9.: Klassendiagramm der Komponente **Settings**

Mit der Klasse **Settings** werden die Einstellungen (Implementierung siehe Abschnitt 6.10) des Assistenzplaners verwaltet. In der Klasse **Passwords** sind alle Funktionen vorhanden, die für die Benutzerverwaltung (Implementierung siehe Abschnitt 6.4) benötigt werden.

Komponente TeamOrganisation

Die Komponente **TeamOrganisation** (siehe Abbildung 5.10) enthält die Klassen **Team** und **TeamMember**.



5. Software-Design

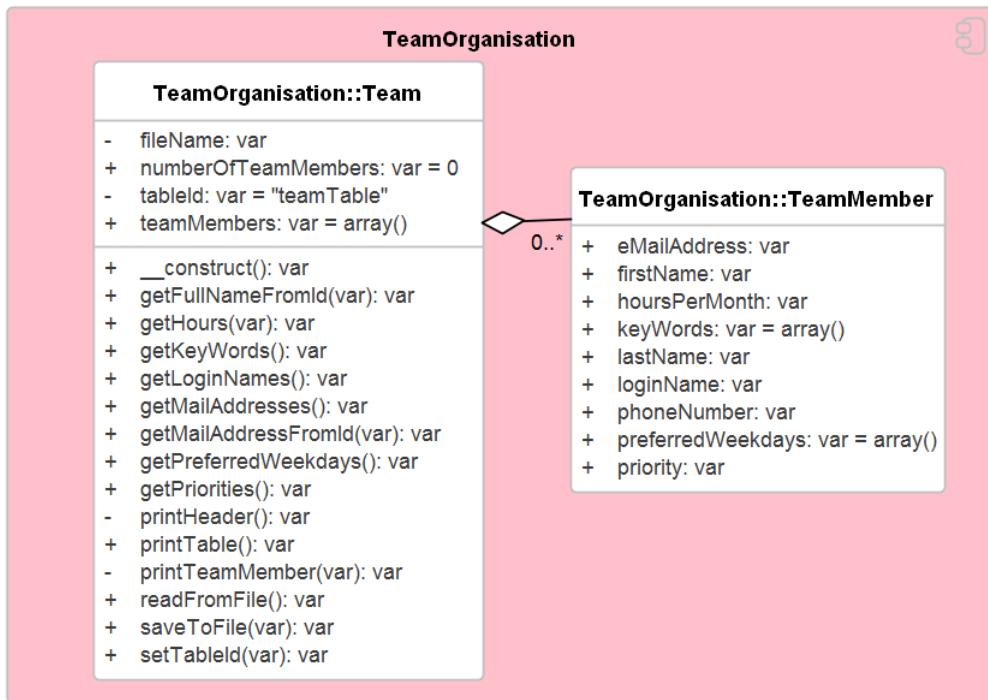


Abbildung 5.10.: Klassendiagramm der Komponente **TeamOrganisation**

In der Klasse **Team** ist die Funktionalität für die Team-Verwaltung (Implementierung siehe Abschnitt 6.12) implementiert. **Team** hat soviele Instanzen von **TeamMember**, wie das Assistenz-Team Mitglieder hat. Die Klasse **TeamMember** ist eine reine Datenhaltungsklasse¹² ohne Funktionalität.

Komponente **ToDoManager**

Die Komponente **ToDoManager** (siehe Abbildung 5.11) enthält die Klassen **ToDoManager** und **ToDoItem**.

¹²Alle Member sind **public**.



5. Software-Design

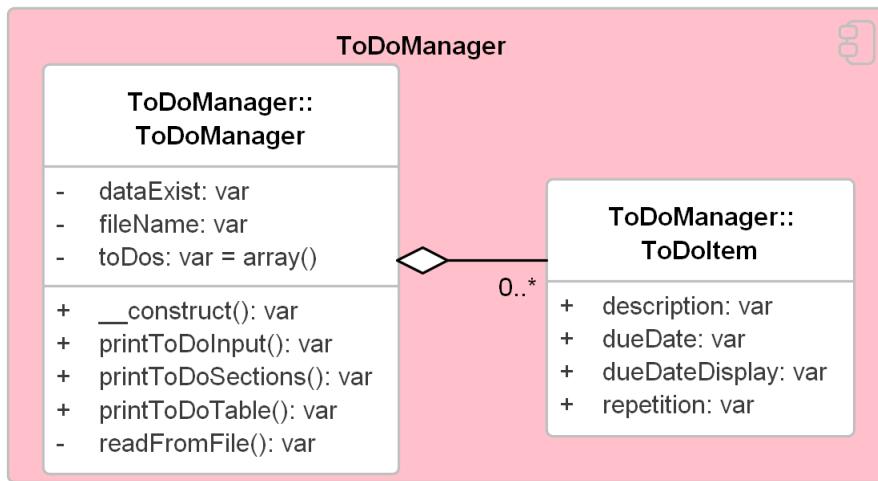


Abbildung 5.11.: Klassendiagramm der Komponente ToDoManager

In der Klasse **ToDoManager** ist die Funktionalität für die Aufgaben-Verwaltung (Implementierung siehe Abschnitt 6.17) implementiert. **ToDoManager** hat soviele Instanzen von **ToDoItem**, wie Aufgaben in der Aufgabenliste hinterlegt sind. Die Klasse **ToDoItem** ist eine reine Datenhaltungsklasse ohne Funktionalität.



6. Implementierung

In diesem Kapitel wird dargestellt, wie die Anforderungen und das Software-Design konkret implementiert wurden. Die Ergebnisse der Implementierung werden kompakt dargestellt. In den Abschnitten „Implementierungsdetails“ werden weiterführende Informationen dargestellt, die für das Gesamtverständnis der Ausarbeitung nicht wichtig sind und übersprungen werden können. Der Schwerpunkt des Kapitels liegt auf der Darstellung des Algorithmus zur Dienstplanerstellung (siehe Abschnitt 6.16).

6.1. Wahl der Programmiersprachen

Für die Implementierung ist es notwendig sich auf Programmiersprachen festzulegen.

Server

Bei der Serverseite ist die Wahl der Programmiersprache auf PHP gefallen. Hauptgründe hierfür waren: die **hohe Verbreitung**¹³ von über 80% und die Tatsache, dass PHP bei den meisten Webhostern vorinstalliert ist. Weitere Gründe [Theis 2010, S. 16] sind:

- Einfache Entwicklung von Programmen
- Unterstützung verschiedener Plattformen
- Leichte Integration in Apache (weitverbreiteter Webserver)

Client

Der Server generiert mit Hilfe von PHP den HTML-Code, der an den Client (Browser) gesendet wird. Im HTML wird ausschließlich die Struktur der angezeigten Seite definiert. Das Aussehen wird mit Hilfe von CSS bestimmt. Um die Funktionalität auf der Seite des Clients zu erweitern, kommt JavaScript zum Einsatz.

¹³http://w3techs.com/technologies/history_overview/programming_language - zuletzt abgerufen am 23.08.2014



6.2. Einheitliches Layout

Mit Hilfe von CSS wird die Darstellung des Assistenzplaners konsistent definiert. Es gibt ein globales Stylesheet, was in allen Webseiten verwendet wird.

Implementierungsdetails

In der Datei `global.css` sind alle globalen Styles definiert. An manchen Stellen des Assistenzplaners wird das globale Stylesheet durch spezifische Stylesheets ergänzt. Zum Beispiel kommt bei der Dokumentation (siehe Abschnitt 6.19) das Stylesheet `documentation.css` zum Einsatz.

6.3. Persistenz der Daten

Alles was persistent gespeichert werden muss, wird in textbasierten Dateien (*.txt) abgelegt. Im Vergleich zu Datenbanken hat man folgende Vorteile:

- Daten können leicht zwischen Produktiv-System und Test-System ausgetauscht werden.
- Datenbank-Verwaltung entfällt.
- Es werden keine Datenbank-Anforderungen an den Server gestellt.

Implementierungsdetails

Eine potentielle Fehlerquelle beim Gebrauch von textbasierten Daten ist der gleichzeitige Schreibzugriff zweier Benutzer auf eine Datei. Das ist derzeit nur bei den Kalender-Eingaben der Assistenten (siehe Abschnitt 6.14) möglich. Um den gleichzeitigen Zugriff zu verhindern wird eine Art Semaphor¹⁴ eingesetzt. Unter PHP gibt es keine echten Semaphoren, aber es gibt die Möglichkeit auf eine Datei sperrend zuzugreifen. So wird verhindert, dass zwei Prozesse gleichzeitig in eine Datei schreiben, falls zwei Assistenten gleichzeitig ihre Daten speichern wollen.

Es gibt einige PHP-Scripte zum Speichern von Daten. Bei allen werden per POST-Parameter die zu speichernden Inhalte übermittelt, ein File schreibend geöffnet und der übertragene Inhalt geschrieben. Die PHP-Scripte, die für das Speichern zuständig sind, heißen `calendarSaver.php`, `changePasswordSaver.php`, `defaultTimesSaver.php`, `monthPlanSaver.php`, `rosterSaver.php`, `settingsSaver.php` und `ToDoSaver.php`.

¹⁴[http://de.wikipedia.org/wiki/Semaphor_\(Informatik\)](http://de.wikipedia.org/wiki/Semaphor_(Informatik)) - zuletzt abgerufen am 05.09.2014



6.4. Benutzer-Verwaltung

Damit der Klient und die Assistenten benutzerspezifische Eingaben tätigen können ist eine Verwaltung der Benutzer notwendig. Jeder Benutzer des Assistenzplaners kann sich mit eigener Kennung und Passwort anmelden (siehe Abbildung 6.1). Es gibt zwei Rollen von Benutzern: die Klienten und die Assistenten. Die Klienten lassen beim Anmelden das Eingabefeld „Assistent“ leer. Die Klienten können sich ihren Zugang selbstständig anlegen (siehe Abschnitt 6.8). Die Zugänge der Assistenten werden von den Klienten verwaltet.¹⁵ Klienten haben die Möglichkeit Assistenten zum Team hinzuzufügen und zu entfernen (siehe Abschnitt 6.12).

Klient:

Assistent:

Passwort:

Abbildung 6.1.: Login-Maske des Assistenzplaners

Implementierungsdetails

Das PHP-Script `login.php` ist für die Erstellung der Eingabefelder zuständig. Wenn der Benutzer auf den Button „Anmelden“ drückt, werden die Daten aus den Eingabefeldern nach einer erfolgreichen Validierung per POST an den `loginHandler.php` geschickt. Der Login-Handler verifiziert, dass der Nutzer besteht und das Passwort korrekt eingegeben wurde.

Nach der erfolgreichen Anmeldung wird eine PHP-Session gestartet. Dies geschieht mit dem PHP-Befehl:

```
session_start();
```

Im Hintergrund wird im Browser eine Session-ID erstellt und gespeichert. Ein Beispiel für solch eine Session-ID findet sich in Abbildung 6.2.

Nach der Anmeldung werden im superglobalen Array `$_SESSION` Variablen gesetzt, die hilfreich für die weitere Verwendung des Assistenzplaners sind. Diese Vorgehensweise (Start einer Session und Setzen der Session-Variablen) ist [Schlossnagle 2006, S. 404] entnommen.

¹⁵Jeder Assistent ist damit fest einem Klienten zugeteilt. Theoretisch kann ein Assistent auch in mehreren Teams Mitglied sein.



6. Implementierung

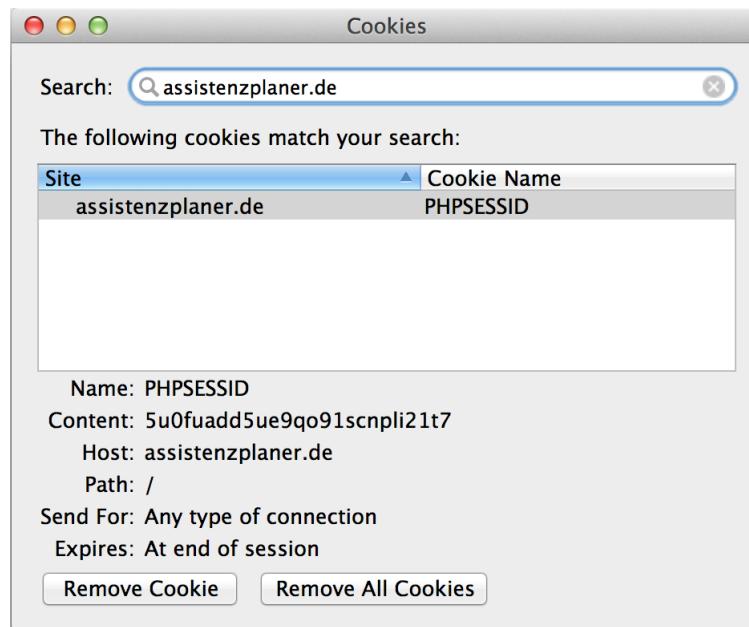


Abbildung 6.2.: Session-ID wird als Cookie im Browser gespeichert

Falls sich der Assistent mit dem Standard-Passwort angemeldet hat, so wird er zu `changePassword.php` weitergeleitet. Dort wird er aufgefordert sein Passwort zu ändern. Ansonsten wird er zur Übersicht (siehe Abschnitt 6.9) weitergeleitet. Eine weitere Möglichkeit ist, dass die Login-Seite mit dem GET-Parameter `redirect` aufgerufen wurde. Das passiert zum Beispiel, wenn der Assistent in der automatisch generierten Nachricht auf einen Link klickt, der auf folgendes Ziel verweist:

`http://assistenzplaner.de/PHP/login.php?redirect=calendarView`

Dann wird der Nutzer nach erfolgreicher Anmeldung direkt zur Kalender-Ansicht weitergeleitet.

6.5. Willkommen-Seite

Auf der **Willkommen-Seite**¹⁶ (siehe Abbildung 6.3) werden dem Interessenten Informationen über den Assistenzplaner präsentiert. Man wird gebeten einen neuen Klienten-Zugang einzurichten oder sich mit einem bestehenden Zugang anzumelden.

¹⁶ <http://www.assistenzplaner.de/master/PHP/index.php> - zuletzt abgerufen am 23.08.2014



6. Implementierung

Willkommen beim Assistenzplaner

Der Assistenzplaner ist ein Hilfsmittel für körperlich behinderte Menschen, die auf Assistenz angewiesen sind.

Mit Hilfe des Assistenzplaners ist es unter anderem möglich:

- Assistenz-Teams zu verwalten
- Dienstpläne zu erstellen
- Aufgaben der Assistenten zu verwalten

Bitte [melden Sie sich an oder legen Sie einen neuen Klienten-Zugang an](#).

Abbildung 6.3.: Willkommen-Seite

Implementierungsdetails

Die Willkommensseite wird durch das PHP-Script `index.php` generiert. Man kann zwei Aktionen durchführen:

1. Ein neues Konto anlegen ⇒ `createNewAccount.php`
2. Anmelden ⇒ `login.php`

6.6. Navigation

Die Navigation steht immer gut sichtbar rechts oben zur Verfügung. Abhängig davon, ob jemand am System angemeldet ist (siehe Abbildung 6.4) oder nicht (siehe Abbildung 6.5) stellt sie unterschiedliche Informationen dar.

Klient: Mustermann Assistent: A1 [Übersicht](#) [Passwort ändern](#) [Dokumentation](#) [Feedback](#) [Impressum](#) [Abmelden](#)

Abbildung 6.4.: Navigation bei angemeldetem Assistenten

[Willkommen](#) [Neuen Klienten-Zugang einrichten](#) [Impressum](#) [Anmelden](#)

Abbildung 6.5.: Navigation ohne Anmeldung

Implementierungsdetails

Die Navigation wird durch das PHP-Script `navigation.php` erzeugt. Dieses Script wird in allen Seiten direkt nach dem HTML-body-Tag inkludiert¹⁷. Im Script werden aus dem superglobalen Array `$_SESSION` einige Informationen ausgelesen und abhängig davon der Inhalt der Navigation variiert. In der Navigation werden Hyperlinks zu anderen PHP-Scripten dargestellt.

¹⁷PHP-Code: <?php include('navigation.php'); ?>



6.7. Monatsnavigation

An einigen Stellen (Dienstplan, Monatsplan, Kalender) möchte der Anwender zwischen den Monaten navigieren können. Mithilfe der Monatsnavigation (siehe Abbildung 6.6) kann man vom derzeit angezeigten Monat immer zum vorherigen, aktuellen und nächsten Monat navigieren. Befindet man sich zum Beispiel heute (12.09.2014) in der Navigation im Monat Januar 2015, so gelangt man mit dem Link „Vorheriger Monat“ zum Dezember 2014, mit dem Link „Aktueller Monat“ zum September 2014 und mit dem Link „Nächster Monat“ zum Februar 2015.

Vorheriger Monat Aktueller Monat Nächster Monat

Abbildung 6.6.: Monatsnavigation

Implementierungsdetails

Die Monatsnavigation wird von der PHP-Klasse `MonthNavigation` erstellt. Es reicht eine Instanz von `MonthNavigation` zu erstellen. Bei der Konstruktion muss man nur einen Pfad auf das aufrufende PHP-Script, das Jahr und den Monat angeben (siehe Listing 6.1).

Listing 6.1: Anlegen der Monatsnavigation

```

1 <?php
2 $navigation = new MonthNavigation($_SERVER['PHP_SELF']), $year,
   $month);
3 ?>

```



Voraussetzung ist, dass das aufrufende PHP-Script mit den GET-Parametern `year` und `month` aufgerufen werden kann. Das ist bei den PHP-Scripten `monthPlanView.php`, `rosterView.php` und `calendarView.php` der Fall.

6.8. Anlegen eines neuen Klienten-Zugangs

Das Anlegen eines neuen Klienten-Zugangs soll auf einfache Weise möglich sein. Es werden bei der Erstellung eines neuen Klienten-Zugangs keine persönlichen Daten erhoben.

Um einen neuen Klienten-Zugang anzulegen, wird man gebeten einen Klientennamen, ein Passwort und die Passwort-Wiederholung einzugeben (siehe Abbildung 6.7).

Bedingungen an einen neuen Klientennamen sind:



6. Implementierung

Name des Klienten:	<input type="text"/>
Passwort:	<input type="password"/>
Wiederholung Passwort:	<input type="password"/>
<input type="button" value="Klient anlegen"/>	

Abbildung 6.7.: Anlegen eines neuen Klienten mit Name und Passwort

- Er darf nicht leer sein.
- Er darf noch nicht vergeben sein.
- Er darf nur aus Buchstaben und Zahlen bestehen. Sonderzeichen und Leerzeichen sind nicht erlaubt.

Wenn eine Bedinung nicht erfüllt ist, so wird der Benutzer darauf hingewiesen. Die Inhalte der Textfelder bleibt erhalten, so dass der Benutzer das Passwort und die Wiederholung dessen nochmal eingeben muss.

Wenn alle Bedingungen erfüllt sind, bekommt der Benutzer eine Meldung, dass das Anlegen eines neuen Kontos erfolgreich war. Weiterhin wird er gebeten, sich mit dem neuen Zugang am System anzumelden.

Zu einem späteren Zeitpunkt kann der Klient in den Einstellungen personenbezogene Daten (Vorname, Nachname, E-Mail-Adresse) angeben, was aber nicht zwingend für die Benutzung des Assistenzplaners notwendig ist.

Implementierungsdetails

Das PHP-Script `createNewAccount.php` erzeugt die Seite mit den Eingabefeldern (siehe Abbildung 6.7). Drückt der Benutzer auf den Button „Klient anlegen“, so verzifert die Funktion `createNewAccount()` des JavaScripts `createNewAccount.js`, dass alle Eingaben korrekt sind und der Klientenname die geforderten Bedingungen erfüllt. Anschließend werden der Klientenname und das Passwort per POST-Parameter an das PHP-Script `createNewAccountHandler.php` geschickt.

Dieses Script legt im Ordner `Data` ein Unterordner mit dem Klientennamen an, in dem alle Daten persistent gespeichert werden. Darunter wird der Unterordner `Team` angelegt. In diesem Ordner wird in der Datei `passwords.txt` der Hash des vergebenen Passworts abgespeichert. Dabei wird die PHP-Funktion `password_hash()`¹⁸ verwendet. Selbst wenn ein Hacker die Passwort-Datei in die Hände bekommt ist

¹⁸ <http://php.net/manual/de/function.password-hash.php> - zuletzt abgerufen am 12.09.2014



6. Implementierung

sie nutzlos, da von einem Hash nicht auf das Passwort zurückgeschlossen werden kann.

6.9. Übersicht

Auf der Übersichts-Seite (siehe Abbildung 6.8) werden dem Benutzer alle verfügbaren Seiten als Hyperlink aufgelistet. Hierbei werden der Benutzertyp (Klient oder Assistent) und die Einstellungen (zum Beispiel Aufgaben-Verwaltung benutzen oder nicht) berücksichtigt.

Übersicht Assistenzplaner

[Dienst-Plan](#)
[Aufgaben](#)
[Monats-Plan](#)
[Team](#)
[Standard Dienst-Zeiten](#)
[Einstellungen](#)

Abbildung 6.8.: Übersicht des Assistenzplaners für Klienten mit Links zu allen verfügbaren Seiten

Implementierungsdetails

Die Übersicht wird von dem PHP-Script `overview.php` erzeugt. Abhängig von den Variablen, die im superglobalen Array `$_SESSION` gesetzt sind, variiert der Inhalt der dargestellten Seite. Weiterhin spielen die Einstellungen, die in der Klasse `Settings` gespeichert wurden eine Rolle.

6.10. Einstellungen



Um den Assistenzplaner zu konfigurieren gibt es die Einstellungen (siehe Abbildung 6.9), die nur für den Klienten zugänglich sind.. Hier können Vorname, Nachname und E-Mail-Adresse des Klienten gespeichert werden. Weiterhin kann hier das Standard-Passwort angesehen und editiert werden. Jeder neu angelegte Assistent kann sich mit dem Standard-Passwort beim Assistenzplaner anmelden. Weiterhin kann das Passwort eines Assistenten vom Klienten auf das Standard-Passwort zurückgesetzt werden, falls der Assistent sein Passwort vergessen hat. Zum Schluss



6. Implementierung

kann der Klient in den Einstellungen festlegen, ob er die Aufgaben-Verwaltung (siehe Abschnitt 6.17) nutzen möchte oder nicht.

Einstellungen

Beschreibung	Wert
Vorname	Max
Nachname	Mustermann
E-Mail-Adresse	max@mustermann.de
Standard-Passwort für Assistenten	Hallo123
Aufgaben-Verwaltung nutzen	<input checked="" type="checkbox"/>

Abbildung 6.9.: Einstellungsmöglichkeiten des Assistenzplaners für Klienten

Implementierungsdetails

Die Einstellungsseite wird von dem PHP-Script `settingsView.php` erstellt. Drückt der Benutzer auf den Button „Einstellungen speichern“, so wird die Funktion `saveSettings` im JavaScript `settings.js` aufgerufen. Diese Funktion parst aus den Einstellungen die Daten heraus und schickt sie per POST-Parameter an das PHP-Script `settingsSaver.php`, welches die Daten persistent in der Datei `settings.txt` im Ordner `Organization` speichert.

6.11. Standard-Dienstzeiten

Bei den Standard-Dienstzeiten (siehe Abbildung 6.10) hat der Klient die Möglichkeit für jeden Tag der Woche festzulegen, wie standardmäßig die Dienstzeiten sind. Diese Standard-Dienstzeiten werden initial im Monatsplan eingetragen, können aber vom Klienten für jeden Tag angepasst werden.

Implementierungsdetails

Die Standard-Dienstzeiten-Seite wird von dem PHP-Script `defaultTimes.php` erstellt. Drückt der Benutzer auf den Button „Speichern“, so wird die Funktion `saveDefaultTimes` im JavaScript `defaultTimes.js` aufgerufen. Diese Funktion parst aus der Standard-Dienstzeiten-Tabelle die Daten heraus und schickt sie per POST-Parameter an das PHP-Script `defaultTimesSaver.php`, welches die Daten persistent in der Datei `defaultTimes.txt` im Ordner `Organization` speichert.



6. Implementierung

Standard-Dienstzeiten

Wochentag	Dienstbeginn	Dienstende (am Folgetag)
Mo	17:00 ▲	08:00 ▲
Di	13:00 ▲	08:00 ▲
Mi	17:00 ▲	08:00 ▲
Do	13:00 ▲	08:00 ▲
Fr	14:00 ▲	13:00 ▲
Sa	13:00 ▲	13:00 ▲
So	13:00 ▲	08:00 ▲

Abbildung 6.10.: Verwaltung der Standard-Dienstzeiten.

6.12. Team-Verwaltung

Mit der Team-Verwaltung kann der Klient die Eigenschaften seiner Team-Mitglieder verwalten.

Jeder Assistent hat folgende Eigenschaften:

- Kennung - eindeutige ID, mit der sich der Assistent am System anmeldet
- Vorname
- Nachname
- Telefonnummer
- Stichwörter - werden bei der Dienstplanerstellung berücksichtigt
- Stundenkontingent - Anzahl der Stunden in einem Monat, die der Assistent zur Verfügung hat
- Priorisierung - ein Wert, um den Assistenten bei der Dienstplanerstellung zu priorisieren (je höher der Wert, desto höher die Wahrscheinlichkeit, dass der Assistent viele Dienste bekommt)
- Bevorzugte Wochentage

Die Eigenschaften werden in tabellarischer Form dargestellt (siehe Abbildung 6.11) und sind editierbar.

Neue Team-Mitglieder können mit Hilfe des Buttons „Neues Mitglied“ hinzugefügt werden. Das neue Team-Mitglied hat initial das Passwort, welches in den Einstellungen hinterlegt wurde.



6. Implementierung

Team Übersicht

Kennung	Vorname	Nachname	E-Mail Adresse	Telefonnummer	Stichwörter (getrennt durch Leerzeichen)	Stundenkontingent	Priorisierung	Bevorzugte Wochentage	Aktionen
Max	Max	Muster	max@muster.de		FCB Tatort	200	2	<input type="checkbox"/> Mo <input type="checkbox"/> Di <input type="checkbox"/> Mi <input type="checkbox"/> Do <input type="checkbox"/> Fr <input type="checkbox"/> Sa <input type="checkbox"/> So	<input type="button" value="Löschen"/> <input type="button" value="Passwort zurücksetzen"/>
Michael	Michael	Muster	michael@muster.de		Handball	200	1	<input type="checkbox"/> Mo <input type="checkbox"/> Di <input type="checkbox"/> Mi <input type="checkbox"/> Do <input type="checkbox"/> Fr <input type="checkbox"/> Sa <input type="checkbox"/> So	<input type="button" value="Löschen"/> <input type="button" value="Passwort zurücksetzen"/>

Abbildung 6.11.: Verwaltung der Assistentz-Team-Mitglieder

Sollte ein Assistent sein Passwort vergessen haben, hat der Klient mit Hilfe des Buttons „Paswort zurücksetzen“ (in der Spalte „Aktionen“) die Möglichkeit das Passwort des Assistenten auf das Standard-Passwort (was in den Einstellungen hinterlegt ist) zu setzen.

Sollte ein Assistent das Team verlassen, so hat der Klient die Möglichkeit mit Hilfe des Buttons „Löschen“ (in der Spalte „Aktionen“) den Assistenten aus der Team-Verwaltung zu entfernen.

Implementierungsdetails

Die Team-Verwaltungs-Seite wird von dem PHP-Script `teamTable.php` erstellt. Drückt der Benutzer auf den Button „Team Speichern“, so wird die Funktion `saveTeam` im JavaScript `team.js` aufgerufen. Diese Funktion ruft die Unterfunktion `checkLoginNames`¹⁹ auf, parst aus der Team-Tabelle die Daten heraus und schickt sie per POST-Parameter an das PHP-Script `teamSaver.php`, welches die Daten persistent in der Datei `team.txt` im Ordner `Team` speichert.

Drückt der Benutzer auf den Button „Neues Mitglied“, so wird die Funktion `newMember` im JavaScript `team.js` aufgerufen. In dieser wird per DOM eine neue Tabellenzeile erstellt und an die bestehende Tabelle angehängt.

Drückt der Benutzer auf den Button „Passwort zurücksetzen“ (in der Spalte Aktionen), so wird die Funktion `resetPassword` im JavaScript `team.js` aufgerufen. Die ID des betroffenen Team-Mitgliedes wird per POST-Parameter an das PHP-Script `resetPassword.php` geschickt. Dieses Script erzeugt eine Instanz der Klasse `Passwords` und ruft die Funktion `resetPassword` mit der übergebenen ID auf. Diese Funktion setzt das Passwort auf das Standardpasswort, welches in der Klasse `Settings` hinterlegt ist.

Drückt der Benutzer auf den Button „Löschen“ (in der Spalte Aktionen), so wird die Funktion `removeMember` im JavaScript `team.js` aufgerufen. Der Benutzer wird gefragt, ob er das wirklich tun möchte. Stimmt er zu, so wird per DOM die Zeile aus der Tabelle entfernt.

¹⁹ `checkLoginNames` prüft unter anderem, dass alle Team-Mitglied-IDs voneinander verschieden sind. Falls etwas nicht den Anforderungen genügt, wird der Benutzer mit einem Info-Fenster benachrichtigt und das Speichern der Team-Tabelle wird abgebrochen.



6. Implementierung

6.13. Monatsplan

Mit dem Monatsplan (siehe Abbildung 6.12) plant der Klient den nächsten Monat. Er kann pro Tag den Dienstbeginn und das Dienstende festlegen und Notizen zum Tag erstellen. Die Notizen unterteilen sich in öffentliche und private Notizen. Öffentliche Notizen können auch vom Assistenzteam eingesehen werden. Private Notizen können nur vom Klienten selbst eingesehen werden.

Monatsplan für September 2014

Datum	Dienstbeginn	Dienstende	Bemerkungen (öffentlich)	Bemerkungen (privat)
Mo, 01.09.	13:00 ↓	13:00 ↓	Urlaub	
Di, 02.09.	13:00 ↓	13:00 ↓	Urlaub	
Mi, 03.09.	13:00 ↓	13:00 ↓	Urlaub	

Abbildung 6.12.: Monatsplan mit der Möglichkeit tagesweise Dienstbeginn und Dienstende festzulegen und Bemerkungen zu hinterlegen

Neben den tagesweisen Bemerkungen kann der Klient eine Nachricht an sein Team verfassen. Hier können allgemeine Hinweise für den Monat untergebracht werden.

Der Klient hat jederzeit die Möglichkeit den Monatsplan zu speichern. Möchte er sein Assistenz-Team auffordern, Termine einzutragen, so kann er auf den Button „Team benachrichtigen“ klicken. Vom Assistenzplaner wird dann eine E-Mail an jedes Mitglied des Assistenz-Teams geschickt mit der Bitte, mögliche Termine einzutragen. Bei Bedarf kann der Klient eine zusätzliche Nachricht an sein Team verfassen. Eine beispielhafte Nachricht findet sich in Abbildung 6.13.

Assistenzplaner - Bitte mögliche Termine eintragen

Liebes Team,

bitte tragt bis **15. August 2014** Eure möglichen Termine für den September 2014 im [Assistenzplaner](#) ein.

Vielen Dank!

Hier noch eine Nachricht von Mustermann:

Liebes Team,

bitte beachtet die veränderten Dienstzeiten aufgrund meines 2-wöchigen Urlaubs.

Viele Grüße,
Max Mustermann

Abbildung 6.13.: Beispielhafte Benachrichtigung der Assistenten mit der Bitte des Klienten, die Termine für den nächsten Monat einzutragen



6. Implementierung

Implementierungsdetails

Der Monatsplan wird vom PHP-Script `monthPlanView.php` erstellt. Darin wird eine Instanz der Klasse `MonthPlan` erzeugt. Drückt der Benutzer auf den Button „Speichern“, so wird die Funktion `save` im JavaScript `monthPlan.js` aufgerufen. Diese Funktion parst aus dem Monatsplan die Daten heraus, ergänzt sie um die Eingaben im Nachrichtenfeld und schickt sie per POST-Parameter an das PHP-Script `monthPlanSaver.php`, welches die Daten persistent in der Datei `YYYY-M.txt`²⁰ im Ordner `MonthPlan` speichert.

Drückt der Benutzer auf den Button „Team benachrichtigen“, so wird die Funktion `notifyTeam` im JavaScript `monthPlan.js` aufgerufen. Per POST-Parameter wird die verfasste Nachricht, das Jahr und der Monat an das PHP-Script `notifyTeam.php` geschickt. Dieses Script erstellt im Wesentlichen eine Instanz von `PHPMailer`²¹ und verschickt eine E-Mail an das Team.

6.14. Kalender

Mit Hilfe des Kalenders (siehe Abbildung 6.14) kann ein Assistent die Termine eintragen, an denen er für den Dienst verfügbar ist. Durch einen Klick auf ein Datum ändert sich der Zustand des Datums und damit auch die Hintergrundfarbe. Rot bedeutet, der Assistent ist an diesem Datum nicht verfügbar. Gelb bedeutet, dass der Assistent zur Not verfügbar wäre. Grün bedeutet, dass der Assistent an dem Datum uneingeschränkt zur Verfügung steht.

Eingabe der Daten

Mo	Di	Mi	Do	Fr	Sa	So
1 17:00 - 08:00	2 13:00 - 08:00	3 17:00 - 08:00	4 13:00 - 08:00	5 14:00 - 13:00	6 13:00 - 13:00	7 13:00 - 08:00
8 17:00 - 08:00	9 13:00 - 08:00	10 17:00 - 08:00	11 13:00 - 08:00	12 14:00 - 13:00	13 13:00 - 13:00	14 13:00 - 08:00
15 17:00 - 08:00	16 13:00 - 08:00	17 17:00 - 08:00	18 13:00 - 08:00	19 14:00 - 13:00	20 13:00 - 13:00	21 13:00 - 08:00
22 17:00 - 08:00	23 13:00 - 08:00	24 17:00 - 08:00	25 13:00 - 08:00	26 14:00 - 13:00	27 13:00 - 13:00	28 13:00 - 08:00
29 17:00 - 08:00	30 13:00 - 08:00					

Abbildung 6.14.: Kalender für die Eingabe der möglichen Termine eines Assistenten

²⁰YYYY steht für das Jahr, M für den Monat - Beispiel 2014-10.txt

²¹<https://github.com/PHPMailer/PHPMailer> - zuletzt abgerufen am 12.09.2014



6. Implementierung

Mit Hilfe des Buttons „Alle Daten markieren“ kann man alle Daten eines Monats grün (bedeutet: Assistent verfügbar) markieren.

Im oberen Bereich der Webseite sind alle Informationen dargestellt (siehe Abbildung 6.15), die dem Assistenten helfen können. Das sind zum einen die öffentlichen Notizen, die der Klient zum Monat erstellt hat, zum anderen die optionale Nachricht, die der Klient an das Team verfasst hat.

Allgemeine Bemerkungen von Mustermann

Liebes Team,

bitte beachtet meinen Urlaub am Anfang des Monats, bei dem ich 24 stündige Assistenzen benötige.

Vielen Dank!

Bemerkungen zu den Terminen

Datum	Bemerkung
Mo, 01.09.	Urlaub
Di, 02.09.	Urlaub

Abbildung 6.15.: Bemerkungen vom Klienten für seine Assistenten

Implementierungsdetails

Die Kalender-Seite wird von dem PHP-Script `calendarView.php` erstellt. Drückt der Benutzer auf den Button „Speichern“, so wird die Funktion `save` im JavaScript `calendar.js` aufgerufen. Diese Funktion parst aus dem Kalender die Daten heraus und schickt sie per POST-Parameter an das PHP-Script `calendarSaver.php`, welches die Daten persistent in der Datei `YYYY-M.txt` im Ordner `AssistanceInput` speichert.

6.15. Dienstplan

Aus Sicht des Klienten

Wenn die Assistenten ihre Termine eingetragen haben, ist der Klient in der Lage den Dienstplan zu erstellen. Falls es Tage gibt, an denen nicht mindestens zwei Assistenten (einer für Dienst und einer für Bereitschaft) Zeit haben, so bekommt er eine Warnung, da dann eine vollständige Dienstplanerstellung nicht möglich ist.



6. Implementierung

Der Assistenzplaner erstellt eine Tabelle (siehe Abbildung 6.16), in der tageweise die Dienstzeiten, die Verfügbarkeiten der Assistenten und die Bemerkungen (öffentliche wie private) aufgeführt werden. Die Verfügbarkeiten der Assistenten werden farblich markiert.

- Grün: Assistent ist voll verfügbar
- Gelb: Assistent ist zur Not verfügbar
- Rot: Assistent ist nicht verfügbar

Dienstplan für September 2014

Letzte Änderung: 08.08.2014 08:13

Datum	Zeit	Michael	Matthias	Markus	Bemerkungen (öffentlich)	Bemerkungen (privat)
Mo, 01.09.	13:00 - 13:00		Dienst	Bereitschaft	Urlaub	
Di, 02.09.	13:00 - 13:00	Bereitschaft	Dienst		Urlaub	
Mi, 03.09.	13:00 - 13:00	Dienst		Bereitschaft	Urlaub	

Abbildung 6.16.: Editierbarer Dienstplan für den Klienten

Der Klient bekommt vom Assistenzplaner mit Hilfe des Algorithmus (siehe Abschnitt 6.16) einen Dienstplan erstellt, den er nach Belieben anpassen kann. Dazu klickt er in die Zellen der Dienstplantabelle und kann dadurch zwischen den Zuständen

- Assistent hat Dienst
- Assistent hat Bereitschaft
- Assistent hat weder Dienst noch Bereitschaft

wechseln.

Unterhalb der Dienstplantabelle zeigt eine weitere Tabelle (siehe Abbildung 6.17) die Stundenverteilung der Assistenten.

Stundenübersicht

Person	Stunden	Benötigte Stunden	Differenz in Stunden	Differenz in Prozent
Michael	209	200	9	5 %
Matthias	125	120	5	4 %
Markus	105	100	5	5 %

Abbildung 6.17.: Stundenübersicht der Assistenten

In der zweiten Spalte sind die zusammengerechneten Arbeitsstunden (Dienststunden plus Bereitschaftsstunden) aufgeführt. In der dritten Spalte sieht man die benötigten Arbeitsstunden. In der vierten Spalte wird die Differenz zwischen Spalte 2 und



6. Implementierung

3 berechnet. In der fünften Spalte ist die Differenz (Spalte 4) ins Verhältnis der benötigten Arbeitsstunden (Spalte 3) gesetzt und prozentual ausgedrückt. Die Zeilen werden farblich hinterlegt:

- Grün: Der Betrag der prozentualen Abweichung (Spalte 5) ist kleiner 10%
- Gelb: Der Betrag der prozentualen Abweichung (Spalte 5) ist zwischen 10% und 20%
- Rot: Der Betrag der prozentualen Abweichung (Spalte 5) ist größer als 20%

Die Stundenübersicht wird jedes Mal aktualisiert, wenn der Klient in der Dienstplattabelle etwas ändert. Dadurch hat der Klient ein direktes Feedback, wie sich eine Dienstplanänderung auf die Stundenausschöpfung der Assistenten auswirkt.

Per Knopfdruck kann der Klient einige Funktionen ausführen:

- Dienstplan prüfen: Es wird geprüft, dass für jeden Tag genau ein Assistent Dienst und genau ein Assistent Bereitschaft hat.
- Dienstplan speichern: Der Dienstplan wird gespeichert. Gleichzeitig wird das Datum der letzten Änderung aktualisiert.
- Dienstplan löschen: Der Dienstplan wird nach Rückversicherung gelöscht. Durch ein Neuladen der Dienstplan-Seite wird vom Assistanzplaner ein neuer Dienstplanvorschlag erstellt.
- Dienstplan als PDF anzeigen: Der Dienstplan wird als *Portable Document Format (PDF)* exportiert. Als PDF kann der Dienstplan leicht per E-Mail verschickt und ausgedruckt werden.

Aus Sicht des Assistenten

Der Assistent hat keine Möglichkeit den Dienstplan zu verändern. Er kann lediglich den fertigen Dienstplan einsehen. In einer Tabelle (siehe Abbildung 6.18) werden tagesweise die Dienst- und Bereitschaftszeiten dargestellt. Weiterhin kann man sehen welcher Assistent Dienst bzw. Bereitschaft hat. Die Dienste und Bereitschaften des angemeldeten Assistenten werden farblich hervorgehoben.

Unter dem allgemeinen Dienstplan gibt es eine Tabelle (siehe Abbildung 6.19), in der nur die Dienste und Bereitschaften des angemeldeten Assistenten angezeigt werden.



6. Implementierung

Dienstplan für September 2014

Letzte Änderung: 08.08.2014 08:13

Datum	Dienst-Zeit	Dienst	Bereitschafts-Zeit	Bereitschaft	Bemerkungen
Mo, 01.09.	13:00 - 13:00	Matthias	10:00 - 11:00 und 18:00 - 19:00	Markus	Urlaub
Di, 02.09.	13:00 - 13:00	Matthias	10:00 - 11:00 und 18:00 - 19:00	Michael	Urlaub
Mi, 03.09.	13:00 - 13:00	Michael	10:00 - 11:00 und 18:00 - 19:00	Markus	Urlaub

Abbildung 6.18.: Gesamter Dienstplan aus Sicht des Assistenten „Markus“ - farbig hervorgehoben sind die Dienste bzw. Bereitschaften des angemeldeten Assistenten

Meine Dienste und Bereitschaften

Datum	Zeit	Typ
Mo, 01.09.	10:00 - 11:00 und 18:00 - 19:00	Bereitschaft
Mi, 03.09.	10:00 - 11:00 und 18:00 - 19:00	Bereitschaft

Abbildung 6.19.: Dienste und Bereitschaften des angemeldeten Assistenten „Markus“

Implementierungsdetails

Die Dienstplan-Seite wird von dem PHP-Script `rosterView.php` erstellt. Drückt der Benutzer auf den Button „Dienstplan speichern“, so wird die Funktion `save` im JavaScript `roster.js` aufgerufen. Diese Funktion prüft die Vollständigkeit des Dienstplans, parst aus dem Dienstplan die Daten heraus und schickt sie per POST-Parameter an das PHP-Script `rosterSaver.php`, welches die Daten persistent in der Datei `YYYY-M.txt` im Ordner `Roster` speichert.

Drückt der Benutzer auf den Button „Dienstplan prüfen“, so wird die Funktion `checkRoster` im JavaScript `roster.js` aufgerufen. Diese Funktion prüft für jeden Tag, dass genau ein Assistent für den Dienst und genau ein Assistent für die Bereitschaft eingetragen ist. Ist dies nicht der Fall, so wird der Benutzer mit einer Message-Box darüber benachrichtigt.

Drückt der Benutzer auf den Button „Dienstplan löschen“, so wird die Funktion `deleteRoster` im JavaScript `roster.js` aufgerufen. Der Benutzer wird gefragt, ob er dies wirklich tun möchte. Nach der Rückversicherung wird das PHP-Script `rosterEraser.php` aufgerufen.

Drückt der Benutzer auf den Button „Dienstplan als PDF anzeigen“, so wird die Funktion `createPdf` im JavaScript `roster.js` aufgerufen. Diese Funktion prüft, ob der Dienstplan vollständig ist und navigiert zum PHP-Script `rosterViewPdf.php`,



6. Implementierung

das die PDF-Version des Dienstplans anzeigt. Hierbei kommt die Klasse **Fpdf**²² zum Einsatz.

6.16. Entwicklung des Algorithmus zur Dienstplanerstellung

Im folgenden Abschnitt wird erläutert, was die Anforderungen an den Algorithmus zur Dienstplanerstellung sind und wie der Algorithmus entwickelt und iterativ verfeinert worden ist.

Anforderungen

Mindestanforderung an den Algorithmus ist, dass er einen Dienstplan erstellt, der eine hundertprozentige Betreuungsabdeckung erreicht. Für jeden Tag muss ein Assistent für den Dienst und ein Assistent für die Bereitschaft im Dienstplan eingetragen sein.

Nebenbedingungen sind, dass

- die Vorlieben des Klienten berücksichtigt werden (zum Beispiel Assistent A lieber am Wochenende, Assistent B bei einem wichtigen Termin, Assistent C bei Fußballspielen)
- die Stundenkontingente der Assistenten gleichmäßig ausgeschöpft werden (vergleiche Abschnitt 3.1)

Eingabe

Der Algorithmus bekommt als Eingabe die Verfügbarkeiten der Assistenten (pro Tag). Diese sind folgendermaßen codiert:

- 0: Assistent ist nicht verfügbar
- 1: Assistent ist zur Not verfügbar
- 10: Assistent ist voll verfügbar

Weiterhin hat der Algorithmus Zugriff auf alle Daten, die in der Team-Verwaltung (siehe Abschnitt 6.12) hinterlegt sind. Welche Version des Algorithmus auf welche Daten zugreift lässt sich Tabelle 6.1 entnehmen.

²²<http://www.fpdf.org> - zuletzt abgerufen am 13.09.2014



6. Implementierung

Version	1	2	3	4	5
Priorisierung	X	✓	✓	✓	✓
Bevorzugte Wochentage	X	X	✓	✓	✓
Stichwörter	X	X	X	✓	✓
Stundenkontingent	X	X	X	X	✓

Tabelle 6.1.: Übersicht der verwendeten Daten aus der Team-Verwaltung

Ausgabe

Der Algorithmus gibt für jeden Tag des Monats einen Assistenten für den Dienst und einen Assistenten für die Bereitschaft aus.

Erste Version - Hundertprozentige Betreuungsabdeckung

Die erste Version des Algorithmus berücksichtigt nur die Mindestanforderung (hundertprozentige Betreuungsabdeckung). Der Algorithmus iteriert über alle Tage des Monats und prüft, ob es mindestens zwei Assistenten gibt, die Zeit haben. Der erste freie Assistent bekommt den Dienst, der zweite freie Assistent bekommt die Bereitschaft zugewiesen. Bei dieser ersten Version entsteht ein vollständiger Dienstplan. Der Klient ist den ganzen Monat betreut, jedoch wurden weder die Stundenkontingente noch die Vorlieben des Klienten berücksichtigt.

Zweite Version - Berücksichtigung der Priorisierungswerte

In der zweiten Version des Algorithmus werden die allgemeinen Vorlieben des Klienten bezüglich der Assistenten berücksichtigt. Für jedes Team-Mitglied kann der Klient einen Priorisierungswert festlegen (siehe Abschnitt 6.12). Es wird eine Punktetabelle erstellt. Die erste Spalte ist der Tag des Monats, jede weitere Spalte repräsentiert die Verfügbarkeit eines Assistenten. Ist der Assistent nicht verfügbar, so wird eine Null eingetragen. Ist der Assistent zur Not verfügbar, so wird eine Eins eingetragen. Ist der Assistent voll verfügbar, so wird eine Zehn eingetragen. Als Beispiel dienen die Zahlen in Tabelle 6.2.

Tag des Monats	Punkte Assistent 1	Punkte Assistent 2	Punkte Assistent 3
1 (Montag)	0	10	10
2 (Dienstag)	1	10	0
3 (Mittwoch)	10	1	10
...

Tabelle 6.2.: Punktetabelle zur Dienstplanerstellung



6. Implementierung

Nun wird jeder Verfügbarkeitswert mit dem Priorisierungswert multipliziert. Mit den beispielhaften Priorisierungswerten aus Tabelle 6.3 verändert sich die Punktetabelle.

Assistent	Priorisierungswert
Assistent 1	1
Assistent 2	2
Assistent 3	3

Tabelle 6.3.: Priorisierungswerte aus der Team-Verwaltung

Das Ergebnis ist in Tabelle 6.4 zu sehen.

Tag des Monats	Punkte Assistent 1	Punkte Assistent 2	Punkte Assistent 3
1 (Montag)	0	20	30
2 (Dienstag)	1	20	0
3 (Mittwoch)	10	2	30
...

Tabelle 6.4.: Punktetabelle nach Multiplikation mit Priorisierungswerten

Nach der Erstellung der Punktetabelle iteriert der Algorithmus über alle Tage und gibt dem Assistenten mit der höchsten Punktzahl den Dienst und dem Assistenten mit der zweithöchsten Punktzahl die Bereitschaft. Mit den Beispielwerten würde sich ein Dienstplan wie er in Tabelle 6.5 dargestellt ist ergeben.

Tag des Monats	Dienst	Bereitschaft
1 (Montag)	Assistent 3	Assistent 2
2 (Dienstag)	Assistent 2	Assistent 1
3 (Mittwoch)	Assistent 3	Assistent 1
...



Tabelle 6.5.: Beispielhafter Dienstplan

Dritte Version - Berücksichtigung der Wochentage

In der dritten Version des Algorithmus werden (zusätzlich zu den personenbezogenen Priorisierungswerten) die Wochentage (Montag, Dienstag, ...) berücksichtigt. Der Klient kann Assistenten an bestimmten Wochentagen bevorzugen (vergleiche Abschnitt 6.12). Der Algorithmus geht genauso vor wie im vorherigen Abschnitt beschrieben. Zusätzlich wird jedoch der Wochentag berücksichtigt. Ist zum Beispiel Assistent 2 am Montag priorisiert, so werden die Werte an allen Montagen in der Punktetabelle verdoppelt (siehe Tabelle 6.6). Somit steigt die Wahrscheinlichkeit, dass Assistent 2 an einem Montag für den Dienst eingeteilt wird. In dem konkreten



6. Implementierung

Beispiel wird Assistent 2 am Montag den Dienst bekommen, obwohl Assistent 3 eine generell höhere Priorisierung hat.

Tag des Monats	Punkte Assistent 1	Punkte Assistent 2	Punkte Assistent 3
1 (Montag)	0	40	30
2 (Dienstag)	1	20	0
3 (Mittwoch)	10	2	30
...

Tabelle 6.6.: Punktetabelle nach Berücksichtigung der Wochentage

Vierte Version - Berücksichtigung der Stichwörter

Der Algorithmus arbeitet prinzipiell wie im vorherigen Abschnitt beschrieben. Bei der Erstellung der Punktetabelle werden jedoch die privaten Bemerkungen im Monatsplan tagesweise nach Stichwörtern durchsucht. Ist ein Stichwort in der Team-Verwaltung bei einem Assistenten hinterlegt, so wird der Punktwert des Assistenten an dem betreffenden Tag verzehnfacht.

Zur Erläuterung ein Beispiel: Angenommen beim Assistenten 3 ist in der Team-Verwaltung das Stichwort „FCB“ hinterlegt. Wenn gleichzeitig bei den privaten Bemerkungen am ersten Tag des Monats ein „FCB“ enthalten ist, so wird der Punktwert des Assistenten von 30 auf 300 verzehnfacht (vergleiche auch Tabelle 6.7). Damit hat Assistent 3 die höchste Punktzahl für den ersten Tag des Monats und wird für den Dienst eingeteilt.

Tag des Monats	Private Bemerkungen	Assistent 1	Assistent 2	Assistent 3
1 (Montag)	FCB	0	40	300
2 (Dienstag)		1	20	0
3 (Mittwoch)		10	2	30
...

Tabelle 6.7.: Punktetabelle nach Berücksichtigung der Stichwörter

Fünfte Version - Berücksichtigung der Stundenkontingente

In der letzten Verfeinerung des Algorithmus werden die Stundenkontingente berücksichtigt. Bisher hat der Algorithmus über alle Tage iteriert und jeden Tag isoliert betrachtet. Für jeden Tag wurde ein lokales Optimum gefunden. In der Regel führt dieses Vorgehen allerdings zu einer sehr unausgewogenen Stundenverteilung zwischen den Assistenten. Um die Stundenkontingente der Assistenten gleichmäßig auszuschöpfen ist es notwendig den gesamten Monat auf einmal zu betrachten.



6. Implementierung

Dazu werden die Daten in ein anderes Tabellenformat (siehe Tabelle 6.8) konvertiert. In die erste Spalte werden die Punkte (wenn sie größer Null sind) eingetragen, in der zweiten Spalte steht der Assistent und in der dritten Spalte der Tag des Monats.

Punkte	Name	Tag
1	Assistent 1	2
10	Assistent 1	3
40	Assistent 2	1
20	Assistent 2	2
2	Assistent 2	3
300	Assistent 3	1
30	Assistent 3	3

Tabelle 6.8.: Umgewandelte Punktetabelle

Die Punktetabelle wird anschließend randomisiert (durcheinandergewürfelt, gemischt), damit es nicht zur Blockbildung bei den Diensten (ein Assistent hat mehrere Tage hintereinander Dienst) kommt. Anschließend wird sie nach der ersten Spalte (dem Punktwert) sortiert (siehe Tabelle 6.9).

Punkte	Name	Tag
300	Assistent 3	1
40	Assistent 2	1
30	Assistent 3	3
20	Assistent 2	2
10	Assistent 1	3
2	Assistent 2	3
1	Assistent 1	2

Tabelle 6.9.: Umgewandelte Punktetabelle, randomisiert und sortiert nach Punkten

Weiterhin werden alle Stundenkontingente der Assistenten, die in der Team-Verwaltung (siehe Abschnitt 6.12) hinterlegt sind, addiert.

Dann werden alle Dienst- und Bereitschaftsstunden addiert. Ist die Summe der Dienst- und Bereitschaftsstunden höher als die Summe der Stundenkontingente, so wird ein Skalierungsfaktor (Summe der Dienst- und Bereitschaftsstunden geteilt durch die Summe der Stundenkontingente) ermittelt und die Stundenkontingente mit dem Skalierungsfaktor multipliziert, sprich die Stundenkontingente aller Assistenten werden um einen Prozentsatz x erhöht. Ist die Summe der Dienst- und Bereitschaftsstunden kleiner oder gleich als die Summe der Stundenkontingente, bleiben die Stundenkontingente der Assistenten unverändert.

Nach den Vorbereitungsarbeiten wird der Kern des Algorithmus gestartet. Zuerst werden alle Dienste vergeben, anschließend die Bereitschaften. Der Algorithmus ite-



6. Implementierung

riert über die nach Punkten sortierte Tabelle 6.9. In jeder Zeile der Tabelle prüft der Algorithmus ob der Assistent noch genug Stunden zur Verfügung hat und ob der Dienst an dem entsprechenden Tag noch frei ist. Falls beide Bedingungen erfüllt sind, bekommt der Assistent den Dienst eingetragen und seine verfügbaren Stunden werden um die Netto-Dienststundenanzahl des Tages verringert.

Durch die unterschiedlich langen Dienstzeiten von Tag zu Tag und unterschiedlichen Stundenkontingente wird nach dem ersten Durchgang nicht für jeden Tag ein Assistent für den Dienst bestimmt werden können.

Folgendes Beispiel soll dies erläutern. Angenommen der Monat hätte nur drei Tage. Die Dienstzeiten betragen 18, 16 und 14 Stunden. Die Summe der Dienstszeiten wäre 48. Es gibt zwei Assistenten (A1 und A2) mit Stundenkontingenten von 28 und 20 Stunden. Beide Assistenten haben an allen drei Tagen Zeit und haben gleiche Priorisierungswerte. Der Skalierfaktor wäre genau 1 und die Stundenkontingente der Assistenten würden nicht angepasst werden. Nach dem ersten Durchlauf (bei gleichen Priorisierungswerten) hätte der erste Assistent (A1) den 18 Stunden-Dienst und der zweite Assistent (A2) den 16 Stunden-Dienst. Der 14 Stunden-Dienst würde bei keinem der beiden Assistenten mehr ins Stundenkontingent passen (siehe Abbildung 6.20).

A1	18 Stunden Dienst an Tag 1	10 Stunden frei verfügbar
A2	16 Stunden Dienst an Tag 2	4 Stunden

Abbildung 6.20.: Verteilung der Dienste mit 0 Stunden Toleranz

Um die angesprochene Problematik zu lösen wird um den Kern des Algorithmus eine Schleife hinzugefügt, die einen Toleranzwert (beginnend bei Null Stunden) immer um eine volle Stunde inkrementell erhöht. Bei der Prüfung, ob der Assistent noch genug Stunden zur Verfügung hat, wird der Toleranzwert zum noch verbleibenden Stundenkontingent hinzugefügt. Die hinzugefügte Schleife endet, sobald für jeden Tag ein Dienst vergeben wurde.

Angewendet auf das oben genannte Beispiel würde der die Schleife im fünften Durchlauf (Toleranz = 4 Stunden - siehe Abbildung 6.21) dem ersten Assistenten (A1) den Dienst geben, da die verbleibenden 10 Stunden (28 Stunden aus dem Stundenkontingent abzüglich der 18 Stunden des ersten Dienstes) seines Kontingentes mit der Toleranz von 4 Stunden addiert genau die 14 Stunden Dienstzeit ergibt (siehe Abbildung 6.22).

A1	18 Stunden Dienst an Tag 1	10 Stunden frei verfügbar	4 h Toleranz
A2	16 Stunden Dienst an Tag 2	4 Stunden	4 h Toleranz

Abbildung 6.21.: Stundenübersicht mit 4 Stunden Toleranz



6. Implementierung

A1	18 Stunden Dienst an Tag 1		14 Stunden Dienst an Tag 3
A2	16 Stunden Dienst an Tag 2	4 Stunden	4 h Toleranz

Abbildung 6.22.: Verteilung der Dienste mit 4 Stunden Toleranz

Nachdem alle Dienste vergeben wurden, werden auf dieselbe Weise die Bereitschaften vergeben. Einziger Unterschied ist, dass der Toleranzwert immer um eine halbe Stunde inkrementiert wird.

Durch das Randomisieren der Liste (siehe Tabelle 6.9) können mehrere Dienstpläne erstellt werden. Um von den vielen möglichen Dienstplänen einen zu wählen, der die Stundenkontingente möglichst gleichmäßig ausschöpft, werden 1000 Dienstpläne erstellt und am Ende ermittelt, wie gut jeder Dienstplan ist. Als Metrik dient die Summe der beiden Toleranzwerte (Dienst und Bereitschaft). Das theoretische Optimum ist Null. Der Dienstplan mit der niedrigsten Summe der Toleranzwerte wird dem Benutzer angezeigt.

Implementierungsdetails

Der Quellcode der kommentierten Implementierung der fünften Version des Algorithmus findet sich in Listing 6.2.

Listing 6.2: Algorithmus zur Dienstplanerstellung

```

1 <?php
2
3 // Die Funktion createRosterAlgorithm5() ist eine private Funktion der
4 // Klasse Roster ($this) und hat damit Zugriff auf die Kalender-Eingaben
5 // der Assistenten und die Daten aus der Team-Verwaltung
6
7 function createRosterAlgorithm5()
8 {
9     if (!$this->assistanceInput->dataExist) {
10         // wenn die Assistenten keine Termine eingetragen haben, kann kein
11         // Dienstplan erstellt werden
12         return;
13     }
14
15     for ($i = 0; $i < $this->daysPerMonth; $i++) {
16         // fuer jeden Tag des Monats wird verifziert, dass mindestens zwei
17         // Assistenten verfuegbar sind
18         $countOfAvailableAssistants = 0;
19         foreach ($this->assistanceInput->assistanceInput as $name => $dates) {
20             if ($dates[$i] > 0) {
21                 $countOfAvailableAssistants++;
22             }
23         }
24         if ($countOfAvailableAssistants <= 1) {
25             continue;
26         }
27         // hier wird der Dienstplan erstellt
28     }
29 }
```



6. Implementierung

```

18     }
19 }
20 if ($countOfAvailableAssistants < 2) {
21     return;
22 }
23 }

// Die Gesamtstunden aller Assistenten (die moegliche Daten eingegeben
// haben) werden zusammengerechnet
26 $totalQuotaOfHours = 0;
27 $quotaOfHours = $this->team->getHours();
28 foreach ($quotaOfHours as $name => $value) {
29     if (array_key_exists($name, $this->assistanceInput->assistanceInput)) {
30         $totalQuotaOfHours += $value;
31     }
32 }

// Die Gesamtstunden (Dienstzeiten + Bereitschaftszeiten) werden berechnet
35 $totalOfServiceHours = 0;
36 $totalOfStandbyHours = 0;
37 for ($i = 1; $i <= $this->daysPerMonth; $i++) {
38     $totalOfServiceHours += $this->monthPlan->days[$i]->serviceHours;
39     $totalOfStandbyHours += $this->monthPlan->days[$i]->standbyHours;
40 }

// Der Skalierfaktor wird ermittelt
43 $scaleFactor = ($totalOfServiceHours + $totalOfStandbyHours) /
    $totalQuotaOfHours;

44 if ($scaleFactor < 1) {
45     // Wenn der Skalierfaktor kleiner 1 ist, so wird er auf 1 korrigiert
46     // Damit wird den Vorlieben des Klienten Vorrang vor einer
        // gleichmaessigen Stunden-Kontingent-Ausschoepfung gewaehrt
47     $scaleFactor = 1;
48 }
49

// Vorbereitung fuer die Berechnung der Punktetabelle
51 $priorities = $this->team->getPriorities();
52 $scoreTable = $this->assistanceInput->assistanceInput;
53 $preferredWeekdays = $this->team->getPreferredWeekdays();
54 $keyWords = $this->team->getKeyWords();

56
57 foreach ($this->assistanceInput->assistanceInput as $name => $dates) {
58     for ($i = 1; $i <= $this->daysPerMonth; $i++) {
59

```



6. Implementierung

```

60     // allgemeine Vorlieben des Klienten werden beruecksichtigt
61     $scoreTable[$name][$i - 1] *= $priorities[$name];
62
63     // bevorzugte Wochentage werden beruecksichtigt
64     if ($preferredWeekdays[$name][$this->monthPlan->days[$i]->weekday - 1]
65         == 1) {
66         $scoreTable[$name][$i - 1] *= 2;
67     }
68
69     // Stichwoerter werden beruecksichtigt
70     foreach ($keyWords[$name] as $keyWord) {
71         if ($keyWord != "") {
72             if (strpos(strtolower($this->monthPlan->days[$i]->privateNotes),
73                 strtolower($keyWord)) !== false) {
74                 // wenn ein Assistenten-Stichwort in den privaten Bemerkungen des
75                 // Tages enthalten ist, so wird der Prioritaetswert mit dem
76                 // Faktor 10 multipliziert
77                 $scoreTable[$name][$i - 1] *= 10;
78             }
79         }
80     }
81
82     // konvertiere die Punkte-Tabelle, so dass eine Betrachtung des ganzen
83     // Monats moeglich ist
84     $convertedData = array();
85     foreach ($this->assistanceInput->assistanceInput as $name => $dates) {
86         for ($i = 1; $i <= $this->daysPerMonth; $i++) {
87             if ($scoreTable[$name][$i - 1] == 0) {
88                 continue;
89             }
90             $entry = array();
91             array_push($entry, $scoreTable[$name][$i - 1]);
92             array_push($entry, $name);
93             array_push($entry, $i);
94
95             array_push($convertedData, $entry);
96         }
97     }
98
99     $countOfRuns = 1000;
100    $smallestDifference = PHP_INT_MAX;
101    $servicePersonsBest = array();
102    $standbyPersonsBest = array();

```



6. Implementierung

```

100
101 // Erstelle 1000 Dienstplaene und nimm am Ende den besten
102 for ($run = 0; $run < $countOfRuns; $run++) {
103
104     shuffle($convertedData); // randomisieren, damit es keine Block-Bildung
105         bei den Diensten gibt
106     usort($convertedData, 'compare'); // nach Punkten sortieren - die
107         compare-Funktion ist in diesem Listing nicht dargestellt
108
109     // Dienstplan (vom vorherigen Durchlauf) loeschen
110     for ($i = 1; $i <= $this->daysPerMonth; $i++) {
111         $this->servicePerson[$i] = "";
112         $this->standbyPerson[$i] = "";
113     }
114
115     // skalierte Stundenkontigente der Assistenten
116     $quotaOfHours = $this->team->getHours($scaleFactor);
117
118     // Bestimmung der Assistenten fuer die Dienste
119     $serviceTolerance = 1;
120     $serviceRun = 0;
121     while (!$this->isServiceComplete()) { // Laufe solange, bis alle Dienste
122         eingeteilt sind
123         for ($i = 0; $i < count($convertedData); $i++) { // Laufe ueber alle
124             Elemente der konvertierten Tabelle
125             if ($this->servicePerson[$convertedData[$i][2]] == "") { // Pruefe,
126                 ob Dienst noch frei
127                 if ($quotaOfHours[$convertedData[$i][1]] - $this->monthPlan->days[
128                     $convertedData[$i][2]]->serviceHours > 0 - ($serviceTolerance *
129                     $serviceRun)) { // Pruefe, ob Dienst noch in das
130                     Stundenkontigennt (+ Toleranz) reinpasst
131                     $this->servicePerson[$convertedData[$i][2]] = $convertedData[$i
132                         ][1]; // Weise Dienst zu
133                     $quotaOfHours[$convertedData[$i][1]] -= $this->monthPlan->days[
134                         $convertedData[$i][2]]->serviceHours; // Ziehe Stunden des
135                         zugewiesenen Dienstes von Stundenkontingenetn ab
136                 }
137             }
138         }
139         // inkrementiere den Durchlauf-Zaehler, damit die Stunden Toleranz
140             beim naechsten Durchlauf eine Stunde groesser ist
141         $serviceRun++;
142     }
143
144     // Bestimmung der Assistenten fuer die Bereitschaften

```



6. Implementierung

```

133 // Vorgehensweise analog zur Bestimmung der Assistenten fuer die Dienste
134 $standbyTolerance = 0.5;
135 $standbyRun = 0;
136 while (!$this->isStandbyComplete()) {
137     for ($i = 0; $i < count($convertedData); $i++) {
138         if ($this->standbyPerson[$convertedData[$i][2]] == "" && $this->
139             servicePerson[$convertedData[$i][2]] != $convertedData[$i][1]) {
140             if ($quotaOfHours[$convertedData[$i][1]] - $this->monthPlan->days[
141                 $convertedData[$i][2]]->standbyHours > 0 - ($standbyTolerance *
142                 $standbyRun)) {
143                 $this->standbyPerson[$convertedData[$i][2]] = $convertedData[$i
144                     ][1];
145                 $quotaOfHours[$convertedData[$i][1]] -= $this->monthPlan->days[
146                     $convertedData[$i][2]]->standbyHours;
147             }
148         }
149     }
150     $standbyRun++;
151 }
152
153 // Bestimmung der "Metrik" des gerade eben erstellten Dienstplans
154 $currentDifference = $serviceRun * $serviceTolerance + $standbyRun *
155     $standbyTolerance;
156
157 if ($currentDifference < $smallestDifference) {
158     // wenn aktueller Dienstplan besser als der bisher Beste, dann als
159     // besten Dienstplan merken und kleinste Differenz aktualisieren
160     $smallestDifference = $currentDifference;
161     $servicePersonsBest = $this->servicePerson;
162     $standbyPersonsBest = $this->standbyPerson;
163 }
164
165 // Am Ende den besten der 1000 erstellten Dienstplaene speichern
166 $this->servicePerson = $servicePersonsBest;
167 $this->standbyPerson = $standbyPersonsBest;
168 }
169
170 ?>
```

Performance des Algorithmus

Die Laufzeit des Algorithmus hängt stark von den möglichen Terminen der Assistenten ab. Die durchschnittliche Laufzeit einer Dienstplanerstellung liegt bei einer



6. Implementierung



Millisekunde. Da 1000 Dienstpläne erstellt werden und der beste gewählt wird liegt die durchschnittliche Gesamtaufzeit bei einer Sekunde. Da der Klient nur einmal im Monat einen Dienstplan erstellen lässt, ist diese Laufzeit vertretbar.

Am Algorithmus ließen sich noch ein paar Dinge optimieren, aber nach dem Motto „**Vorsicht vor Optimierungen**“²³ von Clean Code Developer wurde zu Gunsten der Code-Lesbarkeit vorerst davon abgesehen.

6.17. Aufgaben-Verwaltung

Dem Klienten und seinem Team steht eine Aufgaben-Verwaltung (siehe Abbildung 6.23) zur Verfügung.

Erledigt

Einkaufen - fällig am 07.08.2014 - wiederholt sich alle 2 Tage ab dem Erledigungsdatum

Heute

Spülen - wiederholt sich jeden Tag ab dem Erledigungsdatum

Übermorgen

Einkaufen - wiederholt sich alle 2 Tage ab dem Erledigungsdatum

Zukünftig

Fenster putzen - fällig am 20.09.2014 - wiederholt sich alle 6 Monate ab dem Erledigungsdatum
 Steuererklärung - fällig am 01.03.2015 - wiederholt sich jedes Jahr ab dem Fälligkeitsdatum

Abbildung 6.23.: Aufgaben-Verwaltung mit einigen Test-Aufgaben

Eigenschaften der Aufgaben

Die Aufgaben haben folgende Eigenschaften:

- Beschreibung
- Fälligkeitsdatum
- Wiederholungstyp (wiederholt sich nie, wiederholt sich ab Erledigungsdatum, wiederholt sich ab Fälligkeitsdatum)
- Wiederholungsintervall (zum Beispiel alle 2 Tage) - vorausgesetzt, dass sich die Aufgabe wiederholt

²³<http://www.clean-code-developer.de/Roter-Grad.ashx> - zuletzt abgerufen am 29.08.2014



6. Implementierung

Gruppierung der Aufgaben

Die Aufgaben werden an Hand ihres Fälligkeitsdatums in folgende Gruppen sortiert:

- Erledigt
- Heute
- Morgen
- Übermorgen
- Zukünftig
- Ohne Datum

Neue Aufgaben erstellen

Der Klient hat als Einziger die Berechtigung neue Aufgaben zu erstellen. Hierfür hat er eine Eingabemaske, wie sie in Abbildung 6.24 zu sehen ist.

Neue Aufgabe: fällig am wiederholt sich Tage Erledigungsdatum

Abbildung 6.24.: Aufgaben-Verwaltung - Erstellen einer neuen Aufgabe



Bei der Auswahl des Fälligkeitsdatums wird er durch einen Kalender (siehe Abbildung 6.25) unterstützt. Dieser öffnet sich automatisch, sobald das Eingabefeld für das Datum den Fokus bekommt und schließt sich wieder, sobald der Klient ein Datum gewählt hat oder auf „Abbrechen“ klickt.

Aufgaben als erledigt markieren

Sowohl Klient als auch Assistenten können Aufgaben als erledigt markieren. Hierfür wird die Checkbox links der Aufgabenbeschreibung angeklickt. Die Aufgabe wird mit Hilfe von JavaScript durchgestrichen und in die Kategorie „Erledigt“ sortiert. Falls die Aufgabe ein Wiederholungsintervall besitzt, wird eine Kopie der Aufgabe mit neu berechnetem Fälligkeitsdatum erstellt. Weiterhin wird protokolliert, wer die Aufgabe als erledigt markiert hat.

Hat man aus Versehen eine Aufgabe als erledigt markiert, kann man dies rückgängig machen, indem man wieder die Checkbox links von der Aufgabenbeschreibung anklickt. Die Aufgabe ist nicht mehr durchgestrichen, wird in die richtige Kategorie (Heute, Morgen, ...) zurücksortiert und die gegebenenfalls erstellte Kopie der Aufgabe (mit neu berechnetem Fälligkeitsdatum) wird gelöscht.



6. Implementierung



Abbildung 6.25.: Aufgaben-Verwaltung: Wahl des Datums

Speichern

Um Änderungen an den Aufgaben (neue Aufgaben erstellt oder Aufgaben erledigt) zu sichern, muss man auf den Knopf „Speichern“ drücken.

Implementierungsdetails

Die Aufgaben-Seite wird vom PHP-Script `ToDoManagerView.php` erstellt. Der größte Teil der Funktionalität ist clientsseitig im JavaScript `ToDoManager.js` implementiert. Hier wird viel mit dem `Date`-Objekt gearbeitet, das mit dem Prototyping-Mechanismus (siehe Abschnitt 2.2) in der Datei `DateExtended.js` um einige Funktionen erweitert wurde. Weiterhin ist in der Datei `ToDoManager.js` die Klasse `ToDo` enthalten. Der `DatePicker` ist in der Datei `datePicker.js` definiert.

Das PHP-Script `ToDoSaver.php` speichert die offenen Aufgaben in die Datei `toDos.txt` und die erledigten Aufgaben in die Datei `done.txt`. Beide Dateien liegen im Ordner `ToDoManager`.

6.18. Feedback

Mit der Feedback-Funktion hat der Benutzer die Möglichkeit Kontakt zum Entwickler des Assistenzplaners aufzunehmen. Es gibt ein Formular (siehe Abbildung 6.26), worin Feedback, Änderungswünsche und Kritik geäußert werden können. Wenn für



6. Implementierung

den Benutzer eine E-Mail-Adresse hinterlegt ist (Klient \Rightarrow Einstellungen, Assistent \Rightarrow Team-Verwaltung), so wird diese automatisch als E-Mail-Adresse für eine Antwort vorgeschlagen. Der Benutzer kann aber auch eine andere Antwort-Adresse angeben.

Feedback

E-Mail Adresse für Antwort:

Abbildung 6.26.: Eingabemaske für das Feedback

Implementierungsdetails

Die Generierung des Feedback wird vom PHP-Script `feedbackView.php` durchgeführt. Drückt der Benutzer auf den Button „Feedback abschicken“, so wird die Funktion `sendFeedback` im JavaScript `feedback.js` aufgerufen. Diese Funktion schickt das Feedback per POST-Parameter an das PHP-Script `feedbackSender.php`. Dieses Script erstellt im Wesentlichen eine Instanz von `PHPMailer`²⁴ und verschickt eine E-Mail an den Entwickler des Assitenzplaners. Die E-Mail-Adresse für die Antwort ist in der versendeten E-Mail enthalten, so dass der Entwickler den direkten Kontakt mit dem Benutzer, der das Feedback formuliert hat, aufnehmen kann.

6.19. Dokumentation

Für die Benutzer des Assitenzplaners steht jederzeit im Navigations-Bereich (oben rechts) ein Link zur Verfügung, der zur Dokumentation führt. Der Dokumentations-Inhalt wird individuell erzeugt. Klienten bekommen eine andere Dokumentation zu Gesicht als Assistenten, da beide auf unterschiedliche Weise mit dem Assitenzplaner arbeiten. Weiterhin werden die Einstellungen berücksichtigt, zum Beispiel wird das Dokumentations-Kapitel über die Aufgaben-Verwaltung nur angezeigt, wenn diese auch verwendet wird.

Im oberen Bereich der Dokumentation werden Inhalts-, Abbildungs- und Tabellenverzeichnisse angezeigt, damit man leicht zum entsprechenden Kapitel der Dokumentation navigieren kann.

²⁴<https://github.com/PHPMailer/PHPMailer> - zuletzt abgerufen am 12.09.2014



6. Implementierung

Implementierungsdetails

Die Dokumentations-Seite wird vom PHP-Script `documentation.php` erstellt. Abhängig von den Variablen, die im superglobalen Array `$_SESSION` gesetzt sind, wird der variiert. Es werden weitere PHP-Scripts²⁵ per lude eingebunden.

Im oberen Bereich der Dokumentation werden mit Hilfe des JavaScripts `documentation.js` automatisch Inhalts-, Abbildungs- und Tabellenverzeichnisse erstellt. Für die Erstellung des Inhaltsverzeichnisses werden alle Überschriften²⁶ geparsst und deren Inhalte ausgelesen. Die Erstellung der Abbildungs- und Tabellenverzeichnisse funktioniert analog.

Wenn in Abbildungen (HTML-Tag: `img`) oder Tabellen (HTML-Tag: `table`) das Attribut `title` gesetzt ist, so werden Abbildungs- bzw. Tabellenunterschriften mit dem Inhalt des `title`-Attributes erstellt.

6.20. Impressum

Um eine Abmahnung zu vermeiden urde dem Assistenzplaner eine Seite hinzugefügt, die jederzeit über die Navigation zu erreichen ist. Diese beinhaltet das Impressum mit allen Angaben gemäß § 5 Telemediengesetz, einen Haftungsausschuss (Disclaimer) sowie eine Datenschutzerklärung.

Implementierungsdetails

Das Impressum ist im PHP-Script `impressum.php` hinterlegt.

²⁵Die weiteren PHP-Scripts beginnen alle mit dem Präfix `documentation`, zum Beispiel `documentationCalendar.php`

²⁶HTML-Tags: `h1`, `h2`, `h3`, `h4`, `h5`, `h6`



7. Überführung in den Produktiv-Betrieb

Die Überführung in den Produktiv-Betrieb wurde iterativ durchgeführt.

Den Dienstplan für den Monat April 2014 hat der Klient noch komplett manuell erstellt.

Schon bei der Dienstplanerstellung für den Mai 2014 wurde der Klient technisch unterstützt. Es war möglich die Termin-Antworten der Assistenten übersichtlich in einer Tabelle darzustellen. Somit musste der  Klient nicht mehr sieben E-Mail Antworten gleichzeitig überblicken. Es war möglich Dienste und Bereitschaften manuell einzutragen und den Dienstplan zu speichern. Der Autor war bei der Dienstplanerstellung dabei und hat viel über die Hintergründe erfahren und herausfinden können, worauf es dem Klienten ankommt.

Bei der Dienstplanerstellung für Juni 2014 hat schon ein Algorithmus im Hintergrund gearbeitet und Dienstplanvorschläge erstellt. Der Klient hat den Dienstplan eigenständig in Anwesenheit des Autors erstellt.

Bei der Dienstplanerstellung für Juli 2014 waren alle nötigen Komponenten vorhanden. Zur Sicherheit haben der Klient und der Autor nochmal gemeinsam den Dienstplan erstellt und verifiziert, dass alle Komponenten ordnungsgemäß funktionieren.

Ab dem Dienstplan für den Monat August 2014 haben der Klient und die Assistenten ihre Eingaben selbstständig getätigt. Es war kein Zutun des Autors mehr notwendig. Der Algorithmus zur Dienstplanerstellung hat einen ausgewogenen Dienstplan erstellt, der vom Klienten angenommen wurde.

Durch die iterative Vorgehensweise fühlte sich der Klient sehr gut eingebunden. Er konnte in regelmäßigen Intervallen den aktuellen Stand der Software sehen und sein Feedback äußern, was direkt in die weitere Entwicklung einfloss.





8. Ergebnis

Der **Assistenzplaner**²⁷ wurde als serverbasierte Anwendung geschaffen, um körperlich behinderte Menschen (Klienten) und ihre Assistenz-Teams bei der Planung zu unterstützen.

Der Klient hat die Möglichkeit sein Assistenz-Team zu verwalten. Dabei kann der Klient neben den Kontaktdaten auch Priorisierungswerte und Stundenkontigente der Assistenten definieren.

Der Klient kann mit dem Monats-Plan die Dienstzeiten für einen Monat festlegen. Seine Assistenten werden per Knopfdruck benachrichtigt und gebeten ihre Termine in einen Kalender einzutragen.

Ein Algorithmus erstellt automatisch einen Dienstplanvorschlag, der alle Vorlieben des Klienten und die möglichst gleichmäßige Ausschöpfung der Stundenkontigente berücksichtigt. Der Klient kann sich mehrere Dienstplanvorschläge erstellen lassen. Weiterhin ist es möglich den Dienstplan manuell anzupassen.

Eine Aufgaben-Verwaltung rundet den Assistenzplaner ab.

Die Webanwendung wurde mit vielen gängigen Browsern (Firefox, Chrome, Safari) erfolgreich getestet.

²⁷<http://assistenzplaner.de/>



9. Fazit und Ausblick

Es wurde eine Webanwendung erschaffen, die dem Klienten sehr viel Arbeit abnimmt. Der Klient ist mit der Lösung sehr zufrieden und wird in Zukunft nur noch den Assistenzplaner zur Dienstplanerstellung verwenden. Alle gewünschten Benutzergeschichten (siehe Abschnitt 3.4) wurden umgesetzt.

Die entstandene Lösung ist voll funktionsfähig und in sich geschlossen. Dennoch gibt es einige Stellen mit Verbesserungs- oder Erweiterungspotential.

Der Autor plant, das Projekt weiterzuführen und den **Assistenzplaner**²⁸ kontinuierlich zu verbessern und zu erweitern. Die Quellen des Assitenzplaners sind Open-Source und in einem öffentlichen Repository bei **GitHub**²⁹ eingeccheckt. So hat jeder die Möglichkeit sich den Assitenzplaner auf dem eigenen Server einzurichten.



9.1. Einsatz bei mehreren Teams

Der Assistenzplaner ist derzeit (September 2014) eine spezialisierte Lösung für einen Klienten mit seinem Team. Der Autor hatte am 01.09.2014 ein Gespräch mit dem **Landesbeauftragten für die Belange behinderter Menschen in Rheinland Pfalz - Matthias Rösch**.³⁰ Hier hat der Verfasser den Assistenzplaner vorgestellt und sich vom Landesbeauftragten einige Ansprechpartner im Bereich Assistenzplanung nennen lassen. Der **Landesbeauftragte** hat selbst sechs Assistenten, erstellt derzeit seine Dienstpläne manuell und war sehr interessiert am Assistenzplaner.

Der Autor hofft, dass sich einige Beta-Tester bereit machen den Assistenzplaner einzusetzen. Sicherlich werden noch Anforderungen hinzu kommen, da jeder Klient mit seinem Team einen anderen Prozess und unterschiedliche Rahmenbedingungen bei der Dienstplanerstellung hat.

Sobald der Assistenzplaner von einigen Klienten mit ihren Teams erfolgreich erprobt ist, möchte der Autor den Assistenzplaner öffentlich bewerben.³¹ Einige mögliche Kanäle sind nachfolgend aufgezählt:

²⁸ <http://assistenzplaner.de>

²⁹ <https://github.com/Ulle84/AssistancePlaner>

³⁰ <http://inklusion.rlp.de/der-landesbeauftragte/> - zuletzt abgerufen am 02.09.2014

³¹ Der Autor hat keinerlei finanzielles Interesse. Die Nutzung des Assistenzplaners ist kostenlos und wird es auch immer bleiben.



9. Fazit und Ausblick

- Blog von Matthias Rösch, Landesbeauftragter für die Belange behinderter Menschen in Rheinland-Pfalz³²
- kabinet Nachrichten - Online-Nachrichten von und für behinderte Menschen³³
- ForseA - Forum selbstbestimmter Assistenz behinderter Menschen e.V.³⁴
- Flyer im ZSL Mainz³⁵, ZSL Bad Kreuznach³⁶, Commit Mainz³⁷ und weiteren Einrichtungen hinterlegen
- Nachricht über den E-Mail-Verteiler des Landesbeauftragten 

9.2. Mögliche Verbesserungen

In diesem Abschnitt werden mögliche Verbesserungen des Assistenzplaners vorgestellt und diskutiert.

Barrierefreiheit

„Barrierefreiheit bezeichnet im deutschen Sprachgebrauch eine Gestaltung der baulichen Umwelt in der Weise, dass sie von Menschen mit Behinderung in der selben Weise genutzt werden kann wie von Menschen ohne Behinderung.“ [Bremus 2013, S. 28]

„Barrierefreiheit als Thema für Webentwickler und -designer wird auch deshalb immer wichtiger und interessanter, weil die Menschen eine immer höhere Lebenserwartung haben, und gerade Menschen mit Behinderungen immer besser medizinisch versorgt werden können.“ [Bremus 2013, S. 29]

„Eine barrierefreie Webanwendung basiert grundsätzlich auf zwei Säulen: der Accessibility und der Usability.“ [Bremus 2013, S. 30]

„Unter Accessibilty ist die Zugänglichkeit bzw. Erreichbarkeit in Bezug auf eine Anwendung zu verstehen. Sie beschreibt die Fähigkeit, Informationen für jeden Benutzer zugänglich zu machen, unabhängig von technischen und körperlichen Voraussetzungen und Einschränkungen.“ [Bremus 2013, S. 31]

³² <http://inklusion-blog.rlp.de> - zuletzt abgerufen am 02.09.2014

³³ <http://www.kabinet-nachrichten.org> - zuletzt abgerufen am 02.09.2014

³⁴ <http://www.forsea.de> - zuletzt abgerufen am 02.09.2014

³⁵ <http://www.zsl-mz.de> - zuletzt abgerufen am 02.09.2014

³⁶ <http://www.zsl-bad-kreuznach.org> - zuletzt abgerufen am 02.09.2014

³⁷ <http://www.commit-cbf.de> - zuletzt abgerufen am 02.09.2014



9. Fazit und Ausblick

„Die Usability strebt nach der idealen Strukturierung von Informationen, um eine effiziente Nutzung von Informationen und Technologie überhaupt erst zu ermöglichen.“ [Bremus 2013, S. 31]

Da der Assistenzplaner langfristig von vielen körperlich behinderten Menschen bedient werden soll, ist eine Untersuchung der entstandenen Softwarelösung auf gute Accessibilty und Usability notwendig. Es muss verifiziert werden dass die Seite im Idealfall barrierefrei, mindestens jedoch barrieararm ist.³⁸



AJAX tiefer verwurzeln

Derzeit wird AJAX nur auf Knopfdruck eingesetzt. An vielen Stellen muss der Benutzer explizit auf einen Speichern-Button drücken, um die eingegebenen Daten persistent zu sichern. In einer moderaten Webanwendung ist dies ungewöhnlich. Besser wäre es, wenn Änderungen, die der Benutzer vornimmt, direkt per AJAX an den Server geschickt werden und der Server die Daten unverzüglich persistent ablegt.

Responsive Webdesign

Responsive (reaktionsfähiges) Webdesign bedeutet auf „die steigende Anzahl verschiedener Display-Größen und Geräte angemessen zu reagieren“ [Zillgens 2013, S. 8].

Auf Desktops und Tablets wird der Assistenzplaner ordentlich dargestellt und kann gut bedient werden. Auf Smart-Phones ist das Aussehen und die Bedienung verbessert würdig (siehe Abbildung 9.1). Die Eingabefelder sind klein und die Display-Größe kann besser ausgenutzt werden.

Bei der Optimierung des Assistenzplaners für Handys muss umgedacht werden. Man muss wegkommen von dem horizontal-basierten Layout mit Spalten hin zu einem linear-vertikalen Layout [Bieh 2008, S. 67].

Team-Übersicht überarbeiten

Derzeit wird das Team mit Hilfe einer Tabelle verwaltet. Zu Beginn der Entwicklung war das die richtige Entscheidung. Im Laufe der Entwicklung wurden Spalten ergänzt und die Tabelle wurde mit der Zeit immer breiter (siehe Abbildung 6.11). Eine

³⁸ „Verstärkt setzt sich in der letzten Zeit der Begriff des barriearmen Webdesigns durch. Das beruht auf der Tatsache, dass eine 100%-ige Barrierefreiheit einer Webanwendung nicht zu erreichen ist. Verschiedene Benutzerinteressen, Hilfsmittel und technische Voraussetzungen machen die Barrierefreiheit einer Anwendung unmöglich.“ [Bremus 2013, S. 29]



9. Fazit und Ausblick

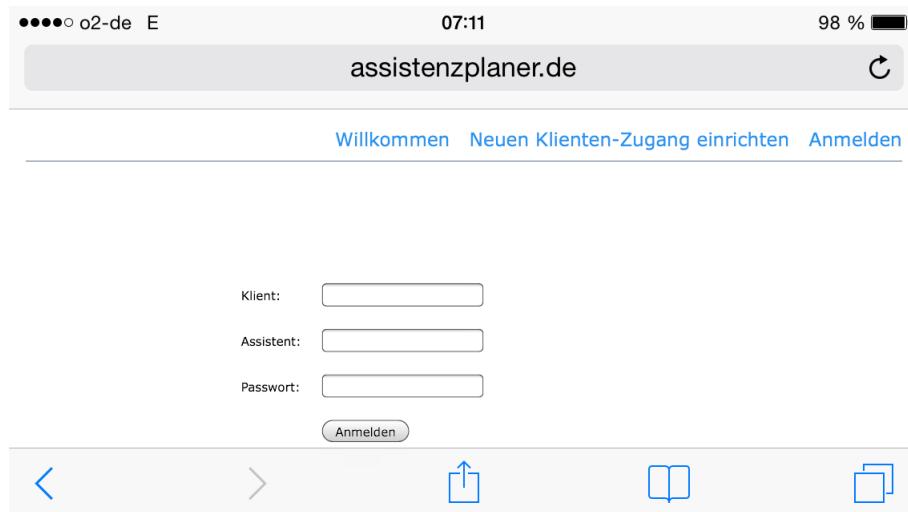


Abbildung 9.1.: Darstellung des Assistenzplaner auf einem Smart-Phone..

mögliche Alternative zur breiten Tabelle ist in Abbildung 9.2 gezeigt. Für jeden Assistenten gibt es eine eigene Übersicht der Eigenschaften.

Markus

Vorname	Markus
Nachname	Mustermann
E-Mail Adresse	markus@mustermann.de
Telefonnummer	0800-555000
Stichwörter	FCB
Stundenkontingent	200
Priorisierung	10
Bevorzugte Tage	<input type="checkbox"/> Mo <input type="checkbox"/> Di <input type="checkbox"/> Mi <input type="checkbox"/> Do <input type="checkbox"/> Fr <input type="checkbox"/> Sa <input type="checkbox"/> So
Aktionen	Editieren Löschen Passwort zurücksetzen

Abbildung 9.2.: Mögliche neue Darstellungsform eines Assistenten in der Team-Übersicht

Derzeit hat nur der Klient Einblick in die Team-Verwaltung. Es wäre wünschenswert, wenn Assistenten untereinander ihre Kontaktdaten (die ersten vier Zeilen in Abbildung 9.2) freigeben könnten.

Überarbeitung der Dokumentation

Derzeit ist der Assistenzplaner für Klienten und Assistenten auf einer zentralen Seite (siehe Abschnitt 6.19) dokumentiert. Wünschenswert wäre es, wenn kontextsensitive



9. Fazit und Ausblick

Informationen zusätzlich über den gesamten Assistenzplaner kontextsensitiv leicht zugänglich verteilt wären. Eine Möglichkeit der Realisierung besteht mit Hilfe von **jQuery UI Tooltip**³⁹ (siehe Abbildung 9.3). Sobald der Benutzer in das Feld „Lastname“ klickt, bekommt er einen Hilfstext (Tooltip) angezeigt.

Tooltip

Customizable, themeable tooltips, replacing native tooltips.

Abbildung 9.3.: jQueryUI Tooltip

Erhöhung der Sicherheit

* Passwortlänge * Hin und wieder neues Passwort vergeben * Passwortinhalt - Buchstaben + Zahlen * Captcha beim Anlegen eines neuen Klienten-Kontos * E-Mail Adresse des Klienten zur Verifizierung - verhindern dass Bots Konten anlegen können

Verwendung von Frameworks

Derzeit ist alles von Grund auf händisch programmiert. Der Autor hat sich bewusst gegen den Einsatz von Frameworks entschieden, um genau zu verstehen, was im Hintergrund passiert und jederzeit die Kontrolle zu haben. Es ist ein schlankes System entstanden, das unabhängig von Drittanbietern ist.

Unter dem Gesichtspunkt, dass der Autor langfristig eventuell nicht der einzige Entwickler des Assistenzplaners sein wird, wäre es wünschenswert auf Frameworks aufzubauen, die verbreitet sind. Ein Beispiel für ein weit verbreites JavaScript-Framework ist **jQuery**.⁴⁰ Ein weiterer Vorteil wäre, dass die Abdeckung aller gängigen Browser garantiert wäre.

³⁹ <http://jqueryui.com/tooltip/> - zuletzt abgerufen am 22.08.2014

⁴⁰ <http://jquery.com> - zuletzt abgerufen am 29.08.2014



9. Fazit und Ausblick

Serverseitig sei das PHP-Framework [Symfony](#)⁴¹ erwähnt, dass das Prinzip des *Model View Controller (MVC)* konsequent einsetzt.

9.3. Mögliche Erweiterungen

In diesem Abschnitt werden mögliche Erweiterungen des Assistenzplaners vorgestellt und diskutiert.

Mehrere Dienste pro Tag

Derzeit ist nur ein Dienst pro Tag planbar. Um noch mehr Klienten zu erreichen wäre es sinnvoll, wenn mehrere Dienste pro Tag planbar wären.

Internationalisierung

Derzeit ist der Assistenzplaner komplett in Deutsch. Um auch andere Sprachen zu unterstützen wäre ein grundlegender Umbau notwendig. Weiterführende Informationen finden sich in der offiziellen [PHP-Dokumentation](#).⁴²

iCalendar Export

Im iCalendar-Format können Termine ausgetauscht werden. Ein Beispiel für ein Ereignis im iCalendar Format:

Listing 9.1: Beispiel für ein iCalendar-Termin

```

1 BEGIN:VCALENDAR
2 VERSION:2.0
3 PRODID:http://www.assistenzplaner.de/
4 METHOD:PUBLISH
5 BEGIN:VEVENT
6 UID:ae993c01a0c40fb4a93b9a79363502df@assistenzplaner.de
7 ORGANIZER;CN="Klient":MAILTO:info@assistenzplaner.de
8 LOCATION:Heim des Klienten
9 SUMMARY:Assistenz/Bereitschaft
10 DESCRIPTION:Assistenz/Bereitschaft + Notizen
11 CLASS:PUBLIC
12 DTSTART:20140822T140000Z
13 DTEND:20140823T090000Z

```

⁴¹ <http://symfony.com> - zuletzt abgerufen am 29.08.2014

⁴² <http://php.net/manual/en/book.intl.php> - zuletzt abgerufen am 21.08.2014



9. Fazit und Ausblick

14 DTSTAMP:20140822T064500Z

15 END:VEVENT

16 END:VCALENDAR

In dem Ereignis ist der Dienst vom 22.08.2014 14:00 Uhr bis 23.08.2014 9:00 Uhr beschrieben.

Für die Assistenten wäre es interessant ihre Dienst- und Bereitschaftszeiten im iCalendar-Format abbonieren zu können. So hätten die Assistenten jederzeit aktuell ihre Arbeitszeiten verfügbar, ohne ständig online auf dem Assistenzplaner nachschauen zu müssen. Ein iCalendar lässt sich leicht in den eigenen Online-Kalender integrieren, so dass sich die Dienst- und Bereitschaftszeiten jederzeit von unterwegs auf dem Smart-Phone einsehen lassen.

Feiertage importieren

Für die Erstellung des Monatsplans wäre es für den Klienten hilfreich, wenn man gesetzliche Feiertage importieren könnte. Im Idealfall würde sogar das Bundesland, in dem der Klient lebt, berücksichtigt werden. Zum Beispiel könnte man auf der Webseite [iFeiertage⁴³](#) einen Feiertage-Kalender abonnieren. Dieser wäre als [iCalendar⁴⁴](#) verfügbar.

Denkbar wäre, dass der Klient beim Monatsplan einen Button „Feiertage importieren“ zur Verfügung gestellt bekommt. Betätigt er diesen Knopf, so werden alle Feiertage für diesen Monat in den öffentlichen Notizen eingetragen. Weiterhin werden die Dienstzeiten auf 24-stündige Dienste angepasst, da der Klient an Feiertagen rund um die Uhr Assistenz benötigt.

Logging

Derzeit ist nicht nachvollziehbar, wer wann welche Änderungen am Assistenzplaner vorgenommen hat. Für zukünftigen Support und Fehlersuche wäre es wünschenswert ein Log-File zu nutzen. Daraus wäre bei Bedarf ersichtlich, welche Änderungen in der letzten Zeit gemacht wurden und von wem diese durchgeführt wurden.

⁴³ <http://www.ifeiertage.de> - zuletzt abgerufen am 05.09.2014

⁴⁴ <webcal://ifeiertage.de/rp.ics> - zuletzt abgerufen am 05.09.2014



9. Fazit und Ausblick

Aufgaben-Verwaltung erweitern

Die Aufgaben-Verwaltung wurde nur rudimentär entwickelt. Schon zu einem frühen Zeitpunkt hat der Klient entschieden die Aufgaben-Verwaltung für sein Team nicht einzusetzen, da der Mehraufwand durch die Verwaltung zu groß wäre. Aufgrund dieser Entscheidung wurde die Entwicklung nach einer Woche wieder eingestellt. Für andere Modelle in der Pflege wäre es aber eventuell von Vorteil die Aufgaben-Verwaltung zu nutzen. Mögliche Erweiterungen werden in den folgenden Abschnitten dargestellt.

Hervorhebungen bei Datumswahl

In der Datumswahl (siehe Abbildung 6.25) sollte der aktuelle Tag sowie der gewählte Tag farblich hervorgehoben werden.

Aufgaben editieren

Aufgaben lassen sich derzeit nur anlegen, nicht im Nachhinein verändern. Eine Editierfunktion wäre von großem Nutzen.

Aufgabe löschen

Aufgaben lassen sich derzeit nur anlegen, aber nicht löschen.

Test Todo

Übersicht erledigter Aufgaben



Von Vorteil (gerade für die Nachweise der Leistungserbringung) wäre es, wenn nachvollziehbar wäre, welcher Assistent wann welche Aufgabe erledigt hat.



10. Literaturverzeichnis

Bieh 2008

BIEH, M.: *Mobiles Webdesign: Konzeption, Gestaltung, Entwicklung ; [Struktur, Design und Programmierung ; Umsetzung mit (X)HTML, CSS und PHP ; Standards und Best Practices]*. Galileo Press, 2008 (Galileo computing). – ISBN 9783836211536 9.2

Bremus 2013

BREMUS, T: *Barrierefreiheit: Webanwendungen ohne Hindernisse*. entwickler Press, 2013. – ISBN 9783868020953 9.2, 38

Comer 2002

COMER, D.: *Computernetzwerke und Internets: mit Internet-Anwendungen*. Pearson Studium, 2002 (I Informatik). – ISBN 3-8273-7023-X 2.1, 2.1, 2.1, 2.1, 2.1, 2.1, 2.1, 2.1

Flanagan 2002

FLANAGAN, D.: *JavaScript: das umfassende Referenzwerk*. O'Reilly, 2002. – ISBN 3-89721-330-3 2.2

Koch 2011

KOCH, S.: *JavaScript: Einführung, Programmierung und Referenz*. dpunkt.Verlag, 2011 (iX-Edition). – ISBN 9783898647311 2.2

Münz und Gull 2012

MÜNZ, S. ; GULL, C.: *HTML5-Handbuch*. Franzis, 2012 (Know-how ist blau). – ISBN 9783645601511 2.2, 2.2

Pichler 2008

PICHLER, R.: *Scrum: agiles Projektmanagement erfolgreich einsetzen*. dpunkt.Verlag, 2008. – ISBN 978-3-89864-478-5 3.4

Rupp u. a. 2012

RUPP, C. ; QUEINS, S. ; SOPHISTEN, die: *UML 2 glasklar: Praxiswissen für die UML-Modellierung*. Carl Hanser Verlag, 2012. – ISBN 978-3-446-43057-0 2.3



Schlossnagle 2006

SCHLOSSNAGLE, G.: *Professionelle PHP 5-Programmierung - Entwicklerleitfaden für große Webprojekte mit PHP 5*. Addison-Wesley, 2006 (open source library). – ISBN 978-3-8273-2381-1 6.4

Teague 2011

TEAGUE, J.C.: *CSS3*. Peachpit Press, 2011 (Visual quickstart guide). – ISBN 9780321719638 2.2

Theis 2010

THEIS, T.: *Einstieg in PHP 5.3 und MySQL 5.4*. Galileo Press, 2010 (Galileo computing). – ISBN 9783836215442 6.1

Weilkiens 2014

WEILKIEENS, T.: *Systems engineering mit SysML/UML: Anforderungen, Analyse, Architektur*. dpunkt.Verlag, 2014. – ISBN 978-3-86490-091-4 5.1, 5.1, 5.1, 5.1

Wirdemann 2011

WIRDEMANN, R.: *Scrum mit User Stories*. Carl Hanser Verlag, 2011. – ISBN 978-3-446-42660-3 3.4

Zillgens 2013

ZILLGENS, C.: *Responsive Webdesign Reaktionsfähige Websites gestalten und umsetzen*. Carl Hanser Verlag, 2013. – ISBN 978-3-446-43015-0 9.2

*A. Tabellenverzeichnis*

A. Tabellenverzeichnis

6.1.	Übersicht der verwendeten Daten aus der Team-Verwaltung	61
6.2.	Punktetabelle zur Dienstplanerstellung	61
6.3.	Priorisierungswerte aus der Team-Verwaltung	62
6.4.	Punktetabelle nach Multiplikation mit Priorisierungswerten	62
6.5.	Beispielhafter Dienstplan	62
6.6.	Punktetabelle nach Berücksichtigung der Wochentage	63
6.7.	Punktetabelle nach Berücksichtigung der Stichwörter	63
6.8.	Umgewandelte Punktetabelle	64
6.9.	Umgewandelte Punktetabelle, randomisiert und sortiert nach Punkten	64



B. Abbildungsverzeichnis

2.1. Ergebnis Zusammenspiel HTML, CSS und JavaScript	16
4.1. Screenshot QTime	31
5.1. SysML Anforderungsdiagramm - Enthältbeziehung	34
5.2. SysML Anforderungsdiagramm - Erfüllungsbeziehung	34
5.3. SysML Anforderungsdiagramm - Verfeinerungsbeziehung	34
5.4. SysML Anforderungsdiagramm	35
5.5. UML Komponentendiagramm	36
5.6. Klassendiagramm der Komponente MonthNavigation	37
5.7. Klassendiagramm der Komponente MonthOrganisation	38
5.8. Klassendiagramm der Komponente Roster	39
5.9. Klassendiagramm der Komponente Settings	40
5.10. Klassendiagramm der Komponente TeamOrganisation	41
5.11. Klassendiagramm der Komponente ToDoManager	42
6.1. Login-Maske des Assistenzplaners	45
6.2. Session-ID als Cookie im Browser	46
6.3. Willkommen-Seite	47
6.4. Navigation bei angemeldetem Assistenten	47
6.5. Navigation ohne Anmeldung	47
6.6. Monatsnavigation	48
6.7. Anlegen eines neuen Klienten	49
6.8. Übersicht Assistenzplaner für Klienten	50
6.9. Einstellungsmöglichkeiten des Assistenzplaners	51
6.10. Standard-Dienstzeiten	52
6.11. Verwaltung Assistenz-Team	53
6.12. Monatsplan	54
6.13. Benachrichtigung der Assistenten	54
6.14. Kalender für die Eingabe möglicher Termine	55
6.15. Bemerkungen des Klienten	56
6.16. Editierbarer Dienstplan	57
6.17. Stundenübersicht Assistenten	57
6.18. Gesamter Dienstplan aus Sicht des Assistenten	59

*B. Abbildungsverzeichnis*

6.19. Dienste und Bereitschaften des angemeldeten Assistenten	59
6.20. Verteilung der Dienste mit 0 Stunden Toleranz	65
6.21. Stundenübersicht mit 4 Stunden Toleranz	65
6.22. Verteilung der Dienste mit 4 Stunden Toleranz	66
6.23. Aufgaben-Verwaltung	71
6.24. Erstellen einer neuen Aufgabe	72
6.25. Wahl des Datums	73
6.26. Eingabemaske für das Feedback	74
9.1. Darstellung des Assitzenzplaners auf einem Smart-Phone	81
9.2. Mögliche neue Darstellungform des Teams	81
9.3. jQueryUI Tooltip	82
F.1. Infodialog der Tex-Distribution	94
F.2. Infodialog des TeXMakers	95
F.3. Infodialog der IDE PhpStorm	96
F.4. Infodialog des FTP-Programms Cyberduck	96
F.5. Infodialog des Programs GitHub	97
F.6. Infodialog des Programs Firefox	97
F.7. Infodialog der Firefox-Erweiterung Firebug	97
F.8. Verwendete PHP Version	98



2.1. Beispiel für ein XML Dokument	12
2.2. Beispiel für ein HTML-Dokument	13
2.3. Beispiel für ein CSS-Dokument	14
2.4. Beispiel für ein JavaScript-Dokument	15
2.5. Beispiel für die Erweiterung eines bestehenden Objekts in JavaScript	15
2.6. Beispiel für ein PHP-Dokument	17
6.1. Anlegen der Monatsnavigation	48
6.2. Algorithmus zur Dienstplanerstellung	66
9.1. Beispiel für ein iCalendar-Termin	83



D. Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
CSS	Cascading Stylesheets
DOM	Document Object Model
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	Hypertext Transport Protocol
ID	Identifikator
IDE	Integrated Development Environment
MVC	Model View Controller
PDF	Portable Document Format
PHP	PHP Hypertext Preprocessor
SysML	System Modeling Language
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language



E. Inhalt der beiliegenden CD

Auf den Ordner der beiliegenden CD sind folgende Daten enthalten:

- **BibTeX** ⇒ BibTeX-Datei, in der alle Literaturquellen enthalten sind
- **CSS** ⇒ CSS-Dateien, in denen die Style-Formatierungen enthalten sind
- **Data** ⇒ Die Daten, die der Assistenzplaner persistent speichert
- **ExternalResources** ⇒ FreePDF und PHPMailer
- **HTML** ⇒ HTML-Dateien
- **iCalendar** ⇒ iCalendar-Beispiel-Datei
- **Images** ⇒ Screenshots und weiter Bilder
- **JavaScript** ⇒ JavaScript-Dateien
- **LaTeX** ⇒ Quelldateien dieser Ausarbeitung
- **LibreOffice** ⇒ Tabelle
- **Modell** ⇒ UML- und SysML-Modelle
- **PDF** ⇒ PDF-Dokumente
- **PHP** ⇒ PHP-Dateien
- **XML** ⇒ XML-Beispiel-Datei



F. Verwendete Software

F. Verwendete Software

In diesem Abschnitt wird erläutert, welche Software für die Erstellung dieser Master-Thesis eingesetzt wurde.

LATEX

Die vorliegenden Ausarbeitung wurde mit LATEX geschrieben. Verwendet wurde die Distribution TeXLive (Details siehe Abbildung F.1). Als Editor wurde TeXMaker (Version siehe Abbildung F.2) verwendet.

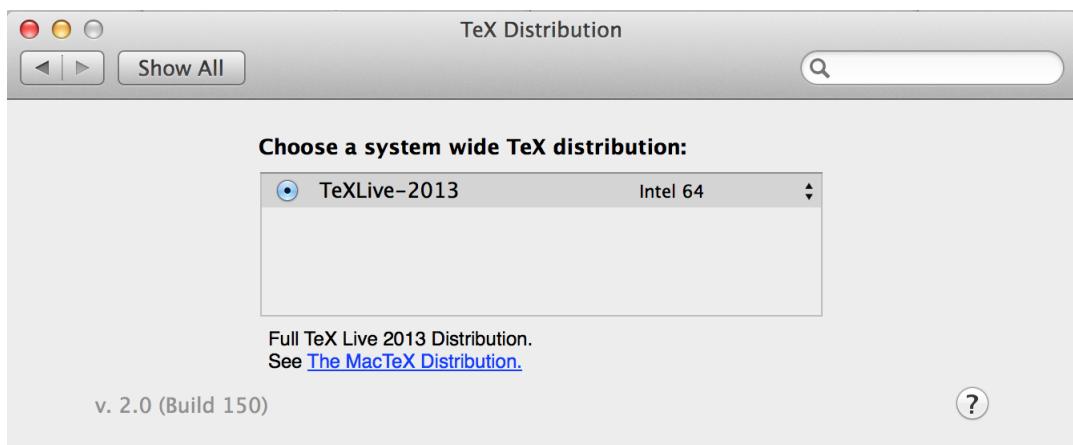


Abbildung F.1.: Infodialog der Tex-Distribution

PhpStorm

Als *Integrated Development Environment (IDE)* für die Entwicklung von PHP, HTML, CSS und JavaScript wurde PhpStorm (Version siehe Abbildung F.3) verwendet.



F. Verwendete Software

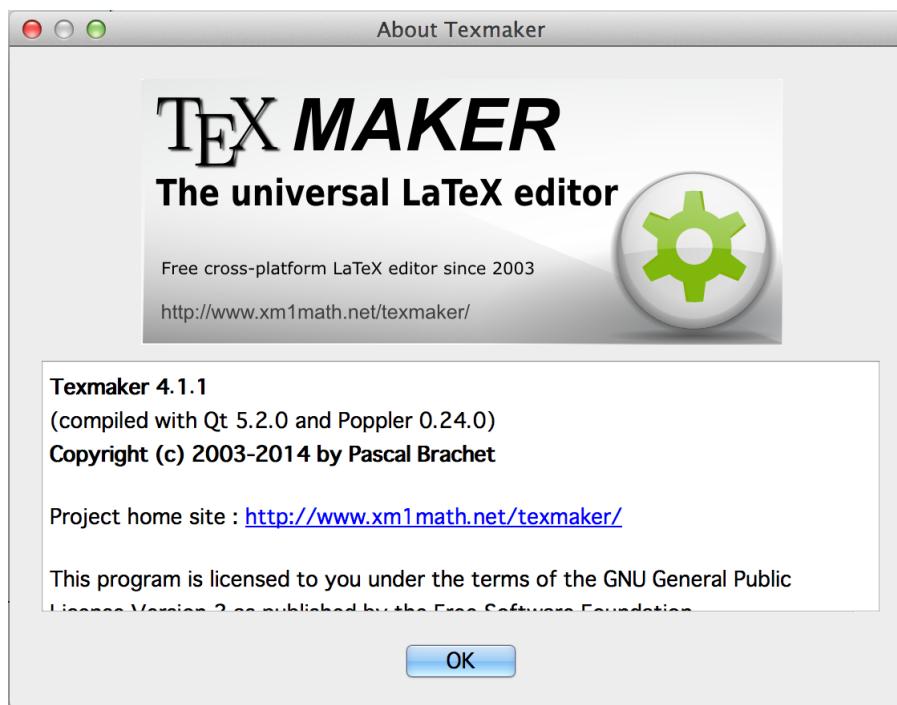


Abbildung F.2.: Infodialog des TeXmakers

Cyberduck

Als Client für den Transfer von Daten zwischen Entwicklerlaptop und Server via *File Transfer Protocol (FTP)* wurde Cyberduck (Version siehe Abbildung F.4) eingesetzt.

GitHub

Der Autor hat ein öffentliches Repository auf [GitHub](#)⁴⁵ erstellt.

Als Client für den Transfer von Daten zwischen Entwicklerlaptop und GitHub-Server wurde GitHub (Version siehe Abbildung F.5) eingesetzt.

Firefox

Zum Testen des HTML und CSS-Codes wurde Firefox (Version siehe Abbildung F.6) eingesetzt.

Zum Debuggen von JavaScript-Code kam die Firefox-Erweiterung Firebug (Version siehe Abbildung F.7) zum Einsatz.

⁴⁵ <https://github.com/Ulle84/AssistancePlaner>



F. Verwendete Software

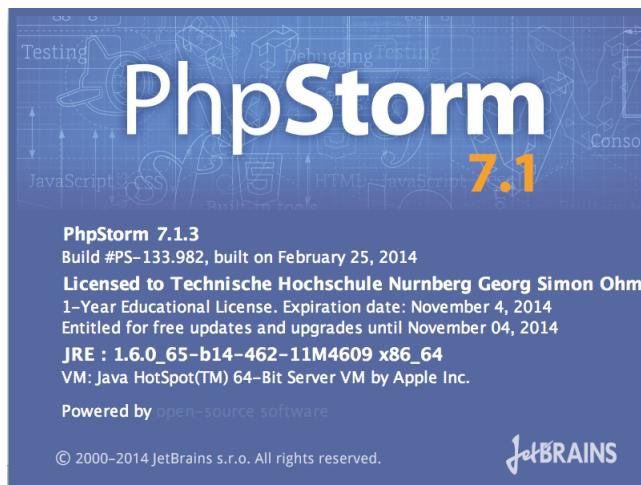


Abbildung F.3.: Infodialog der IDE PhpStorm



Abbildung F.4.: Infodialog des FTP-Programms Cyberduck

PHP

PHP wurde in der Version 5.5 (siehe Abbildung F.8) eingesetzt.

Weiterhin wurden die Klassen **PHPMailer**⁴⁶ und **FPDF**⁴⁷ verwendet. Beide sind frei verfügbar und bedenkenlos in dem Open-Source-Projekt Assistenzplaner einsetzbar.

⁴⁶ <https://github.com/PHPMailer/PHPMailer> - zuletzt abgerufen am 12.09.2014

⁴⁷ <http://www.fpdf.org> - zuletzt abgerufen am 13.09.2014



F. Verwendete Software



Abbildung F.5.: Infodialog des Programs GitHub



Abbildung F.6.: Infodialog des Programs Firefox

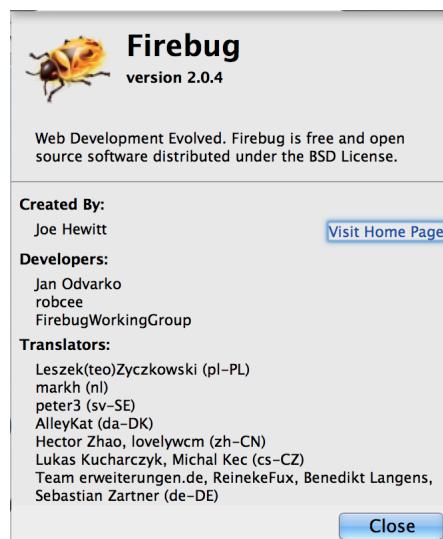


Abbildung F.7.: Infodialog der Firefox-Erweiterung Firebug



F. Verwendete Software

PHP Version 5.5.16



System	SunOS localhost 5.10 Generic_142901-13 i86pc
Build Date	Aug 22 2014 13:22:48

Abbildung F.8.: Verwendete PHP Version