

Seminararbeit

# Professionelle Textsatzsysteme

Bachelor Elektro- und Informationstechnik

Erstellen einer Rechnungs-Webapplikation  
mit Hilfe von Latex

Eine Arbeit für das StartUp Kiwilabs

Betreuer Prof. Dr. Paul Spannaus

Wintersemester 2017/2018



Technische Hochschule  
Ingolstadt



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>4</b>
<b>2</b>	<b>Theoretische Grundlagen</b>	<b>5</b>
2.1	HTML . . . . .	5
2.2	CSS . . . . .	5
2.3	JavaScript . . . . .	8
<b>3</b>	<b>Projektdurchführung</b>	<b>10</b>
3.1	Erstellen der Latex Vorlage . . . . .	10
3.2	Erstellen der Webseite . . . . .	11
3.2.1	Gestaltung des Frontends . . . . .	11
3.2.2	Logik des Frontends . . . . .	17
<b>4</b>	<b>Fazit und Ausblick</b>	<b>19</b>
	<b>Literaturverzeichnis</b>	<b>20</b>
4.1	Bücher . . . . .	20
4.2	Internetquellen . . . . .	20

# Abbildungsverzeichnis

2.1	Aufbau eines HTML-Dokuments [9]	6
2.2	Das Box-Modell [10]	8
3.1	Die fertige Kiwilabs-Rechnungsvorlage	10
3.2	Vollständiges Layout der Website	12
3.3	Angaben zum Rechnungssteller	13
3.4	Mobile Ansicht	14
3.5	Desktop Ansicht	14
3.6	Fertige Ansicht der Rechnungspositionen	16

# 1 Einleitung

Das StartUp Kiwilabs, dessen Gründer Sascha Kirstein und Michael Oldenburger sind, benötigt ein Tool zur professionellen Erstellung von Rechnungen und Angeboten. Das Unternehmen möchte in Ingolstadt die lokale StartUp Szene fördern, indem es bestehende StartUps unterstützt oder Ideen zu neuen StartUps formt.

Die Anforderungen für das Rechnungstool umfassen dabei

- ein professionelles Angebots- & Rechnungslayout,
- eine einfache Verwendbarkeit,
- eine Pflege gestellter Angebote und Rechnungen,
- eine Verwaltung von Firmen mit Kontaktpersonen,
- eine Ausfüllhilfe bestehender Firmenkontakte,
- eine eigenständigen Berechnung der Positionen,
- eine Benutzer-Authentifizierung

ohne einer Notwendigkeit der Anwendungsinstallation bei den Benutzern.

Latex bietet für viele der genannten Anforderungen gute Lösungen und eignet sich daher hervorragend als Basis dieses Projekts. Als Benutzerschnittstelle bietet sich eine Webseite an, in der ein autorisierter Benutzer Angebote und Rechnungen einfach erstellen kann. Die dabei erzeugten Daten werden dem Webserver übermittelt, der im Hintergrund die Latex Vorlage derart verändert, dass die gewünschte Rechnung nach dem Kompilieren entsteht und übergibt das fertige Portable Document Format (PDF)-Dokument zurück an den Ersteller. Für die Ausfüllhilfe im Frontend liefert der Webserver alle benötigten Daten.

Im Umfang dieser Arbeit wurde die Latex-Vorlage erstellt, welche die Grundlage für alle Angebote und Rechnungen bietet. Desweiteren wurde das Frontend mit allen Funktionalitäten implementiert. Die Daten der Ausfüllhilfen werden momentan als Mock-Up realisiert, in einer Form, wie sie ein späterer Server ausliefern würde. Die gesamten Angebot- bzw. Rechnungsdaten werden beim Absenden im Frontend als ein Datenobjekt per Hypertext Transfer Protocol (HTTP)-GET versendet und sind damit für einen Server empfangbar.

Für Projektinteressierte ist das Ergebnis der Arbeit auf <https://github.com/chh1399/RechnungsAppKiwilabs.git> aufrufbar.

## 2 Theoretische Grundlagen

### 2.1 HTML

Um das Verständnis für die Projekteinführung zu erlangen und die im Kapitel 3 beschriebene Durchführung nachvollziehen zu können, wird in diesem Kapitel zuerst der literarische Bogen von Hypertext Markup Language (HTML) über Cascading Style Sheet (CSS), JavaScript und abschließend zu AngularJS gespannt. HTML ist, ähnlich wie Latex, eine textbasierte Skriptsprache zur Beschreibung von Websites. Diese werden von Webbrowsern, wie z.B. Google Chrome, zur Darstellung interpretiert.

#### Aufbau einer HTML Datei

```
1 <!DOCTYPE html>
2 <html lang="de">
3 <head>
4     <title>Kiwilabs Rechnungs-WebApp</title>
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6
7 </head>
8 <body>
9     <h1> Dies ist eine Überschrift </h1>
10 </body>
```

Generell kann das HTML Dokument in den sog. *Head* und *Body* unterteilt werden. Solche Abschnitte werden mit einem entsprechenden Tag geöffnet, bzw. geschlossen. Ein Tag ist eine Anweisung in einer spitzen Klammer und liefert dem Browser Informationen über die Struktur des zu darstellenden Dokuments. Der Aufrufer der WebSite sieht nur die Daten, die im Body-Tag beschrieben sind. [5]. Ein *<p>-Tag* erzeugt einen Absatz mit Zeilenumbruch, ein *<h1, h2, h3, etc.>* Tag erzeugt eine Überschrift, wie hier der h1 Tag `<h1> Dies ist eine Überschrift </h1>`, Z.9. Der sogenannte *<div>-Tag* wird vor allem benutzt, um mehrere HTML-Blöcke zu gruppieren und anschließend mit Hilfe von CSS zu formatieren.

### 2.2 CSS

#### Die Definition und Bedeutung von CSS

Die Seitenbeschreibungssprache HTML dient zusammenfassend dem grundsätzlichen Aufbau der Website und definiert somit den Content. Standardmäßig werden allerdings die einzelnen HTML Elemente nur untereinander aufgereiht und besitzen nur wenige, von

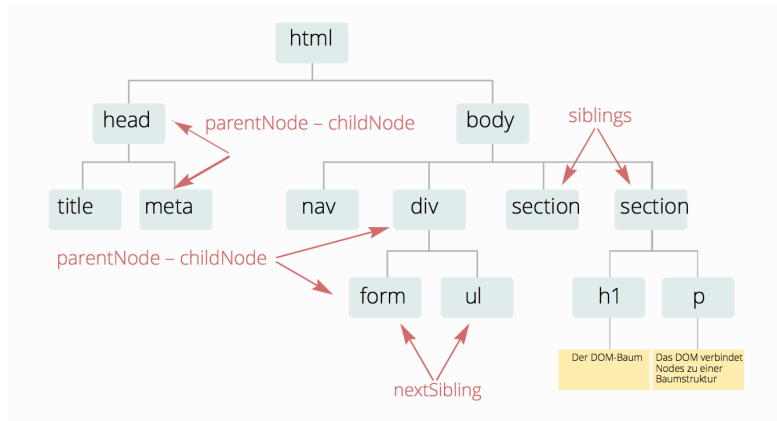


Abbildung 2.1: Aufbau eines HTML-Dokuments [9]

dem Browser vorgegebene Formatierungen. Hier schafft CSS Abhilfe, auf das in diesem Abschnitt eingegangen wird. So ist es eine Formatierungs- und Gestaltungssprache, mit der man fähig ist, die zuvor mittels HTML erzeugten Elemente gezielt zu platzieren, deren Farbe und Form zu verändern und die aufzubauende Website folglich nach eigenem persönlichem Geschmack zu formatieren. Die Philosophie hierfür ist eine Trennung des Aufbaus der Website in Inhalt (HTML) und Gestaltung (CSS) [11].

### Der Aufbau eines Stylesheets - Verwendung der CSS-Regeln

Der Aufbau des Stylesheets erfolgt durch die Festlegung der CSS Regeln. Diese definieren jeweils die Formatierung eines HTML Elements. So erfolgt der Aufbau durch eine Voranstellung eines Selektors und eines darauf folgenden Deklarators. Ersteres wählt das gewünschte, zu formatierende HTML Element aus, zum Beispiel durch die Verwendung eines `<p>` Tags. Davon gibt zweiteres die Art und Weise an. Diese wird von geschweiften Klammern umgeben und enthält, von einem Doppelpunkt getrennt, zuerst die Eigenschaft und dann den Wert [4].

Externe Stylesheets	Definition der Regeln in externer Formatvorlage
Inline Styles	Direktes Festlegen der Regeln in öffnendem Tag der Datei
Styles im Head Bereich	Definition der Regeln im Kopf Bereich der HTML Datei

Tabelle 2.1: Integrationsmöglichkeiten von CSS in HTML

## Die diversen Arten der Selektoren - Tags, IDs und Klassen

Im Folgenden wird die mögliche Verwendung von Selektoren beschrieben. Diese können entweder Tags, IDs oder Klassen sein. *Tag-Selektoren* wählen, wie der Name sagt, alle Tags einer Art aus. Der *ID-Selektor* wird verwendet, um einzelnen HTML Elementen Eigenschaften zu zuweisen. Dieser wird durch das Hashtag Symbol begonnen und darauf folgend ist die *HTML-ID*. Die definierten Eigenschaften gelten nun nur für das spezifische Element mit der identischen ID. Wenn HTML Elemente, die mehrere, bzw. unterschiedliche Tags besitzen, nach gleichem Style formatiert werden sollen, muss auf *Klassen-Selektoren* zurück gegriffen. Diese werden mit einem anfänglichem Punkt und dem Klassennamen definiert.

## Die Verwendung von externen Stylesheets

Es gibt verschiedene Integrationsmöglichkeiten von *CSS* in die *HTML* Datei. Im Zuge der Arbeit wird allerdings nur auf die in dem Projekt angewandte Möglichkeit der Benutzung von externen Stylesheets eingegangen. Hier wird die Formatierung und Gestaltung in einer externen Datei ausgelagert. Die direkte Verknüpfung erfolgt dann mit einem `<link>` Tag. Dieses muss immer im Head der HTML Datei stehen und verlangt zwei Attribute. Zum einem *rel*, was die Beziehung des verlinkten Dokuments zur HTML Datei angibt. Zum Anderen *href*, das die URL zum verlinkten Dokument mitgibt. Der Vorteil der Verwendung von einem externen Stylesheet ist die erhöhte Flexibilität des Websitenstyles und die gesteigerte Effizienz, da diese gestaltungsgebende Datei nach Bedarf leicht ausgetauscht und zudem für mehrere HTML-Dateien verwendet werden kann [6].

9 `<link rel="stylesheet" href="w3.css">`

## Die Gestaltung des Seitenlayouts

In CSS können HTML-Elemente grundsätzlich in zwei Arten unterteilt werden: Block- und Inline Elemente. Erstere sind klar abgegrenzte Bereiche, da sie vor und hinter sich einen Zeilenumbruch bewirken und als rechteckige Boxen dargestellt werden. Deren Breite, Höhe oder Abstand kann man beliebig festlegen. Die Darstellung von dieser Elementart und Zusammenfassung in einem Layout erfolgt über das Box-Modell.(vgl.Abb.2.2) Mit diesem kann das Layout in 4 Bereiche differenziert werden und hilft so zu einem übersichtlichen Seitenlayout Entwurf. Inline Elemente hingegen erzeugen keinen Zeilenumbruch, somit kann man zum Beispiel einzelne Wörter als solche festlegen. Die Maße sind folglich nicht individuell setzbar, sondern automatisch so lang wie das Wort.

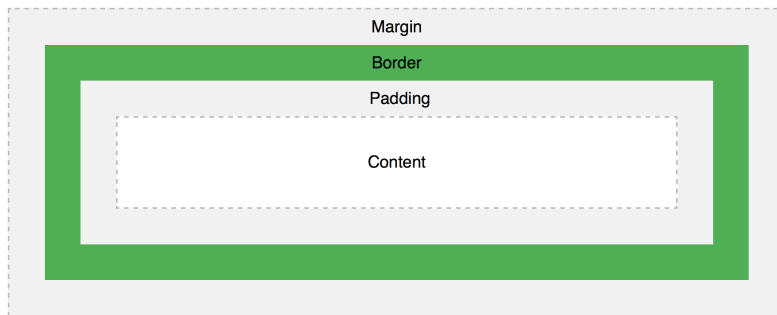


Abbildung 2.2: Das Box-Modell [10]

## 2.3 JavaScript

### Die Intention der Verwendung von Javascript

Für eine Webseitenprogrammierung wird neben dem Wissen über CSS und HTML auch ein Repertoire an verfügbarem Javascript Know-How benötigt. JavaScript wird in eine HTML Datei eingebunden und man ist mit Hilfe dieses Werkzeug in der Lage, beliebigen Text in einer schon dargestellten Website zu ändern. Dies ist beispielsweise bei einer immer aktuellen Anzeige der Uhrzeit auf der Website notwendig. Außerdem können dynamische Charakteristika, wie etwa das Öffnen oder Schließen von Optionen/Dateien, durch Javascript gesteuert werden [8].

### Das Einbinden in das HTML-Dokument

Implementiert wird Javascript mit dem Kommando `<Script>` und beendet mit `</script>`, in diesem Bereich können Aufforderungen über die Dynamik der Website definiert werden. Dies kann durch einen direkten Aufruf oder über eine Einbindung einer externen Script Datei geschehen [8].



## AngularJS im Rahmen der Anwendung von Javascript

Eine Option für ein JavaScript ist Angular JS. Es ist OpenSource von Google entwickelt worden. Es vereinfacht unter Anderem den Datenaustausch zwischen Frontend und Logik. Deswegen wird im folgenden auf dieses Javascript Webframework eingegangen, da es eine wichtige Rolle für die Umsetzung des Projektes spielt. Mit AngularJs werden Direktiven, sogenannte *ng-directives* in die HTML-Datei eingebunden. Dadurch können Eingaben des Webseitenbesuchers übernommen werden. Die Direktive *ng-model* verbindet eine Variable mit dem Frontend. Diese kann eine Eingabe des Nutzers entgegen nehmen oder auf der Website ausgegeben werden. Diese zwei Direktiven sind die grundlegendsten Arten. Auf die in dem Projekt tatsächlich verwendeten Direktiven wird später eingegangen. An dieser Stelle ist die Unterscheidung der AngularJS Applikationen in die sogenannten *modules* und *controllers* wichtig. Ersteres definiert die AngularJS Applikation [7]. Beispielcode:

```
1 var applikation = angular.module('myapplikation', []);
```


Der *controller* steuert einen Teil der Applikation. In folgendem Beispielcode besitzt er den Namen *myController*. AngularJS ruft den Controller mit einem *scope* Objekt auf. In diesem legt der *controller* zwei Variablen an: *firstName* und *lastName*.

```
1 app.controller('myCtrl', function($scope) {  
2     $scope.firstName = "Max";  
3     $scope.lastName = "Muster";  
4 });
```

Dies ist das grundlegende Wissen, das für die Durchführung des Projektes benötigt wird.

## 3 Projektdurchführung

### 3.1 Erstellen der Latex Vorlage



KiwiLabs UG, Kiwilabsstr. 1, 85049 Ingolstadt

Muster Unternehmen  
Berta Beispiel  
Musteradresse 42  
85049 Ingolstadt

Anschrift KiwiLabs UG  
Kiwilabsstr. 1  
85049 Ingolstadt

Ansprechpartner Sascha Kirstein  
Mobil +49 0123 45689  
E-Mail Regie@KiwiLabs.de  
Webseite www.KiwiLabs.de  
Steuernummer DE313106876

Institut Musterbank  
IBAN DE28 1010 0000 6500 1111 23  
BIC MUDEB23XXX

Rechnungsnummer:  
R-20171130-1

Ingolstadt, 30.11.2017

**Rechnung**

Sehr geehrte Damen und Herren,

Für die von KiwiLabs erbrachte Leistung erhalten Sie hiermit Ihre Rechnung.

Pos.	Bezeichnung	Rabatt	USt.	Menge	Einzel	Gesamt
1.	Musterposition (1h) November 2017		19%	500	100 €	50000 €
Summe:						50000 €
Netto:						50000 €
Umsatzsteuer 19%:						9500 €
<b>Brutto:</b>						<b>59500 €</b>

Ich bedanke mich sehr herzlich für Ihren Auftrag. Bitte zahlen Sie den unten aufgeführten Gesamtbetrag unter Angabe der Rechnungsnummer (R-20171130-1) innerhalb der nächsten 30 Tage auf das angegebene Bankkonto ein.

Mit freundlichen Grüßen,

Sascha Kirstein  
(KiwiLabs UG)

Abbildung 3.1: Die fertige Kiwilabs-Rechnungsvorlage

Im ersten Schritt wird das Latex Dokument erstellt. Die Grundlage bildet dabei die Briefklasse *scrlltr2* aus dem KOMA-Script. Bestimmte Stellen müssen am Ende aus der Java Anwendung verändert werden. Diese Textstellen werden zur verbesserten Strukturierung nicht in der zentralen Latex-Datei, sondern in der *Data.tex* Datei zur Trennung des festen Layouts und der veränderlichen Dateien ausgelagert. Dafür wird die Funktion `\newcommand` verwendet, um die Textbereiche in der Vorlage durch Werte in der *Data.tex* zu ersetzen. In diesem Beispiel `\newcommand{\senderCompany}{KiwiLabs UG}` wird in *Data.tex* das Kommando `\senderCompany` definiert. In der Vorlage wird anstelle des festen Werts „KiwiLabs UG., der neue Befehl aufgerufen. Soll durch das Java Programm der Wert verändert werden, reicht eine Anpassung von *Data.tex* um alle Vorkommen in der Vorlage auszutauschen.

## 3.2 Erstellen der Webseite

### 3.2.1 Gestaltung des Frontends

Ausgehend von der Latex Vorlage sind folgende Elemente auf der Website notwendig:

- Rechnungssteller mit Eingabefeld
- Rechnungsempfänger mit Eingabemöglichkeiten für
  - den Firmennamen
  - den Kundennamen
  - und der Adresse
- Rechnungspositionen mit
  - Bezeichnung
  - Menge
  - Einzelpreis
  - Rabatt
  - Steuern
- Button für die Erstellung des Latex-Dokuments

Das fertige Layout wird Schritt für Schritt in den nächsten Absätzen der Arbeit erklärt und verdeutlicht, wie der Quellcode implementiert worden ist. An dieser Stelle wird für einen Überblick das Resultat gezeigt.

Kiwilabs Rechnungs-WebApp

Angaben zum Rechnungssteller  
Name

Angaben zum Rechnungsempfänger  
Firmenname  
Kundenname  
Straße  
Postleitzahl  
Stadt

Rechnungspositionen  
Bezeichnung  
Menge  
Einzelpreis  
Rabatt (%)  
USt. (%)

19% (Normalsatz) 7% (Ermäßigter Satz)

Pos.	Bezeichnung	Rabatt	USt.	Menge	Einzelpreis	Gesamtpreis
Rechnung erstellen						

Abbildung 3.2: Vollständiges Layout der Website

Ausgehend von den HTML-Kenntnissen (siehe Seite 5) wird der *Head* des Dokuments angelegt.

```

1 <!DOCTYPE html>
2 <html lang="de">
3 <head>
4   <meta charset="UTF-8">
5   <title>Kiwilabs Rechnungs-WebApp</title>
6
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8
9   <link rel="stylesheet" href="w3.css">
10
11   <script src="angular.min.js"></script>
12
13 </head>
14 <body ng-app="invoiceApp" ng-controller="myCtrl">

```

In Zeile 9 wird ein externes Stylesheet [12] eingebunden. Diese Vorlage ist lizenzfrei verfügbar von w3. Der *Body* des HTML-Dokuments wird in Zeile 14 begonnen. 3.2.1

## Angaben zum Rechnungssteller

```
22 <div class="w3-card">
23
24   <div class="w3-container w3-light-grey w3-section">
25     <h3>Angaben zum Rechnungssteller</h3>
26     <label class="w3-text-black"><b>Name</b></label>
27     <input ng-click="toggleSenderList()" ng-model="senderName" class=
      "w3-input w3-border" id="senderName" name="senderName" type="
      text"></p>
28
29     <div ng-show="showSenderList">
30       <ul class="w3-ul w3-hoverable">
31         <li ng-repeat="x in myData" ng-click="chooseSender(x)">
32           {{ x.FirstName + ' ' + x.LastName }}
33         </li>
34       </ul>
35     </div>
36
37   </div>
38 </div>
```

In Zeile 22 wird ein neuer *div*-Tag mit der Klasse *w3-card* eröffnet. Dies ist eine Box mit Schatteneffekt. In Zeile 24 wird ein neuer *div*-Tag geöffnet, der ein Container mit einem Padding von 16 Pixel mit der Hintergrundfarbe von hellgrau darstellt. Dieses Element hat die schwarze Überschrift *Angaben zum Rechnungssteller*.

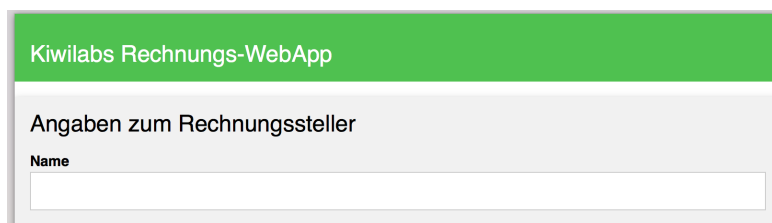


Abbildung 3.3: Angaben zum Rechnungssteller

Ab Zeile 27 wird der Input beschrieben. Die darauf folgenden Attribute *ng-click* und *ng-model* sind Elemente von AngularJS, die die Reaktion der Eingabefelder gewährleisten. Dieser Punkt wird später näher beschrieben. *w3-input* ist die Definition für einen Input und *w3-border* sorgt für eine Umrandung des HTML-Elements. Zur korrekten Vervollständigung wurde außerdem eine ID, ein Name und ein Typ angegeben. Der nächste Bereich, eingeleitet durch den Aufruf *div* in Zeile 29, dient zur Implementierung des PopUp Effekts für eine Schnellauswahl der Rechnungssteller. Die Elemente dieser werden durch eine ungeordnete Liste, abgekürzt durch *ul*, dargestellt. *w3-hoverable* sorgt für die dynamische Hervorhebung des Namens, wenn die Maus des Nutzers über die Stelle fährt.

## Angaben zum Rechnungsempfänger

Die Layouterstellung des Abschnittes über die *Angaben zum Rechnungsempfänger* sind analog zu *Angaben zum Rechnungssteller*, weswegen nicht näher darauf eingegangen wird.

## Darstellung der Rechnungspositionen

Der Abschnitt zu den Rechnungspositionen wird in Zeile 88 begonnen. In Zeile 98 beginnt eine *w3-row*. Diese sorgt für eine Reihung der Elemente *Menge*, *Einzelpreis* und *Rabatt* nebeneinander. Für den Anwender soll das Layout nicht nur in der Ansicht eines Laptops oder PC, sondern auch in einer mobilen Version ansehnlich sein. Dies wird in Zeile 101 mittels der Klasse *w3-col* gewährleistet. *m3* definiert die Darstellung für alle Medium Geräte, diese sind bei *w3school* größer als 601 Pixel. In dieser Darstellung nimmt es 3/12 der Bildschirmbreite ein. Ein Endgerät mit weniger als 601 Pixel Bildschirmbreite wird von *w3.css* als *small* bezeichnet. Dieses würde in diesem Fall das Element auf der gesamten Bildschirmbreite angezeigt bekommen. In Zeile 106 wird der Abstand zwischen den Feldern *Menge* und *Einzelpreis* bei großen Bildschirmen eingefügt. Der Aufruf *w3-hide-small* entfernt den Abstand in der Ansicht von kleinen Bildschirmen.

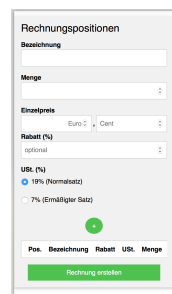


Abbildung 3.4: Mobile Ansicht

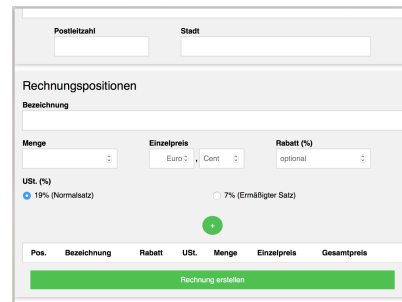


Abbildung 3.5: Desktop Ansicht

```
88 <div class="w3-card">
89
90   <div class="w3-container w3-light-grey w3-section">
91
92     <div>
93       <h3>Rechnungspositionen</h3>
94       <label class="w3-text-black"><b>Bezeichnung</b></label>
95       <input ng-model="positionDescription" class="w3-input w3-
          border" id="positionDescription" name="positionDescription
            " type="text"></p>
96     </div>
97
98     <div class="w3-row">
99
100
101       <div class="w3-col m3">
102         <label class="w3-text-black"><b>Menge</b></label>
```

```

103         <input ng-model="amount" class="w3-input w3-border" id="
           amount" name="amount" type="number"></p>
104     </div>
105
106     <div class="w3-col m1 w3-hide-small">
107         <p></p>
108     </div>

```

Das Feld für den Einzelpreis wird ab Zeile 111 definiert, welches in Euro und Cent dargestellt werden soll, um den individuellen Kostenbetrag darstellen und eine weitere Verarbeitung des Betrages erreichen zu können. Wenn der Kostenbetrag mit Kommatarennung des Euro- und Centanteils in ein Feld geschrieben werden würde, müsste dieser mit einer Parse-Funktion eventuell später überarbeitet werden. Dieser Schritt wird durch eine explizite Trennung in zwei Eingabefelder eingespart. Diese beiden Felder werden jeweils durch eine *w3-cell* definiert, diese ist ein einfaches und horizontal orientiertes Block-Element. Auf die Eingabefunktion und Übernahme des Betrages zu einem späteren Zeitpunkt eingegangen.

```

111     <div class="w3-col m3">
112         <label class="w3-text-black"><b>Einzelpreis</b></label>
113         <div class="w3-cell">
114             <input ng-model="unitCostEUR" class="w3-input w3-
               border w3-right-align" id="unitCostEUR" name="
               unitCostEUR" type="number" placeholder="Euro">
115         </div>
116         <div class="w3-cell w3-center" style="width: 10px">
117             <b>,</b>
118         </div>
119         <div class="w3-cell">
120             <input ng-model="unitCostCENT" class="w3-input w3-
               border" id="unitCostCENT" name="unitCostCENT" type
               ="number" placeholder="Cent">
121         </div>
122
123
124     </div>
125
126
127     <div class="w3-col m1 w3-hide-small">
128         <p></p>
129     </div>

```

Das nächste Feld der Rechnungspositionen ist der Rabatt, dies wird in der Zeile 132 begonnen und in Zeile 137 mit `</div>` beendet.

```

132     <div class="w3-col m3">
133         <label class="w3-text-black"><b>Rabatt (%)</b></label>
134         <input ng-model="discount" class="w3-input w3-border" id=
           "discount" name="discount" type="number" placeholder="
           optional"></p>
135     </div>
136
137 </div>

```

Das letzte Element für die Rechnungspositionen ist die zu addierende Umsatz- und Mehrwertsteuer, die in der Preisberechnung berücksichtigt werden muss. Hier wird wieder eine *w3-row* eröffnet, die mit USt. und 7 Prozent Ermäßigter Satz beschriftet wird. Dazwischen befinden sich Elemente, die durch AngularJs interaktiv auf den Nutzer der Website reagieren.

```

141     <div class="w3-row">
142
143         <div> <label class="w3-text-black"><b>USt. (%)</b></label></div>
144
145         <div class="w3-col m6">
146             <input ng-model="valueAddedTax" class="w3-radio" type="
147                 radio" id="valueAddedTax19" name="valueAddedTax" value=
148                 ="19" checked><label>19% (Normalsatz)</label></p>
149
150         <div class="w3-col m6">
151             <input ng-model="valueAddedTax" class="w3-radio" type="
152                 radio" id="valueAddedTax" name="valueAddedTax7" value=
153                 ="7"><label>7% (Ermäßigter Satz)</label></p>
154
155     </div>

```

Um weitere Positionen hinzufügen zu können, ist eine Add-Funktion gefordert, die benutzerfreundlich durch einen Plus-Button repräsentiert wird. Als letzter Punkt soll in der Ansicht der Website in einer Zeile alle Punkte der Rechnung übersichtlich dargestellt sein, was nun definiert wird. In Zeile 154 wird hierzu das Element in dem Zentrum des Bildschirms angeordnet. Innerhalb des *ng-click* von AngularJS wird das Feld als runder Button mit grüner Füllung und automatisch berechnetem Abstand.

```

155     <div class="w3-center">
156         <button ng-click="addPosition()" class="w3-button w3-circle
157             w3-green w3-margin">+</button>
158     </div>

```

Abbildung 3.6: Fertige Ansicht der Rechnungspositionen

Nun sind die benötigten Punkte der Rechnungspositionen komplettiert und zur Veranschaulichung der gesamten Rechnung werden am Ende des Webseitenlayouts alle Teile in einer Reihe aufgeführt.

Für diese Anforderung wird eine Tabelle eröffnet, die sich dynamisch an die jeweilige Bildschirmgröße anpasst. (*w3-responsive*, Z. 158f. Die einzelnen Punkte werden in der



Tabelle durch den `<t>` Tag in Zeile 160 und die jeweiligen `<th>` Tags als einzelne Elemente in einer neuen Zeile implementiert.

```

158     <div class="w3-responsive">
159         <table class="w3-table-all ">
160             <tr>
161                 <th>Pos.</th>
162                 <th>Bezeichnung</th>
163                 <th>Rabatt</th>
164                 <th>USt.</th>
165                 <th>Menge</th>
166                 <th>Einzelpreis</th>
167                 <th>Gesamtpreis</th>
168                 <th></th>
169             </tr>
170             <tr ng-repeat="pos in invoicePositions">
171                 <td>{{$index}}</td>
172                 <td>{{pos.positionDescription}}</td>
173                 <td>{{pos.discount + "%"}}</td>
174                 <td>{{pos.valueAddedTax + "%"}}</td>
175                 <td>{{pos.amount}}</td>
176                 <td>{{pos.unitCost.toFixed(2)}}</td>
177                 <td>{{pos.totalPrice.toFixed(2)}}</td>
178                 <td> <button ng-click="deletePosition($index)" class=
                    "w3-button w3-red w3-small w3-padding-small">-</
                    button></td>
179             </tr>
180         </table>
181     </div>
182

```

## Button für die Erstellung des Latex-Dokuments

Der Button für die Erstellung des Latex-Dokuments wird zentriert auf dem Display des Nutzers (vgl. Z.184 ) platziert. Außerdem ist der *w3-button* ein grüner Block, weswegen er die gesamte Breite des Bildschirms benutzt und die Aufschrift „*Rechnung erstellen*“ trägt. Die Erstellung in ein pdf wird über AngularJs in *Logik des Frontends* gewährleistet.

### 3.2.2 Logik des Frontends

#### Logik für Rechnungssteller und -empfänger

Die Logik des Frontends ist ein substantieller Teil der Webseitencharakteristik. Durch diese wird die Reaktion der Website auf Benutzereingaben oder Veränderungen der Fenstergröße garantiert. AngularJS wird das erste Mal in dem Projekt benötigt, wenn bei der Auswahl des Rechnungsstellers zwischen zuvor hinterlegten Personen gewählt werden soll, die als neue Option auf dem Desktop erscheinen sollen. Eine solche Interaktion wird in Zeile 27 durch den *input*-Aufruf *ng-click* und *ng-model* implementiert. Die erste Direktive teilt dem Java-Framework AngularJs mit, was bei einem Klick des Nutzers geschehen soll. Dies ist in diesem Fall *toggleSenderList()*, diese Funktion wird in *invoiceApp.js*

definiert. Die Direktive *ng-model* verbindet das Eingabefeld mit der in *invoiceApp.js* definierten Variable *senderName*. *ng-show* ermöglicht die Darstellung der Sender Liste. Die Namen werden über die REST Schnittstelle im JSON Format übergeben und dienen der Vorauswahl. Wenn neue Namen eingegeben werden, die noch nicht existent sind, werden diese an den Server übergeben und hinterlegt. Die Logik für die Auswahl der Rechnungsempfänger (vgl. 48ff) ist entsprechend diesem Prinzip.

## PDF-Erstellung

```
183     <div class="w3-center w3-margin">
184         <button ng-click="downloadPDF()" class="w3-button w3-block w3-
           -green">Rechnung erstellen</button>
185     </div>
```

In Zeile 185 des html-Dokuments wird mittels *ng-click* mitgeteilt, dass bei einem Klick auf den Button *downloadPDF* die gleichnamige, in *invoiceApp.js* definierte Funktion ausgeführt werden soll. Diese legt das Objekt *data* neu an und befüllt deren Variablen mit denen aus der Website. Im nächsten Schritt wird *data* an den *localhost* gepostet. Falls dies nicht funktioniert, wird die Nachricht "Fehler" gemeldet.

```
79     $scope.downloadPDF = function() {
80
81         var data = new Object();
82         data.senderName = $scope.senderName;
83         data.customerCompany = $scope.customerCompany;
84         data.customerName = $scope.customerName;
85         data.customerStreet = $scope.customerStreet;
86         data.customerZIP = $scope.customerZIP;
87         data.customerCity = $scope.customerCity;
88         data.invoicePositions = $scope.invoicePositions;
89
90
91         $http.post("localhost", data).then(
92             function(response){
93                 alert(response);
94             },
95             function(response){
96                 alert("Fehler");
97             }
98         );
99     }
```

## 4 Fazit und Ausblick

Dieses Layout wurde mittels HTML, CSS und dem JavaScript Framework Angular JS umgesetzt. Die gestellten Anforderungen werden bis auf die Pflege gestellter Angebote und Rechnungen, sowie eine Verwaltung von Firmen mit Kontaktpersonen erfüllt. Insgesamt ist das Frontend fertig. In den nächsten Schritten können der, bis jetzt fehlende, Webserver und eine Datenbank, die mithilfe von SQL verwirklicht werden kann, aufgesetzt werden. Auch die Java-Applikation für eine Datenannahme der Kundeninformationen und die anschließende Verarbeitung zur Ausgabe einer pdf-Datei ist angedacht, dessen Grundfunktion bereits implementiert ist. Nach dieser Umsetzung werden schlussendlich alle gestellten Anforderungen erfüllt.

# Literaturverzeichnis

## 4.1 Bücher

- [1] *Einstieg in JavaScript: Dynamische Webseiten erstellen. Inkl. Zusammenspiel von HTML, CSS, Ajax, jQuery, jQuery mobile u.v.m.* Rheinwerk Computing
- [2] *Grundlagen und Praxiswissen: Von Animationen bis Responsive Webdesign.* Rheinwerk Verlag
- [3] *JavaScript. Einführung - Programmierung - Referenz.* dpunkt Verlag

## 4.2 Internetquellen

- [4] DIGITALAGENTUR KICHBERGER KNORR: *css-lernen*. <http://www.css-lernen.net>
- [5] HU-BERLIN: *Grundgerüst einer HTML Datei*. <https://www2.informatik.hu-berlin.de/Themen/www/selfhtml/html/allgemein/grundgeruest.htm>. Version: 2001
- [6] RALPH G. SCHULZ: *CSS-Artikel: Kaskade*. [http://www.css-cafe.de/art\\_kaskade.php](http://www.css-cafe.de/art_kaskade.php)
- [7] SASCHA BRINK: *Angular JS Tutorial für Einsteiger*. <https://angularjs.de/artikel/angularjs-tutorial-deutsch/>. Version: 2014
- [8] U. HÄSSLER: *Javascript Basis*. <https://www.mediaevent.de/javascript/>. Version: 2016
- [9] U. HÄSSLER: *Javascript DOM (Document Object Model)*. <https://www.mediaevent.de/javascript/DOM.html>. Version: 2016
- [10] W3SCHOOLS: *The CSS Box Modell*. [https://www.w3schools.com/css/css\\_boxmodel.asp](https://www.w3schools.com/css/css_boxmodel.asp). Version: 2017
- [11] W3SCHOOLS: *CSS Tutorial*. <https://www.w3schools.com/css/default.asp>. Version: 2017
- [12] W3SCHOOLS: *W3.CSS Tutorial*. <https://www.w3schools.com/w3css/default.asp>. Version: 2017

# Abkürzungsverzeichnis

**PDF** Portable Document Format

**CSS** Cascading Style Sheet

**HTML** Hypertext Markup Language

**HTTP** Hypertext Transfer Protocol