

Bachelor-Thesis

zur Erlangung des akademischen Grades

Bachelor of Science (B. Sc.)

an der Hochschule für Technik und Wirtschaft des Saarlandes

im Studiengang Kommunikationsinformatik

der Fakultät für Ingenieurwissenschaften

Migration eines webbasierten Bestellungssystems in eine .Net-Umgebung mit Umbraco Content-Management Funktionalität

vorgelegt von

Bozhidar Aleksandrov

betreut und begutachtet von

Prof. Dr. Helmut Folz

Thomas Beckert, M.Sc.

Saarbrücken, 30. September 2018

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend“ zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Saarbrücken, 30. September 2018

Bozhidar Aleksandrov

Zusammenfassung

In dieser Arbeit wird eine webbasierte Software betrachtet, die auf ASP basiert ist. Weil sie zu alt ist, wird sie analysiert, bewertet und es wird ein neues Konzept entwickelt. Es geht um eine Migration von einer Software zu einem Content Management System. Unter mehrere solche wird es Umbraco entschieden. Es wird erklärt, warum Umbraco auf erste Linie der Entscheidung steht. Es wird erläutert, die Methodiken, wie das Konzept entwickelt wird, welche Werkzeuge benutzt werden und am Ende wird das Konzept mithilfe von einem Prototyp Programm angegeben.

Model-driven software development offers the method of choice when it comes to manage complex software production projects. However, these concepts face some obstacles when applied to maintenance of existing software systems. In order to ally such modern methods it is frequently assumed that re-coding cannot be circumvent

— Dr.-Ing. Hans-Georg Pagendarm [knuth:1974]

Danksagung

Zunächst möchte ich Prof. Dr. Helmut Folz für die kompetente Betreuung der Arbeit und die Begutachtung der Bachelorarbeit danken.

Herrn M. Sc. Thomas Beckert danke ich für das interessante Arbeitsthema, die Bereitschaft, diese Arbeit als Zweitgutachter zu lesen und zu bewerten.

Außerdem danke ich Herrn B. Sc. Matheo Zech für professionelle Korrektur und nützliche Hinweise, um dieser Arbeit besser zu werden.

Zu guter Letzt ein herzliches Dankeschön an meine Eltern, meinen Bruder und ebenso an meine Freunde, die für den nötigen Ausgleich gesorgt und mich auf diese Weise stets motiviert haben, nicht nur während der Bachelorarbeit sondern auch über die gesamte Studienzeit.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Vorstellung des Unternehmens	1
1.2	Motivation	1
1.3	Zielsetzung	2
1.4	Aufgabenstellung	2
2	Genutzte Technologie und Services	3
2.1	HTML	3
2.2	CSS	3
2.3	JavaScript	3
2.4	JQuery	3
2.5	AngularJS	3
2.6	Umbraco	4
2.7	1&1 Website Check	4
2.8	HTTP Obeservatory	5
3	IST-Analyse	7
3.1	Beschreibung des User Interface (UI) des Bestellsystems	7
3.1.1	Anmeldeformular	7
3.1.2	Registerformular	7
3.1.3	Auftraggeber - Verwaltung	7
3.1.4	Kundenansicht	8
3.2	Beschreibung der Funktionalität	8
3.2.1	Anmelde- und Registerformular	9
3.2.2	Auftraggeber-Verwaltung	9
3.3	Testen	14
3.4	Warum ist eine Migration notwendig?	17
4	Anforderungsanalyse	21
4.1	Paket A	21
4.2	Paket B	21
4.2.1	Kundenverwaltung	21
4.2.2	Artikelverwaltung	22
4.2.3	Auftragsverwaltung	22
4.2.4	E-Mail-Verwaltung	23
4.2.5	Umsatzverfassung	23
5	Konzeption und Implementierung	25
5.1	Aufbau vom Umbraco	25
5.2	Paket A	27
5.3	Kundenverwaltung	28
5.3.1	Kundenerfassung	28
5.3.2	Kundeansicht	29
5.3.3	Auftraggeber-Ansicht	31

5.3.4	Kommunikation	32
5.4	Artikelverwaltung	32
5.4.1	Erfassen, editieren und löschen	32
5.5	Auftragsverwaltung	33
5.5.1	Übersicht	33
5.5.2	Detailansicht	34
5.6	E-Mail Verwaltung	34
5.7	Umsatzverwaltung	34
Literatur		35
Abbildungsverzeichnis		37
Tabellenverzeichnis		37
Listings		37
Abkürzungsverzeichnis		39
A	Anhang -Kundenverwaltung	43
A.1	Kundenerfassung	43
A.2	Kundenansicht	47
A.3	Anhang - Auftraggeber-Ansicht	48

1 Einleitung

Seit mehreren Jahrzehnten zeichnet sich eine immer weiter voranschreitende Digitalisierung der Gesellschaft ab. Manche Wirtschaftsbetriebe und Einrichtungen haben sich schon früh damit arrangiert und entsprechende Produkte angeschafft. Dies umfasst fertig käufliche Hard- und Softwareprodukte, sowie individuell entwickelte Lösungen. Diese sind, wie die meisten digitalen Güter, einem schnellen Alterungsprozess unterworfen. Das können Schnittstellen sein, welche vom Hersteller nicht mehr unterstützt werden, oder eine neue Betriebssystem Version die nicht mehr Unterstützt wird. Wenn die Kunden direkt mit einer Software interagieren, ist Userexperience und Design ein nicht zu unterschätzender Faktor.

Die meisten Softwareinfrastrukturen wachsen meist mit einem Unternehmen und seinem Bedarf nach digitalen Lösungen. Das bedeutet, dass eine verworrene Struktur aus Abhängigkeiten entstehen kann. Das ist dabei eher die Regel als die Ausnahme.

Elektronische Datenverarbeitung (EDV)-Systeme haben nicht die klassischen Verschleißerscheinungen, so wie man sie von klassischen Betriebsmitteln kennt. Allerdings entsteht auch so ein Interesse nach einem gewissen Nutzungszeitraum die bestehende Software zu ersetzen. Man bezeichnet diesen Alterungsprozess, welchen man eingehend in der Softwarequalitätsforschung untersucht, als Softwarealterung. Software ist „weich“ und man sollte annehmen sie sei leicht änderbar und wartbar. Dies kann allerdings mit fortschreitendem Alter teurer sein als die Migration zu einem neuen System. Firmen die nicht mit der Zeit gehen werden schnell als alt und uninnovativ wahrgenommen. Dies kann sich schnell auf den Umsatz eines Unternehmens auswirken. Deshalb sind Migrationen gerade im Bereich des Web-Developments besonders häufig, bei denen man von Grund auf ein neues System erstellt.

Es gibt mehrere Fälle von Softwaremigrationen. In manchen Fällen kann Software Hardware ersetzen. In andere Fällen ersetzt wiederum Software Hardware. Wesentlich häufiger ist jedoch das alte Software durch neue ersetzt wird, sowie alte Hardware durch ihre neueren Pendants.

1.1 Vorstellung des Unternehmens

Der Aufgabestellung ist von der Firma SitePoint gegeben. das Unternehmen spezialisiert auf Content Management System, Mobile Web Applications und e-commerce. Die Mitarbeiter sind .Net Spezialisten, die Hochwertige Software auf Microsoft-Technologie setzen. Das ist das einzige Unternehmen im saarland, das Umbraco CMS verwendet. M. Sc. Thomas Beckert ist den Geschäftsführer der Firma und auch den Aufgäbegeber der gegenwärtige Arbeit. [8]

1.2 Motivation

In dieser Arbeit wird die Umstellung eines Bestellsystems eines Catering Services abgebildet werden. Dies bedeutet, das es sich bei dem Thema der Arbeit um eine Software zu Software Migration handelt. Genauer um eine modellgetriebene Softwaremigration.

1 Einleitung

Dabei wird das aus dem Beginn dieses Jahrtausend stammende Bestellsystem, welches auf einem Windows Imaging Components (WIC)-Plugin basiert und einem Active Server Pages (ASP)-Backend. Da die damals verwendeten Technologien vom Hersteller Microsoft seit geraumer Zeit End-of-Life gesetzt wurden, ist eine Migration zu einem aktuelleren Technologiestack zwingend erforderlich. Das Ziel des Migrationsprojekts ist es, das aktuelle Bestellsystem, welches immernoch auf ASP basiert, durch einen modernen Technologiestack zu ersetzen.

Dazu zählt die neue Konzeptionierung des Frontend. Dies soll zur Verbesserung und Erleichterung der Bedienbarkeit führen. Dazu wird eine neue Seite, welche auf Umbraco und ASP.NET basiert, erstellt.

1.3 Zielsetzung

Das betrachtete Projekt wird auf zwei Paketen aufgeteilt. Im Paket A wird angefordert, Möglichkeiten dem Auftraggeber zu vergeben, um er den Inhalt seiner Seite zu editieren und zu ändern. Im Paket B wird angefordert, dass Erstellung von Verwaltungen, durch die das Bestellsystem, Kommunikation zwischen dem Auftraggeber und seine Kunden sowie Buchhaltung und Auftragspeicherung realisiert werden können. Es wird auch eine Konzepte angefordert, durch die die Daten von der alten webbasierten Software zu die obengenannte Anforderungen übertragen werden müssen.

1.4 Aufgabenstellung

Im nachfolgenden Kapitel werden alle verwendeten Technologien erörtert und kurz erklärt. Danach wird im darauf folgenden Kapitel der technische und optische Stand der aktuellen Internetpräsenz analysiert.

Im darauf folgenden vierten Kapitel erfolgt die Erfassung und Anforderungsanalyse der Problemstellung deren theoretischen Lösungen. Im nachfolgenden Kapitel wird darauf aufgebaut und es erfolgt die praktische Umsetzung der Lösungssätze. Im letzten Kapitel wird die Qualität der Umsetzung erörtert und persönliche Designentscheidungen begründet. Danach folgt ein Ausblick auf weitere Verbesserungsmöglichkeiten.

2 Genutzte Technologie und Services

In diesem Kapitel beschäftigt man sich mit den verwendeten Technologien und Webservices. Diese sind in der Webentwicklung sehr entscheidend, da davon die Userexperience und Wartbarkeit abhängt. Besonderer Augenmerk wurde dabei auf die möglichst weite Verbreitung der verwendeten Technologien gelegt. Dies ist ein Vorteil, da daraus resultiert das es eine große Community gibt, die die Projekte aktuell hält.

2.1 HTML

Die Grundlegende Sprache für das erstellen und Rendern von Internetseiten ist Hypertext Markup Language (HTML). Es ist im Grunde DIE Schlüsseltechnologie um Internetseiten aus dem World Wide Web (WWW) anzuzeigen. Natürlich kann man HTML auch für das erstellen und webbasierten lokalen Graphical User Interface (GUI) genutzt werden. Weitere Informationen findet man unter [3].

2.2 CSS

Cascading Style Sheets (CSS) ist eine moderne Technologie zur Gestaltung von Internetseiten. Dies dient der Gestaltung von HTML-Texten und liefert auch teilweise dynamische Funktionalität. Weitere Informationen findet man unter [11].

2.3 JavaScript

JavaScript ist eine objektorientierte Interpretersprache, die Webseiten erst dynamisch macht. Sie wird in diesem Kontext im Webbrowser des Anwenders ausgeführt und stellt die Client-Seite einer Applikation dar. [4]

2.4 JQuery

JQuery ist ein weit verbreitetes JavaScript-Framework. Es stellt Funktionen zur Verfügung welche ein leichtes manipulieren des Seiteninhalts ermöglicht. So kann die Entwicklung von komplexen Webprojekten bedeutend beschleunigt werden.[2]

2.5 AngularJS

Hierbei handelt es sich um ein JavaScript clientseitiges Framework, welches hochdynamische WebAnwendungen oder lokale Anwendungen mit Webtechnologie ermöglicht. Mithilfe von Angular ist Möglich neue Architekturkonzepte auf den Client zu bringen und komplexe Anwendungen zu entwickeln. [5]

2.6 Umbraco

Umbraco ist ein Content Management System (CMS). Es dient zum Erstellen, Bearbeiten und zur Verwaltung dynamischer Webseiten. Umbraco basiert auf C# und auf der ASP.Net-Technologie. Heutzutage werden Microsoft SQL Server, My SQL, VistaDB, Peta Poco und weitere Datenbanken verwendet. Dieses CMS ist Open Source und die erste Version ist vom dänischen Software-Entwickler Niels Hartving im Jahr 2000 veröffentlicht worden.

Aus folgenden Gründen hat man sich bei der Umsetzung für Umbraco entschieden [9]:

- Umbraco ist ein flexibles CMS. Es gibt keine unnötigen Optionen und Schaltflächen. Alles ist einfach zu benutzen und zu verstehen.
- Der intuitive Editor ermöglicht es jede Art von Content einfach einzupflegen. Seiten sind einfach zu bearbeiten oder zu aktualisieren und wird nach dem selben eingängigen Paradigma dargestellt. Es ist ohne Bedeutung mit welchem Gerät auf die Webseite zugegriffen wird - Umbraco ist immer responsiv.
- Es gibt keine Einschränkung welche Webentwicklungsprogrammiersprache man nutzen muss. Umbraco ist sehr anpassungsfähig.
- Sehr gut angepasst für agile Prozesse: „Im Vergleich zu anderen Systemen geht der Livegang Deines Projekts damit sehr schnell. Umbraco unterstützt die agilen Prozesse der modernen Digitalbranche, bei denen es essenziell ist, dass Editoren immer und überall Content publizieren können, ohne damit "Content Freeze" zu verursachen. Gleichzeitig sollen auch Entwickler Bugfixes und Features schnell einbauen können. Umbraco sorgt dafür, dass der Flow nie endet“. [wieder da] [9]
- Umbraco CMS ist integrierbar. Man kann E-Commerce-Plattform, CRM (Custom Relationship Management) System oder 3rd-Party Personalisation Engine verwenden. Ohne Probleme können individuelle Systeme integriert werden. Wegen Application-Programm-Interface (API) werden alle Daten mit sichtbarem Content mit dem Umbraco Front- und Backend vernetzt.
- Die Community vom Umbraco ist groß. Freundliche und aktive Umbraco Nutzer helfen jeder Zeit gegenseitig bei der Verbesserung des Codes. Viele aktive Tester sorgen dafür, dass Umbraco ständig verbessert wird.
- Das Modell der Lastverteilung ist in ASP.NET integriert. Jede ASP.NET-Webseite besitzt eigene Session-Verwaltung, die so konfiguriert werden kann, dass sie die Daten auf den SQL Server verlegen kann. So lassen sich Daten in einem gemeinsamen Datenspeicher ablegen. So ist es möglich, dass jeder Server auf den Datenstand eines Nutzers zugreifen kann.
- Volle Versionskontrolle, volle Integration in vorhandene Strukturen, zeitgesteuertes Veröffentlichen, Workflow-Orientierte Seitenverwaltung, schnelle Seitenvorschau vor der Veröffentlichung, Mehrsprachigkeit, Papierkorb zum einfachen Wiederstellen von gelöschten Elementen und seine Lizenzkostenfreiheit sind weitere Eigenschaften, weshalb man sich für Umbraco als CMS entschieden hat.

2.7 1&1 Website Check

Diese Anwendung überprüft, wie gut die betrachtete Webseite ist und was noch optimiert werden kann. Vier Aspekten werden geprüft [1]:

- Darstellung der Webseite
- Auffindbarkeit in Suchmaschinen
- Darstellung der Webseite
- Sicherheit der Webseite
- Geschwindigkeit der Webseite

Wenn die Internetadresse eingegeben wurde, wird Webseite aufgerufen. Danach wird der Quellcode analysiert.

2.8 HTTP Observatory

Mozilla HTTP Observatory ist ein Set von Tools, die zur Analyse und als Informationsquelle für Verbesserungen der Website dienen.[6]

3 IST-Analyse

In dieser Arbeit wird das Bestellsystem von der Party-Service-Website "jungekueche" betrachtet. Dazu zählen auch das Verwaltungssystem des Auftraggebers und die Verwaltungsseite des Kunden. Zuerst wird das Userinterface aus der Sicht von allen drei Akteuren beschrieben. Danach wird die Funktion des Systems evaluiert.

3.1 Beschreibung des User Interface (UI) des Bestellsystems

3.1.1 Anmeldeformular

Die Anmeldung und die Registrierung für neue Nutzer befindet sich auf einer gemeinsamen Seite. Das Anmeldeformular hat zwei Felder, in denen der Benutzer seinen PIN und seine E-Mail eingeben muss. Neben den Feldern steht ein Hinweis, welcher den Kunden mit den nötigen Informationen versorgt, wie er sich anmelden oder registrieren kann.

Abbildung 3.1: Die bisherige Eingabemaske für das Kundenlogin

3.1.2 Registerformular

Falls der Kunde keine PIN hat, muss er sich registrieren. Dieses Formular besteht aus fünf Abschnitten. Im ersten muss der Kunde seine persönlichen Daten angeben. In dem zweiten steht die Informationen des Auftrags. Menü Auswahl ist der nächste Abschnitt. Im vorletzten wird die Zubehörauswahl und Serviceauswahl angezeigt. Der letzte Abschnitt ist die Erklärung zu dem Registrierungsprozess.

3.1.3 Auftraggeber - Verwaltung

Im Fall einer Registeranfrage kann der Auftraggeber sie bestätigen oder löschen. Dem Auftraggeber stehen drei Optionen zur Verfügung; Auftragsverwaltung, E-Mail Verwaltung und Umsatzverwaltung. Zuerst betrachtet man die Auftragsverwaltung. Dort befinden sich 11 Optionen, 9 davon leiten zu eigenen Unteroptionen weiter. Die anderen Zwei sind Speichern und Äbmelden". Durch diese Optionen kann der Auftraggeber die Kundedaten einsehen, editieren, bearbeiten und löschen. Außerdem kann er die dazugehörigen die Nachrichten lesen und löschen. Auch kann er neue und alte Aufträge einsehen und bearbeiten oder löschen.

3 IST-Analyse

Ist das Ihre erste Bestellung bei uns?
Bitte füllen Sie nachfolgendes Formular aus!

Bestellhilfe bitte hier klicken!

Name / Ansprechpartner:	<input type="text"/>	Vorname:	<input type="text"/>	Firma:	<input type="text"/>
Strasse:	<input type="text"/>	PLZ:	<input type="text"/>	Ort:	<input type="text"/>
		<small>(nur Zahlen erlaubt)</small>		<small>bitte auch Orts/Stadteil angeben.!!</small>	
E-Mail:	<input type="text"/>	<small>(wird auch ihr Login-Kundenname - bitte achten Sie darauf, Ihre E-Mail-Adresse richtig einzugeben.)</small>			
Telefon:	<input type="text"/>				
Handynummer:	<input type="text"/>	<small>unter der Sie auch am Veranstaltungstag zu erreichen sind</small>			

Abbildung 3.2: Die bisherige Eingabemaske für das Registrierungsformular

jungekueche.com **online** **büro** kundendatei auftragsübersicht neue kunden online-editor save Abmelden

neue Nachricht (2)	neue Aufträge (13)	bearbeitete Aufträge (0)	alte Aufträge (7979)	fällig (0)
------------------------------	------------------------------	------------------------------------	--------------------------------	----------------------

neue Nachricht	2
neue Aufträge	13
bearbeitete Aufträge	0
alte Aufträge	7979
fällig	0
neue Kundendaten	0

Abbildung 3.3: Die bisherige Eingabemaske für die Auftragsverwaltung

3.1.4 Kundenansicht

Wenn die Anfrage des Kunden bestätigt wird, bekommt der Kunde einen PIN pe E-Mail. Mit diesem PIN und seiner E-Mail kann der Kunde sich bei "Jungekueche" anmelden. Nach dem Login-Prozess, findet er sich auf einer persönlichen Nutzerseite wieder. Dort hat er eine Übersicht über seine aktuellen und alten Bestellungen, ein Kontaktfenster, in dem er die Nachrichten einsehen oder schreiben kann, sowie eine Art Newsletter. Außerdem gibt es eine Option, durch die der Kunde neue Bestellungen aufgeben kann.

3.2 Beschreibung der Funktionalität

Einer Anmeldung muss immer eine Registrierung vorausgehen. Diese Funktionalität wird mit ASP und ASP.NET realisiert. Die verwendeten Hilfsprogramme sind JQuery, JavaScript, AJAX, HTML und CSS.

Zuerst wird betrachtet, wie die Kommunikation zwischen den verschiedenen Technologien funktioniert. Das ganze Programm besteht aus Front- und Backend. Das Frontend bezeichnet den Teil des Programms, welches der Nutzer verwendet. Im Backend werden die Daten verarbeitet. Die Verbindung zwischen den beiden wird über JavaScript und JQuery realisiert. Wenn der Benutzer eine Tätigkeit ausführt, werden die eingegebenen Daten über JavaScript Methoden zu dem Backend weitergeleitet. Dort werden sie verarbeitet und über JavaScript Methode aufgerufen. Dort werden sie bearbeitet und wieder zu dem Frontend zugeschickt. Das Backend entsteht aus ASP-Funktionen, in denen sich verschiedenen Methoden befinden, sowie Datenbanken. In den Datenbanken werden die

Abbildung 3.4: Die bisherige Eingabemaske für die Auftragsverwaltung

einggegebenen Daten gespeichert.

Abbildung (neue Diagramm.. die Kommunikation zwischen Front- und Backend bzw. Client -Server)

Nach der grundlegenden Kommunikation zwischen Front- und Backend, wird erläutert wie die einzelnen Kommunikationsabläufe zwischen Front- und Backend koordiniert werden.

3.2.1 Anmelde- und Registerformular

Hier wurde noch nicht korrigiert

Es wird mit dem Bestellseite angefangen, bzw. Anmeldeformular. Wenn die Kunde "Anmeldung" drückt, wird eine HTML Post-Methode aktiviert. So werden JavaScript- und ASP- Methoden aufgerufen. Die JavaScript-Methoden werden benutzt, damit die Eingabe geprüft wird, ob sie korrekt ist, und ASP-Methoden werden dann aufgerufen.

Der Registerformular funktioniert auf demselben Prinzip. Wenn die Eingaben eingegeben sind, werden sie geprüft, ob alles korrekt ist, falls alles in Ordnung ist, werden die Eingaben zu dem zugeordneten Daten, bzw. Methoden zugeschickt.

mehr im Anhang - Funktionen

3.2.2 Auftraggeber-Verwaltung

Hier wird man sich nur mit dem Auftragsverwaltung beschäftigen. Die Auftragsverwaltung, wie es schon beschrieben wurde, besteht auf verschiedenen Optionen. Jetzt lässt sich jeweiliges betrachten, wie es funktioniert.

3.2.2.1 Nachricht-Sektion

ermöglicht dem Auftraggeber die Information zu lesen oder löschen. Nach zugeordnetem Wahl wird die Information in der Datenbank "kommentar" entweder aufgerufen oder entfernt.

3.2.2.2 Aufträge: Neue, alte und bearbeitende

Die Funktionalität von allen drei ist dieselbe. Deswegen es wird allgemein erklärt. Der Auftraggeber kann den Auftrag sehen, editieren oder löschen. Alle von diese Aktivitäten ist mit mehrere Datenbanken verbunden, die verschiedenen Informationen über die

3 IST-Analyse

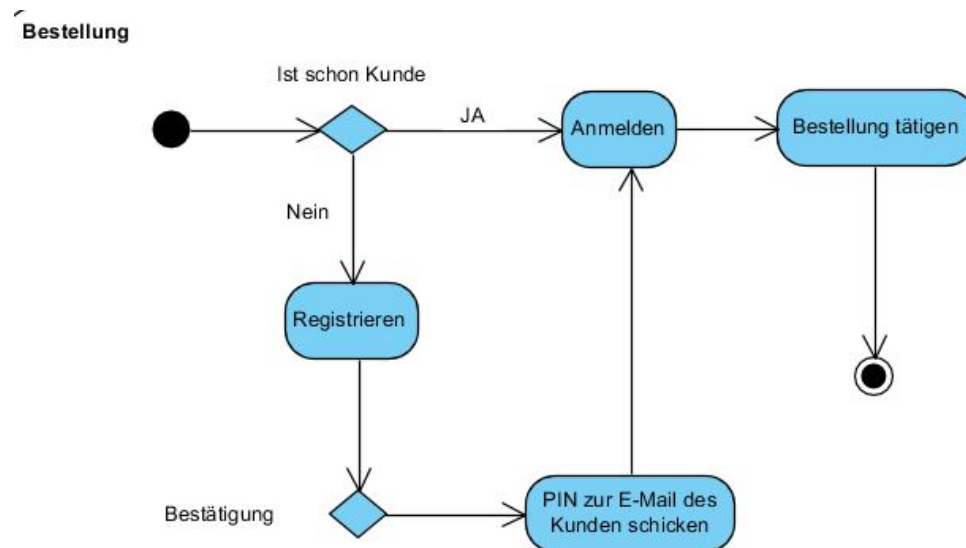


Abbildung 3.5: Die bisherige Eingabemaske für den Vorgang der Bestellung/Registrierung

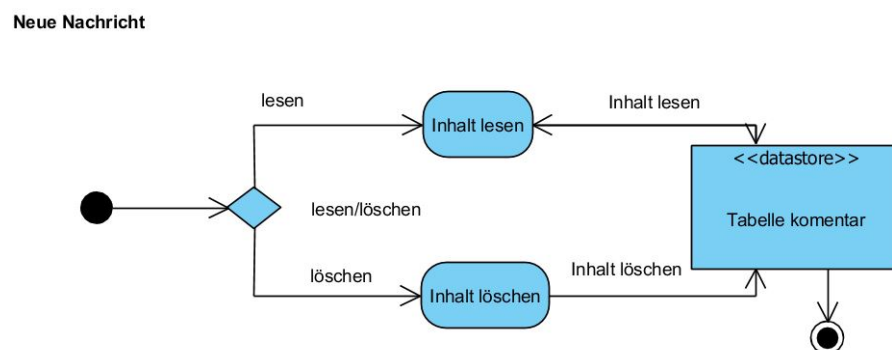


Abbildung 3.6: Die bisherige Eingabemaske für den Nachrichten lesen oder löschen

gewählten Artikel haben. In den Abbildungen 3.7 und 3.8 betrachtet man die jeweilige Funktionalität.

Auftragverwaltung

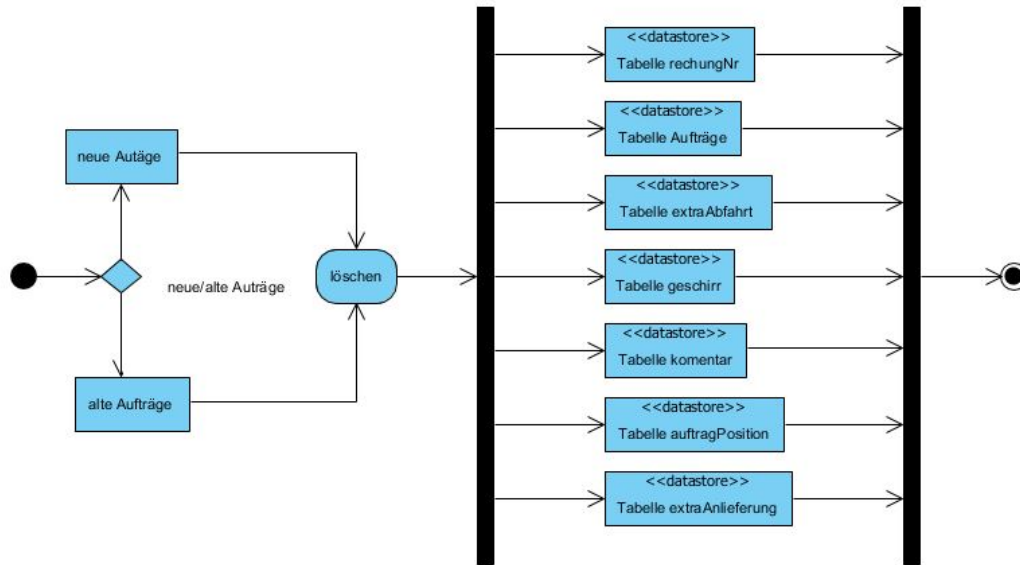


Abbildung 3.7: Die bisherige Eingabemaske für den Nachrichten lesen oder löschen

3.2.2.3 Kundendatei

Hier befindet sich die Information über die Kunden. Da kann man die Information editieren oder löschen. Um der bestimmten Kunde schneller zu finden, steht eine Suchmaschine zur Verfügung. Eine bessere Übersicht wird in der Abbildung 3.9 dargestellt.

Wenn man editieren oder löschen will, wird es über JavaScript Funktion passiert. Diese Funktion ruft die Methoden auf, die sich in der zugeordneten ASP-Datei befindet. Im Editor-Fenster kann man die Daten des Kunden editieren, Login-Daten wie Email an Kunden übermitteln, Wichtige Information für alle Kunden senden, V.I.P-Status des Kunden geben, spezifische Warnungen jeweiligem Kunde aufgeben, Information schreiben, sowie Unterschrift editieren und etc. Besser Verständnis zum diesen Fenster ergibt uns Abbildung 3.10

3 IST-Analyse

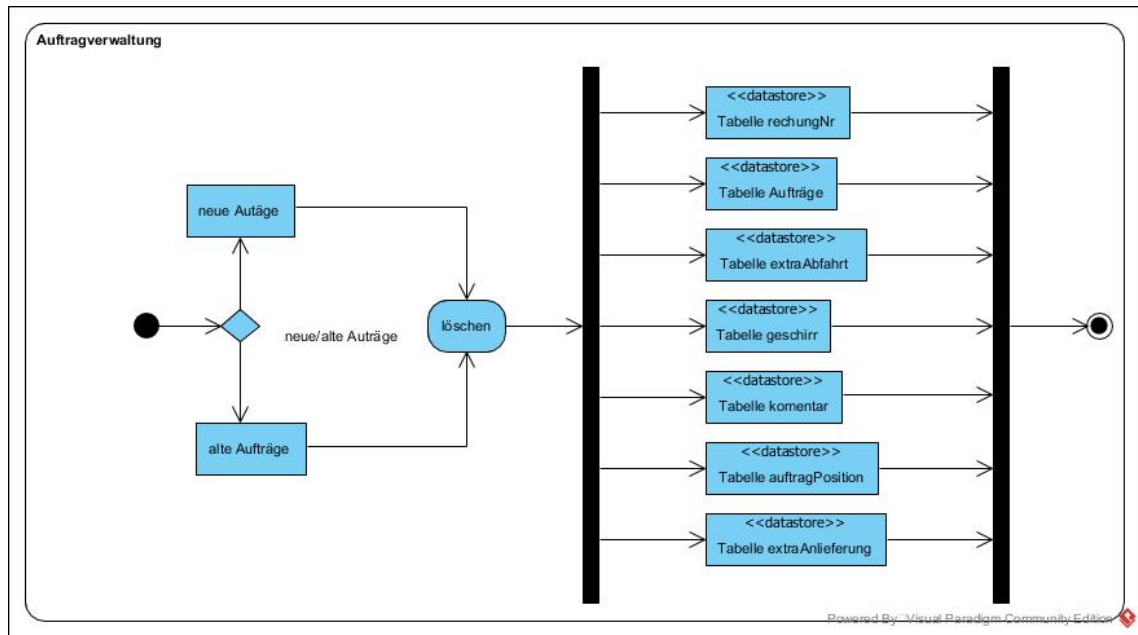


Abbildung 3.8: Die bisherige Eingabemaske für den Nachrichten lesen oder löschen

The screenshot shows a web form for customer data. At the top, there is a search bar with a "suchen" button. Below this, the form is organized into two columns. The left column contains labels for "name:", "adresse:", "ort:", and "email:". The right column contains labels for "telefon:", "mobil:", "facebook", and "pin:". To the right of these columns are two buttons: "Editieren" and "Löschen".

Abbildung 3.9: Die bisherige Eingabemaske für die Kundendatei

Durch JavaScript - Methode wird eine zusätzlichen Möglichkeit zu dem Kunde Nachricht geschrieben. Sie wird aktiviert, wenn der Cursor auf dem bestimmten Kunde, der in der Liste des Adressbuch ist, steht. Die Kommunikation zwischen dem Auftraggeber und dem Kunde wird auch im Datenbank "kommentar" gespeichert.

Nach allem, was geschrieben ist, soll man sich ein vertieftes Verständnis aufbauen. In folgenden Abbildungen 3.11 und 3.12 kann man sehen, wie genau die Aktivitäten des Auftraggeber passieren und wie funktioniert die Nachricht-Kommunikation.

Kundendaten

Name:

Firma:

Adresse:

PLZ:

Ort:

e-Mail:

Telefon:

Handy:

Kunden Pin:

Facebook:

☐ Login-Daten via Email an Kunden übermitteln

Wichtige Informationen für alle Kunden:

24.09.2014 - für Sie jetzt ONLINE:
 Uwe Zimmer - bis 12 Uhr

 - Bitte teilen Sie uns
 Änderungswünsche über das
 Kontaktfenster mit, beachten Sie
 hierbei den Tag für letzte Änderungen.

 - Ab 01. November 2014 ist
 die Zahlung der Rechnung nur noch per
 Vorabüberweisung möglich.

V.I.P.-Status:

Bitte beachten (Kundenspezifisch):

Erfahren:

Info:

<input <="" td="" type="button" value="Unterschrift?"/> <td><input <="" td="" type="button" value="Auswahl?"/> <td><input 218="" 561="" 578"="" 730="" button"="" data-label="Caption" type="button" value="Geduld</td> <td></td> <td></td> <td></td> </tr> </table> </div> <div data-bbox="/>Abbildung 3.10: Die bisherige Eingabemaske für das Kunden-Editor</td></td>	<input <="" td="" type="button" value="Auswahl?"/> <td><input 218="" 561="" 578"="" 730="" button"="" data-label="Caption" type="button" value="Geduld</td> <td></td> <td></td> <td></td> </tr> </table> </div> <div data-bbox="/>Abbildung 3.10: Die bisherige Eingabemaske für das Kunden-Editor</td>	<input 218="" 561="" 578"="" 730="" button"="" data-label="Caption" type="button" value="Geduld</td> <td></td> <td></td> <td></td> </tr> </table> </div> <div data-bbox="/> Abbildung 3.10: Die bisherige Eingabemaske für das Kunden-Editor
---	--	---

3.2.2.4 Online-Editor

Durch diese Option kann man die neuen Artikel zu erstellt, editieren und löschen werden können. Es gibt „Artikel – Arrangements“ und „Artikel – Standard“. Wenn schon ein Artikel erstellt wurde, kann er entweder gelöscht, editiert oder zugeordnet werden. JavaScript Funktionen werden benutzt, um die Eingabe zu prüfen, ob alles korrekt ist. Wenn alles in Ordnung ist, werden die obengenannten Methoden aus den jeweiligen Dateien aufgerufen. In der folgenden Abbildung 3.13 ist die Übersicht der Seite zu sehen.

3 IST-Analyse

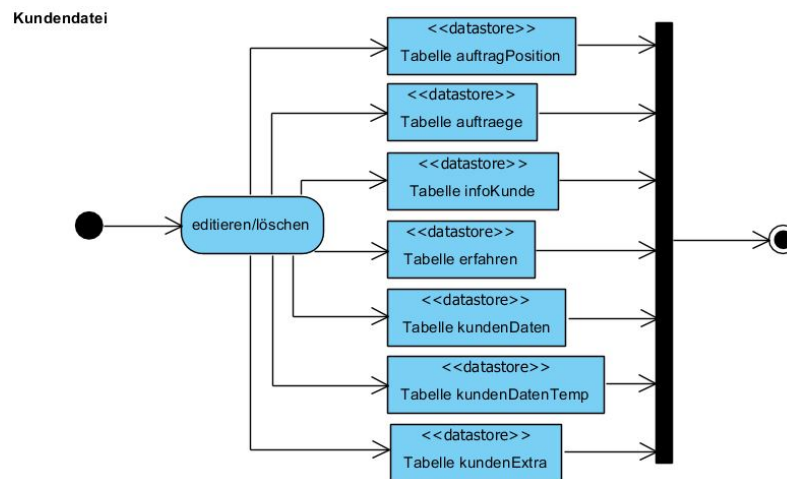


Abbildung 3.11: Die bisherige Eingabemaske für das Editieren/Löschen Funktion

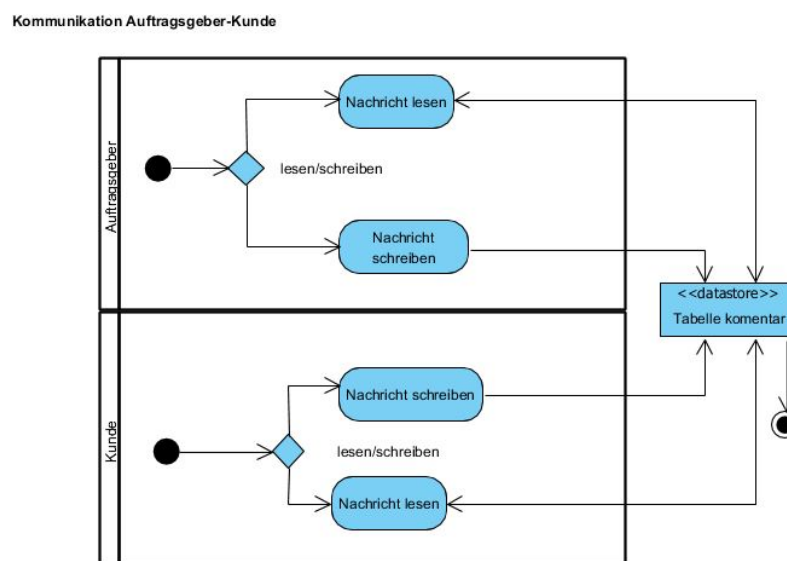


Abbildung 3.12: Die bisherige Eingabemaske für die Kommunikation zwischen dem Auftragsgeber und dem Kunde

Wenn die Option „Editieren“ zu dem „Artikel-Arrangements“ ausgewählt wird, wird neue Seite geöffnet, in der es vielfältige Möglichkeiten gibt, die Inhalt geändert werden kann. Abbildung 3.14 zeigt uns wie diese Seite aussieht.

Artikel-Standard“ hat nur eine Möglichkeit, editiert zu werden. Sie wird in der Abbildung 3.15 dargestellt.

In Abbildung 3.16 ist der Zusammenhang zwischen die verschiedenen Funktionen des „Online-Editor“ zu sehen.

3.3 Testen

Für diese Arbeit wird ein webbasierten Sicherheit-Checker von 1&1. Nach eine vertiefte Recherche, zeigte sich, dass die Webseite auf eine mittlere Sicherheitsposition steht. Die Webseite ist nach vier Kriterien ohne Beanstandung abgesichert.

Online Angebot hinzufügen / löschen

Artikel	Einzelpreis	
<input type="checkbox"/> ** UNPLUGGED ** kalt warmes Gourmet Buffet 26,90 € zzgl. MwSt.	0,00	€
<input type="checkbox"/> ** EVERGREEN ** kalt warmes Gourmetbuffet 25,90 € zzgl. MwSt.	0,00	€
<input type="checkbox"/> ** WIESENGRUND ** kalt warmes Buffet 16,90 € + MwSt.	0,00	€
Standard-Artikel die bei jeder Bestellung vorhanden sind		
<input type="checkbox"/> Anlieferung - Aufschlag 2015 (siehe Website): 12,00 € zzgl. 19 % MwSt.	14,28	€
<input type="checkbox"/> Anlieferung - Aufschlag siehe Website: 10,00 € zzgl. 19 % MwSt.	11,90	€
<input type="checkbox"/> Anlieferung (bitte beachten Sie die Infos auf der Website) ?? km á 0,80 € =	0,00	€

Artikel - Arrangements	Einzelpreis
<input type="text"/>	<input type="text"/>
<input type="button" value="Angebot hinzufügen"/>	

Artikel - Standard	Einzelpreis
<input type="text"/>	<input type="text"/>
<input type="button" value="Angebot hinzufügen"/>	

Artikel - Arrangements ** UNPLUGGED ** kalt warmes Gourmet Buffet 26,90 € zzgl. MwSt.

Artikel - Standard Anlieferung - Aufschlag 2015 (siehe Website): 12,00 € zzgl. 19 % MwSt.

Abbildung 3.13: Die bisherige Eingabemaske für die Übersicht des Menüs

- Secure Sockets Layer (SSL) Verschlüsselung – Über SSL wird sichergestellt, dass zwischen User und Server die übertragenen Daten nicht gelesen werden können.
- Cookies sind auch sicher und so ist der Browser von Dritten über JavaScript unlesbar.
- Apache-Status ist verboten. Die Status-Seite ist öffentlich nicht erreichbar. So wird der Schutz von potenziellen Angreifern erhöht.
- Server Version ist nicht sichtbar. Die Server Version ist öffentlich nicht einsehbar. Damit kann ein Angreifer nicht einfach bekannte Schwachstellen ausnutzen.

Wie man von der Abbildung 3.16 sehen kann, ist die Webseite gesichert, aber es gibt einige Bereiche, die eine Optimierung benötigen würde. Sie lädt sich langsam, wird im Browser relativ oft gefunden, aber ist sie gesichert.

Über noch ein Web-Checker werden die weiteren Funktionalitäten der Webseite analysiert. Observatory[bentley:1999] von Mozilla analysiert den Browser in vier Schritten.

- Hypertext Transfer Protocol (HTTP)
- Transport Layer Security (TLS)
- Secure Shell (SSH)
- **Third-Party Tests! (Third-Party Tests!)**

Erste zwei werden betrachtet, weil nur sie wichtig zu dieser Arbeit sind.

buffet / menü - daten editieren

** UNPLUGGED ** kalt warmes Gourmet Buffet 2€ 0,00

Kunde kann wählen

 3 warmer Spezialitäten mit Gemüsebuffet und 3 Beilagen

1 Gourmetsuppe (zzgl. 1,50 €)

11 kalten Spezialitäten, feine Salate der saison,

 4 Desserts + Käseauswahl + Brotbuffet + Dipsaucen

(Auf Wunsch können Sie sich auch gerne aus dem Buffet Evergreen oder Buffet Saarmania warme Spezialitäten aussuchen, bitte per Telegramm mitteilen)

Anderungen durchführen

Warme Spezialitäten

Ganze Kategorie löschen

aa zarte Médailles von Schweinelendchen mit Pfifferlingsahne

editieren

löschen

ab feiner Wildlachs & Filet vom Saint Pierre in Champagnersauce

editieren

löschen

Dipsaucen

Ganze Kategorie löschen

ai 5 Dips der Saison

editieren

löschen

Kalte Spezialitäten

Ganze Kategorie löschen

ai Roastbeef rosa gebraten mit gefüllten Eiern

editieren

löschen

aj gegrilltes Poulardencarpaccio mit grünem Spargel und Ananas

editieren

löschen

Brotbuffet

Ganze Kategorie löschen

az Partyrad, verschiedene Flutes und kleine Partyweckchen

editieren

löschen

feine Salate

Ganze Kategorie löschen

ba komplettes Salatbuffet der Saison (siehe Website)

editieren

löschen

Dessert

Ganze Kategorie löschen

bm hausgemachte Panna cotta mit Himbeermus im Glas

editieren

löschen

bn Weiße Mousse mit exotischen Früchten

editieren

löschen

Käse

Ganze Kategorie löschen

bs Französische Edelkäse mit Trauben und Salzgebäck

editieren

löschen

position hinzufügen

Warme Spezialitäten

hinzufügen

Abbildung 3.14: Die bisherige Eingabemaske für die Übersicht des Menü-Editor-Arrangements

HTTP Observatory

Hier werden aktuelle Sicherheit der Webseite im öffentlichen Internet beobachtet. Alle Anfragen sind über POST- oder GET-Methoden durchgeführt und die Rückmeldungen sind im JSON-Format. Nach dieser Analyse wird gezeigt, dass **CPS! (CPS!)** nicht implementiert wurde. CSP verhindert Weitbereich von Cross-Site-Scripting (XSS)- und „clickjacking“-Attacken. Cookies sind teilweise gesichert. Hier fehlt es „Secure“-Attribut, sowie „SameSite“ und „Prefix“. Durch „Secure“ werden die Cookies davon abgehalten, dass sie über nicht gesicherten HTTP versendet werden. „SameSite“ hält die Cookies vom Cross-Site-Request-Forgery (CSRF) Attacken und Cross-Site-Übersendung ab, d. h. es ist möglich, dass es einen Hacker-Angriff auf Computersystem des Benutzers durchgeführt werden [bentley:1999]. Es wird keine _Host- und _Secure-Prefixes auf den Namen der Cookies verwendet. Das bedeutet, dass sie nicht gegen Überschreiben geschützt sind. Man kann in der Abbildung 3.18 die genaue Bewertung von der Webseite. stellt das Resultat von der durchgeführten Analyse dar.

TLS

Die betrachtenden von uns Webseite verwendet Signaturverfahren SHA-256-With-RSA. Das bedeutet, dass eine Kombination zwischen SHA256 (kryptologischen Hashfunktion, die Hashwerte mit einer Länge zwischen 256 und 512 Bits erzeugt) und Rivest-Shamir-Adleman

3.4 Warum ist eine Migration notwendig?

Artikel - Standard	Einzelpreis
Anlieferung - Aufschlag 2015 (siehe Website): 12,00 € zzgl. 19 % MwSt.	14,28
Angebot editieren	

Abbildung 3.15: Die bisherige Eingabemaske für die Übersicht des Menü-Editor-Standard

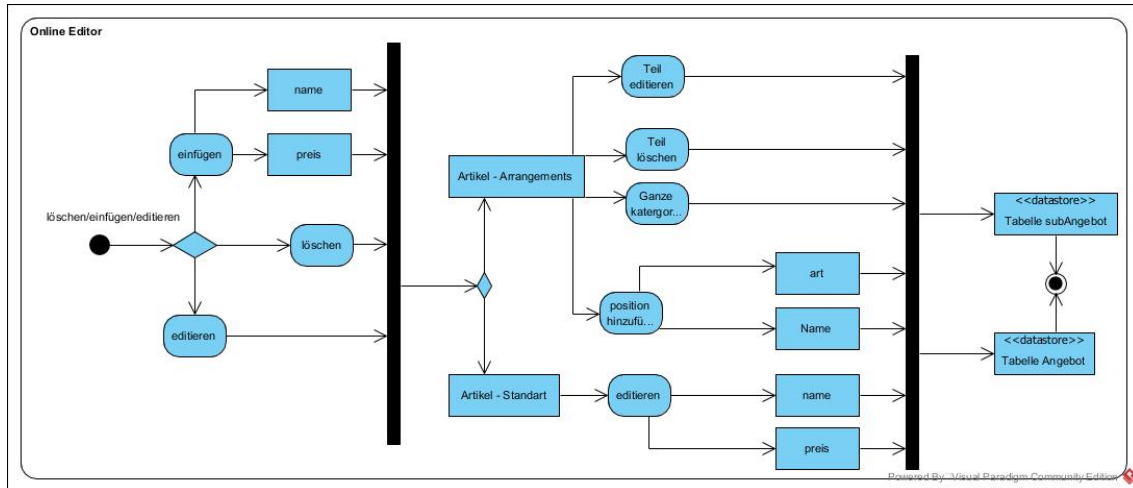


Abbildung 3.16: Die bisherige Eingabemaske für die Funktionalität des Menü-Editor-Standard

(RSA) (ermöglicht symmetrische Schlüssel jeder üblichen Länge verschlüsselt zu werden). In der Abbildung 3.19 stellt dar

3.4 Warum ist eine Migration notwendig?

ASP ist eine nicht Objekt-Orientierte Skripte Sprache. Es ist anfällig zu Unterbrechung der Applikationen, z. B. wenn Internet Information Services (IIS) gestoppt und neu gestartet wird. Wenn eine ASP-Seite aufgerufen wird, wird den Text linear analysiert. Die Frontend Verwendung des Bestellsystems ist verschachtelt und es gibt redundanten Optionen. Auf einer Seite befindet sich Login- und Registrierung Menü, sowie das Bestellformular. Das verlangsamt die Eröffnung der Seite. Es gibt unnötige Quellcode, das entfernt werden kann. Ein anderer Nachteil des Systems ist, dass es nicht dynamisch orientiert ist, d.h. die Umgebung ist nicht Benutzerfreundlich. Der Auftragsgeber muss sich für jede nötige Änderung bei dem Entwickler melden. Das macht den Benutzer abhängig von dem Hersteller des Programms. Heutzutage gibt es vielfältige Auswahl von CMS, durch das der Auftragsgeber mit verbreiteten Möglichkeiten zu Änderungen seiner Website verfügt. Im folgenden Kapitel wird einer diesen CMS erläutert.

3 IST-Analyse

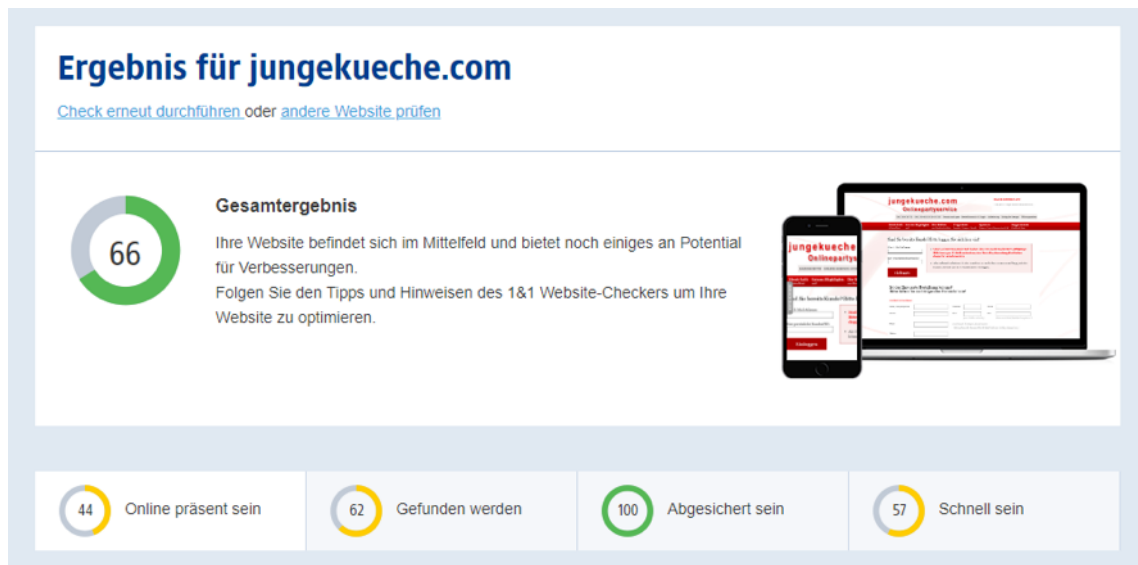


Abbildung 3.17: Ergebnis vom 1zu1

Test Scores			
Test	Pass	Score	Explanation
Content Security Policy	✗	-25	Content Security Policy (CSP) header not implemented ⓘ
Cookies	✗	-40	Session cookie set without using the <code>Secure</code> flag or set over HTTP ⓘ
Cross-origin Resource Sharing	✓	0	Content is not visible via cross-origin resource sharing (CORS) files or headers ⓘ
HTTP Public Key Pinning	=	0	HTTP Public Key Pinning (HPKP) header not implemented (optional) ⓘ
HTTP Strict Transport Security	✗	-20	HTTP Strict Transport Security (HSTS) header not implemented ⓘ
Redirection	✓	0	Initial redirection is to HTTPS on same host, final destination is HTTPS ⓘ
Referrer Policy	=	0	Referrer-Policy header not implemented (optional) ⓘ
Subresource Integrity	✗	-5	Subresource Integrity (SRI) not implemented, but all external scripts are loaded over HTTPS ⓘ
X-Content-Type-Options	✗	-5	X-Content-Type-Options header not implemented ⓘ
X-Frame-Options	✗	-20	X-Frame-Options (XFO) header not implemented ⓘ
X-XSS-Protection	✗	-10	X-XSS-Protection header not implemented ⓘ

Abbildung 3.18: Ergebnis vom HTTP Observatory

3.4 Warum ist eine Migration notwendig?

Certificate Information	
Common name:	jungekueche.com
Alternative Names:	
First Observed:	2018-08-06 (certificate #187995580)
Valid From:	2018-07-27
Valid To:	2018-10-25
Key:	RSA 2048 bits
Issuer:	Let's Encrypt Authority X3
Signature Algorithm:	SHA256WithRSA

Abbildung 3.19: Ergebnis vom TLS Observatory

4 Anforderungsanalyse

In der durchführenden Analyse werden wir uns mit ein neues Bestell- und Verwaltungssystem Konzept beschäftigen. Auf der Basis der IST-Analyse wird eine Verbesserung mit aktuellen Webtechnologie und Verfahren durchgeführt. In diesem Kapitel wird man sich die entstehenden Probleme und Optimierungspotenzial näher anschauen.

Das neue Bestellsystem soll nicht kompliziert sein, sondern Benutzerfreundlich. Die Optionen sollen klar gezeigt werden und einfach sein. Sicherheit ist einen wichtigen Teil der guten webbasierte Entwicklung, aus diesem Grund werden zusätzliche Prüfungen in dieser Richtung entwickelt. Die Auftragsverwaltung und Kundenübersicht werden mit denselben Funktionen bleiben. Das ganze Konzept wird aus einer Umbraco-Instanz heraus verwaltet. In den folgenden Unterkapitel werden die oben beschriebene Anforderungen erweitert und erläutert.

4.1 Paket A

In diesem Unterkapitel wird eine volle Freiheit des Auftraggeber zur Verwendung der Tätigkeiten angefordert. Es wird besonders darauf geachtet, dass die Möglichkeiten zu ändern und editieren, erhalten bleiben. Als Beispiele wurden Schriftart und -farbe vorgegeben. Es wird die Verwendung vom Grids, Macros und Formulare empfohlen. Diese und weitere Begriffen werden im nächsten Kapitel erläutert.

4.2 Paket B

4.2.1 Kundenverwaltung

In diesem Unterkapitel werden wir uns mit den Anforderungen beschäftigen, die der Auftraggeber gestellt hat, um die Kundenaktivitäten besser kontrollieren zu können. Eine eigene Seite wird auch erstellt, in der Kunden Bestellungen erstellen und einsehen können oder Nachrichten verschicken oder lesen können.

4.2.1.1 Kundenerfassung

Hier werden wir die Anforderungen zu den Onlinebestell-System und Kunden Übersicht beschreiben.

1. Hier werden wir die Anforderungen zu den Onlinebestell-System und Kunden Übersicht beschreiben.

Der Kunde kann sich bei der Erstbestellung registrieren und bekommt eine PIN zugeschickt. Bei jede weitere Bestellung kann er sich mit seiner E-Mail und der PIN einloggen. Das was beachtet werden muss ist, dass beim Bestellen die Mail geprüft werden muss und ein Hinweis ausgegeben werden soll, wenn die Mail schon in unsere Datenbank schon existiert/ uns schon bekannt ist. Neukunden kommen in der Bestellübersicht vom Auftraggeber in eine gesonderte Übersicht und müssen von ihm bestätigt oder übernommen werden. Erst dann kann sich der Kunde auch einloggen.

4.2.1.2 Kundenansicht

1. Wenn der Kunde vom Auftraggeber verifiziert wird, kann sich dieser mit Mail und PIN einloggen. Er kann seine aktuelle Bestellung, seine vergangenen Bestellungen und unter Anderen ein Kontaktfenster sehen, über das, die Kommunikation und Absprache mit dem Auftragsgeber läuft.
2. Eine flexible Gestaltung für den Auftraggeber muss auch anwesend sein. So kann er hier auch Informationen platzieren, sowohl allgemein als auch kundenspezifisch.
3. Die Registrierung muss über die Member in Umbraco abgebildet werden.

4.2.1.3 Auftraggeber-Ansicht

1. Der Auftraggeber hat eine Kundenübersicht, die er filtern kann. In der Detailansicht sieht er die Infos zum Kunden wie Bestellverlauf und kann von der Ansicht heraus E-Mails an den Kunden senden (Mail-Vorlage oder frei gestaltbar).

4.2.1.4 Kommunikation

1. Über die Option „neue Nachrichten“ kann der Auftraggeber mit seinen Kunden kommunizieren und Absprachen zu den Aufträgen treffen.
2. Eine übersichtlichere Kommunikationsmethode muss mit Filtern (gelesen, nach Kunden suchen usw.) integriert werden. Der „Chat“ sollte auch aus der Kundenkartei-Ansicht funktionieren und die Kommunikation soll an einen Auftrag gebunden sein.

4.2.2 Artikelverwaltung

4.2.2.1 Artikel erfassen, ändern und löschen

1. Der Auftraggeber kann in einer recht einfachen Maske Artikel verwalten (online-editor). Es gibt nur zwei Kategorien: Arrangements und Artikel-Standard. Im Moment sind die Arrangements noch nicht mit den Webseiten gekoppelt.
2. Ein Arrangement besteht aus Kategorien (fest vorgegeben) und Positionen und kann sich darin unterscheiden, ob Kunden Positionen auswählen dürfen oder nicht.
3. Umsetzungsvorgabe: Artikel sollen einfacher verwaltet werden können und die Artikelseite der Webseite soll direkt mit dem Artikel gekoppelt sein.

4.2.3 Auftragsverwaltung

4.2.3.1 Übersicht

1. Der Auftraggeber hat eine Übersicht über seine aktuellen Aufträge. Für jede Art von Auftrag (neu, bearbeitet, alte und fällig) gibt es ein Select-Befehl.
2. Diese Ansicht soll in einer eigenen Umbraco-subsection umgesetzt werden. Die Artikeldatenbank soll von Access nach SQL transportiert werden. Es muss eine neue Zuordnung Kunde zu Umbraco-Member geben.

4.2.3.2 Detailansicht

1. In der Detailansicht kann der Auftraggeber den Auftrag bearbeiten, den Status ändern, Positionen editieren / hinzufügen und löschen, dem Kunden Freigaben erteilen (z.B. Aussuchen der Positionen) und eine Rechnungsnummer vergeben.
2. Der Auftrag muss auf vier Seiten ausdrückbar (genau vor-gegeben) sein.
3. Die Begrifflichkeiten von zweitem Blatt müssen für Auftraggeber zu Verfügung gelassen werden, damit er sie selbst ändern kann.
4. Quick-Icons erleichtern die Kommunikation mit dem Kunden und die Anpassung an dessen Auswahl zu Auftragsbeginn (z.B. Änderung Serviceauswahl).
5. Diese Ansicht soll in einer eigenen Umbraco-subsection umgesetzt werden.

4.2.4 E-Mail-Verwaltung

1. Hier kann der Auftraggeber E-Mail Vorlagen mit Platzhaltern definieren, die er dann in der Kundenkommunikation auswählen kann.
2. Es gibt ein Form für Terminanfragen über die Webseite. Diese erzeugen eine Mail an den Auftraggeber. Dieser muss in der Mail nur einen Link betätigen, um die Anfrage zu bestätigen. Dies hängt auch mit dem E-Mail Verwaltungssystem zusammen.
3. Diese Ansicht soll in einer eigenen Umbraco-subsection umgesetzt werden.

4.2.5 Umsatzverfassung

1. Hier kann der Auftraggeber sich die Umsätze der letzten Monate / Jahre, die über die Aufträge zustande gekommen sind, an-schauen. Dabei ist es wichtig, dass diese Datensätze nicht direkt an die Aufträge gekoppelt sind, sondern aus einer extra Tabelle kommen, die der Auftraggeber auch selbst noch editieren kann.
2. Diese Ansicht soll in einer eigenen Umbraco-subsection umgesetzt werden.

5 Konzeption und Implementierung

Das neue Konzept, das im diesen Kapitel erklärt wird, geht es um die Migration von Datenbanken von alte webbasierte Software zur neuen Software, die auf Umbraco CMS basiert ist. Die Funktionalitäten müssen gleich bleiben. Das Projekt unterteilt sich auf zwei großen Paketen – Paket A und Paket B. Im ersten Paket wird dem Auftraggeber User Interface (UI) aufgebaut. Das Ziel ist maximale Flexibilität des Website-Besitzers zu haben und durch einfache Tätigkeiten, bestimmten Zweck zu erreichen. Man soll in diesem Bereich die Inhalt der Webseite editieren, ändern und stilisieren. Umbraco verfügt mit sogenannte „Grids“. Es dient dafür, dass man das Design der Seite manipulieren und auch die schon besprochenen Möglichkeiten ausnutzen kann. Es werden auch eigene Macros benutzt, in den ein Quellcode der SHOP – Komponenten hingeschrieben wird. Somit kann der Auftraggeber Macros in beliebigen Teilen der Seiten hinstellen. Umbraco – Forms werden benutzt als Formulare, damit man selber die Felder einordnen kann. Beide Paketen werden als Startknoten aus einer Umbraco-Instanz heraus verwaltet. Paket B ist als Kern der Seite dargestellt. Das ist eigentlich das Online-Bestellsystem. Hier ist es wichtig, dass eine unkomplizierte, bedienbare, reibungslose und flexibel Umgebung aufgebaut wird. Das System besteht aus drei Hauptkerne:

- Bestellsystem
 - Kundenverwaltung (Anmeldung, Kundenbereich, Kommunikation)
 - Artikelverwaltung
 - Auftragsverwaltung
- E-Mail-Verwaltung
 - E-Mail-Vorlagen anlegen, editieren, löschen
- Umsatzerfassung
 - Umsatzübersicht nach Monat und Jahr

5.1 Aufbau vom Umbraco

Wie oben schon erklärt wurde, ist Umbraco [10] ein Content Management System, das flexibel und Benutzerfreundlich ist. Damit man bessere Verständnis hat, worum es geht, wird es in diesem Unterkapitel erläutern.

User Interface von Umbraco ist auf drei Teilen verteilt. Erste Teil ist Hauptfunktion bzw. Setcion. Dort befinden sich die Hauptoptionen: Content, Media. Settings, Developer, Users, Members, Forms. Damit einen klaren Unterschied zwischen „Member“ und „User“ gemacht wird, werden die beiden Begriffe erklärt. „User“ ist jemand, der Zugriff zu „Umbraco-Backoffice“ hat und dort hat bestimmte Rechte.

Ein Member wird von Umbraco für die Registrierung und Authentifizierung eines externen Besuchers benutzt. Das sind Leute, die nur Front-End benutzen dürfen. Man kann auch „Custom Setcion“ erstellen, womit wir uns im weiteren Kapitel beschäftigen.

Nächste Teil ist Unternavigation oder auch Tree genannt. Alle Unteroptionen stehen dort. Jede Hauptfunktion hat ihre eigenen Unteroptionen. Zugehörige Funktionalitäten der Unteroptionen (Trees) werden betrachtet.

- **Trees vom Content-Section:** In diesem Tree befinden sich alle Seiten, die im Website Front-End erschienen werden können. Dort steht auch Recycle Bin, oder auch Papierkorb genannt. So kann man die gelöschte Seite zurücksetzen.
- **Trees von der Media-Section:** Hier stehen alle Videos und Bilder zur Verfügung.
- **Trees vom Settings-Section:** In diesem Tree befindet sich Möglichkeiten CSS, JavaScript, Document Types und zugehörige Templates. PartialView ist auch dort. Es hat eine Übertragungsfunktion von Backend zum Frontend.
- **Trees vom Developer-Section:** Hier stehen zur Verfügung Trees, die den Entwickler ermöglichen, bereits erstellte Seiten weiter zu entwickeln. Das wird durch Data Typ, Macros, Packages, Relation Types, XSLT Files und Partial View Macro Files ermöglicht.
- **Trees vom User:** In diesem Bereich stehen die Benutzer, die mit bestimmtem Rechten Umbraco-Backend benutzen dürfen. Zu einem bestimmten User können verschiedene Rechte abgegeben werden.
- **Trees vom Member-Section:** Hier sind Benutzer, die Frontend benutzen dürfen.
- **Trees vom Forms-Section:** Hier kann man leicht verschiedene Arten von Formularen erstellen.

Der dritte Teil vom Umbraco-Backoffice ist der Editierbereich. Dort werden alle Eigenschaften von jeweiligen Tree-Optionen dargestellt.

Im Abbildung 5.1 kann man sehen, wie Umbraco-Backoffice aussieht.

Aufbau des Backends

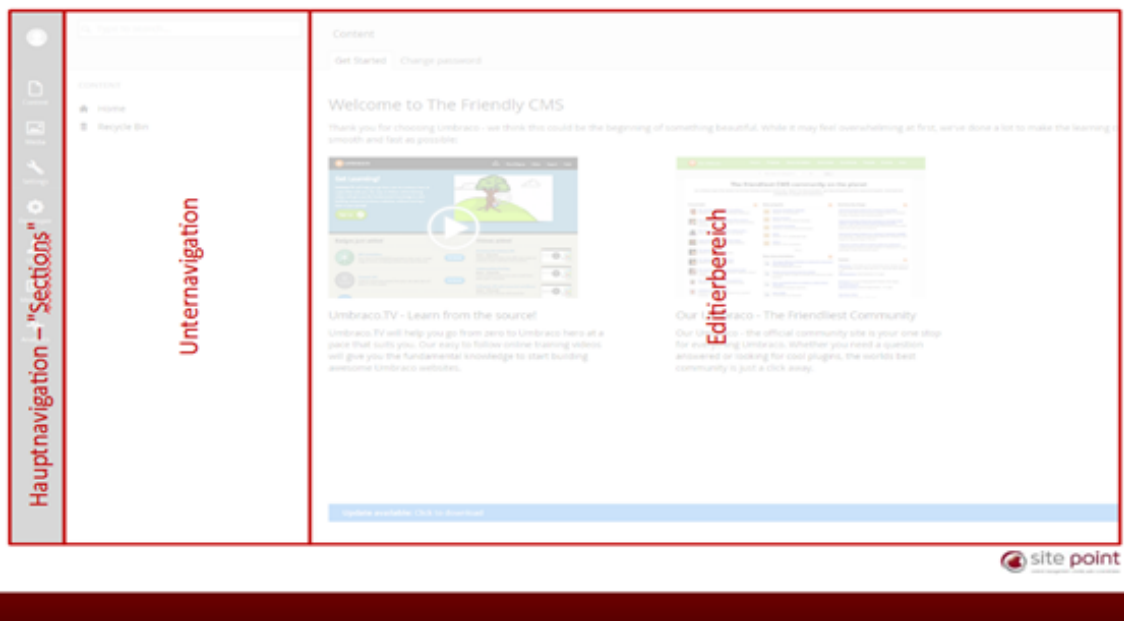


Abbildung 5.1: Übersicht vom Umbraco-Backoffice

5.2 Paket A

Um der Auftraggeber mehr Flexibilität zu haben, die Frontend-Seite zu editieren, werden "Grids Rahmen verwendet. Grid enthält zwölf Spalten. Man kann diese Spalten zusammen binden. So ist Möglich, die Seite auf beliebige Teilen verteilt zu werden, wie im Abbildung 5.2 gezeigt wird.

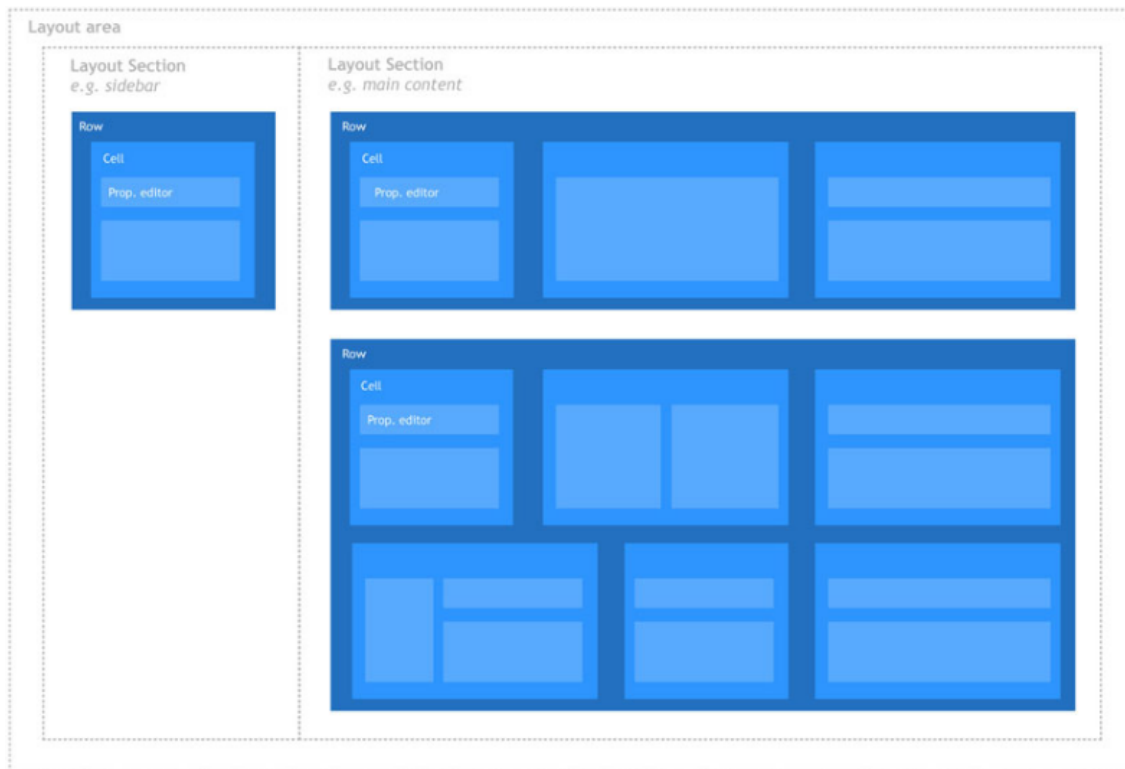


Abbildung 5.2: Übersicht vom Umbraco-Grids

Im Grid kann man eigene Einstellungen machen. In dieser Arbeit werden nur zwei Beispielen gezeigt, wie Grids verwendet könnten: Die Farbe und die Größe vom Schrift ändern. "Controls" wie "Rich text editor", "Image", "Quote", "Macro"... Für angefordertes Ziel werden Rich text editor und Macros verwendet. Man kann Macros auch im Rich text editor benutzen. Das wird im späteren Kapitel gemacht. Jetzt werden nur die schon besprochene Beispiele. Sie werden im Stylesheets unter "rte" eingegeben. In Abbildung 5.3 kann man sehen wie CSS-Befehle im Umbraco geschrieben werden können.

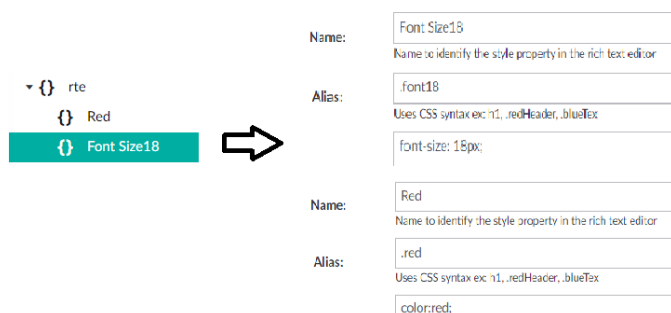


Abbildung 5.3: Styling vom Umbraco-Grids

Über Rich text editor kann man den Text auch editieren, aber nur mit festen Optio-

nen. Diese Stylesheet sind im Richttext editor integrierbar, somit kann Auftraggeber die Schriftart und Farbe ändern.

5.3 Kundenverwaltung

Die Kundenverwaltung fasst den ganzen Bereich um, der sich um die Kunden bezieht. Mehr wird es in nächster Unterkapitel erfahren.

5.3.1 Kundenerfassung

1. Das angestrebte Ziel ist, dass der Kunde sich registrieren kann. Die Registrierung muss gesichert und individuell sein. Damit die Registrierung zuverlässig ist, muss nach den folgenden Regeln gehalten werden: Vertraulichkeit, Integrität, Verfügbarkeit und Authentizität. Ein Personal-Identification-Number (PIN) lässt sich in E-Mail des Kunde zusenden. So wird sichergestellt, dass die Kunde eine gültige E-Mail hat. Nach der Registrierung wird überprüft, ob die E-Mail schon existiert. Wenn ja, das Prozess wird storniert und eine Nachricht erscheint, in der geschrieben ist, dass der Kunde diese E-Mail schon verwendet. Der Auftraggeber entscheidet, ob der sich registrierten Kunde bestellen darf oder nicht. Danach kann der Benutzer sich mit dem PIN und die E-Mail anmelden. Die Kundenverwaltung wird durch Member Application Programming Inter-face (API) von Umbraco realisiert. Wie schon erläutert wurde, ist Umbraco auf ASP.NET basiert. Damit kann man vom CMS unterschiedliche „Services“ benutzen. Ein davon ist „MemberService“. Das ist direkte Kanal zu dem MemberAPI. Es ist in den „Services property“ von dem „SurfaceController“ zu Verfügung gestellt. Die Registrierung wird via ASP.Net Code und Media API programmatisch erstellt. Das ist eine komplexe Methode, in der vielfältige Dateien benötigt werden: (SurfaceController, Model, PartialView und Source Datei, in der View-Methoden über eine Aufruf-Funktion aufgerufen wird). Die benötigte Information, die wir zu der Registrierung brauchen, wird im „Model“-Datei geschrieben. In der Datei „Model“ stehen „Model Properties“. Das sind Parametern, mit denen man arbeitet. Einfach erklärt, über das Model werden die Properties von „Partial-View“ zu dem „SurfaceController“ oder umgekehrt übertragen. „SurfaceController“ ist der „Autobahn“ zu der Umbraco-Datei. Das ist ein Model View Controller (MVC), das mit dem Umbraco interagiert wird. Es wird von der Bibliothek Umbraco.Web.Mvc.SurfaceController geerbt. Über „Partial-View“ werden die Verbindungen zwischen Kontakt Formular und „Model Properties“ geschehen. Das ist eigentlich eine Teilansicht, die von Umbraco Frontend benutzt wird. Dort ist Umbraco User-Interface (UI). Folgende Abbildung 5.4 ergibt bessere Verständnis wie die obengenannten Begriffe zu einander stehen.

Diese Möglichkeit vom Umbraco erlaubt den Benutzer sehr leicht und bequem in dem Server anzumelden oder sich zu registrieren. In Anhang - Kundenerfassung kann man sich übersichtlicher genau anschauen, was es genau gemacht wird, damit diese Anforderung erfüllt wird.

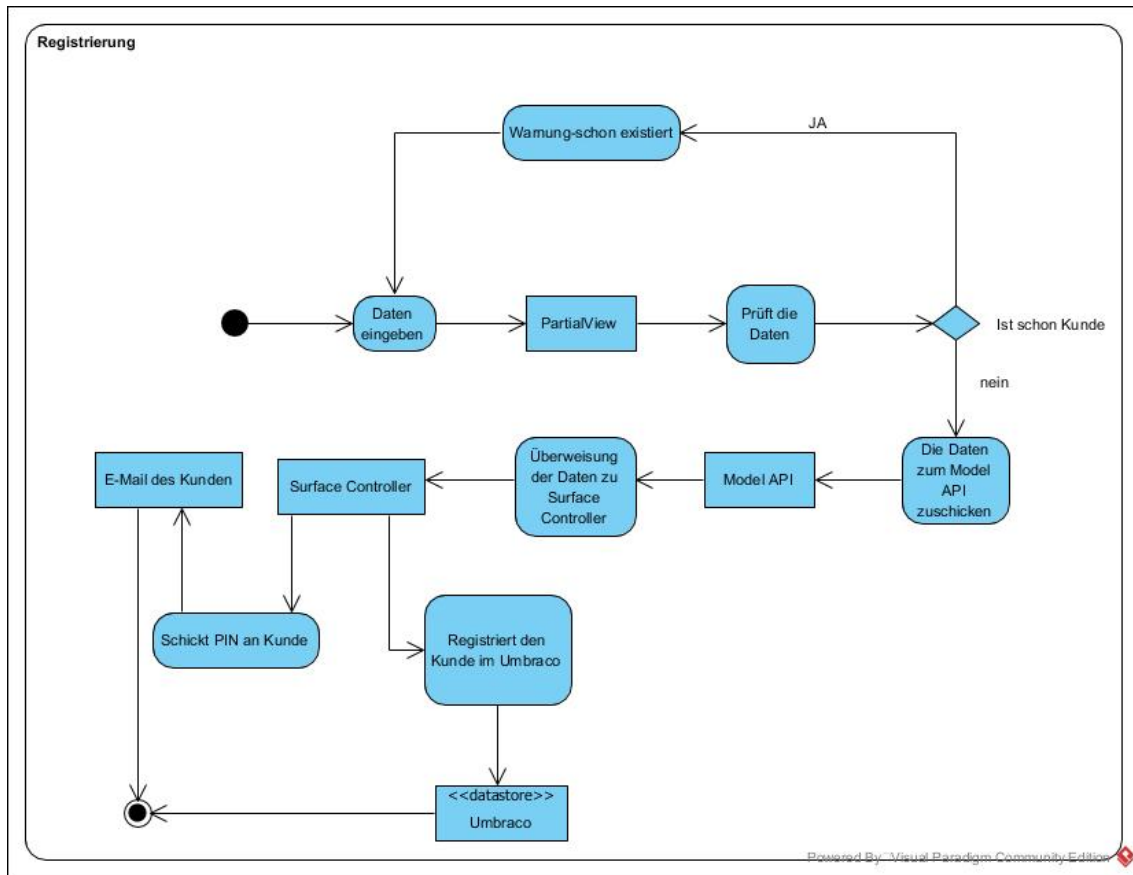


Abbildung 5.4: Neues Konzept zum Registrieren

5.3.2 Kundeansicht

Für Erleichterung des Kunde stehen auf einer Seite alle Möglichkeiten, die der Kunde hat:

- Neue Bestellungen abgeben, aktuelle und vorherige Bestellungen ansehen
- Neue Nachricht schreiben und alte Nachrichten ansehen.
- Wichtige Information zu beachten
- Individuelle Information vom Auftraggeber.

1. Der Kunde bekommt einen PIN an seinem E-Mail. Die Listings 5.1 und A.5 ermöglichen den PIN, obwohl er vom Kunde nicht eingegeben wurde, direkt zur Kunden E-Mail zugechickt zu werden.

Listing 5.1: JavaScript PIN Generator

```

function myFunction() {
document.getElementById('newInput').setAttribute('Value', Math.
    floor((Math.random() * 9000) + 1000));
}window.onload = myFunction;
  
```

2. Nach der Registrierung sieht der Kunde eine neue Seite. Dort kann er die obengenannten Optionen verwenden. Wenn der Kunde neue Bestellung tätigen will, wird ein

5 Konzeption und Implementierung

neues Fenster geöffnet, in dem er erwünschten Artikeln wählen und bestellen kann. Die gewählten Produkte werden in den Datenbanken gespeichert. Von dort werden sie als vergangene Bestellungen verwendet. Das selbe Prinzip steht auch für die Kommunikation zwischen den Auftraggeber und den Kunden. Wenn eine Nachricht geschrieben wird, wird sie in den anderen Datenbanken gespeichert. Vom Umbraco wird die Information direkt zu den Kunden gesendet. Der Auftraggeber, so wie der Kunde, können von der Datenbank die Bestellungen und die Nachfragen ansehen. In den weiteren Kapitel wird detailliert erklärt wie die Kommunikation und die Aufträge funktioniert. 3. Nach angefordertes Ziel wird den Kunde Profile-Seite über Member abgebildet. Das bedeutet, dass diese Seite ist nur zum jeweiligen Kunde personalisiert. Die Abbildung wird über memberId geschehen. Es wird ein Filter erstellt, durch das diese Personalisierung erreichen wird. memberId befindet sich im umbraco-Member Datenbank. Das, was im Kundenansicht gemacht wurde, ist Verwendung von Macros und Grids. Damit man besser versteht, worum es geht, wird folgenden Abbildung 5.5 erstellt.

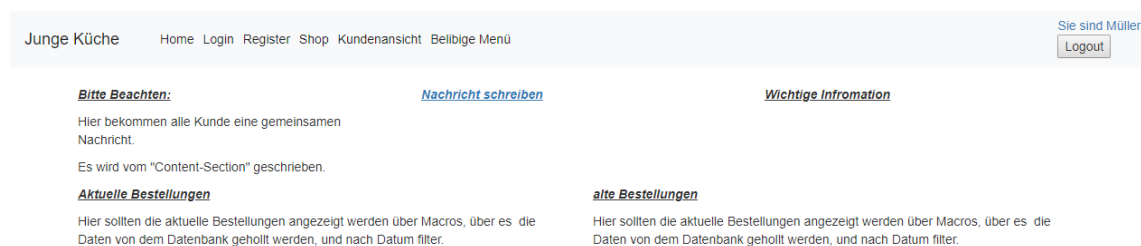


Abbildung 5.5: Kundeansicht

Über ein Macro, das man im Listing 5.2 anschauen kann, ist es möglich die Meldungen vom Auftraggeber unter "Wichtige Information" zu sehen. Im Macro ist eine Methode geschrieben, über die wird die Übertragung vom Member-Section zum Content-Section ermöglicht. Abbildung 5.5 zeigt, wie der Auftraggeber zu dem Kunde eine Informationsmeldung von Member-Bereich schickt.

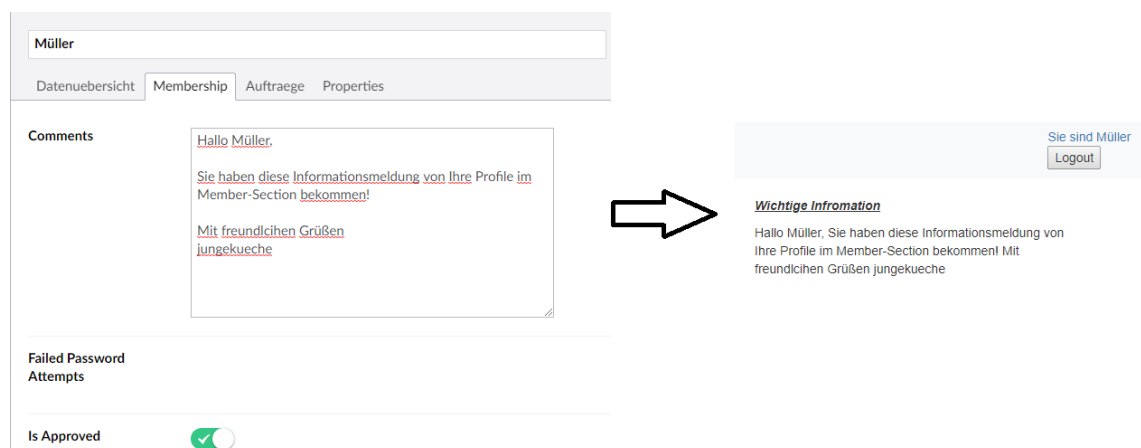


Abbildung 5.6: Übertragung der Informationsmeldung von Member-Section zu Kundenansicht

Listing 5.2: Macro zum Kundenansicht

```

@inherits Umbraco.Web.Macros.PartialViewMacroPage

@{
    var memberID = ApplicationContext.Current.Services.
        MemberService.GetByUsername(Membership.GetUser().
            Username);
    var info = memberID.Comments;
}
@info

```

Das Kontaktfenster und allgemein die Bestellungen werden im weiteren Unterkapitel erläutert. Das gesamte Bild wird im Abbildung 5.7 dargestellt.

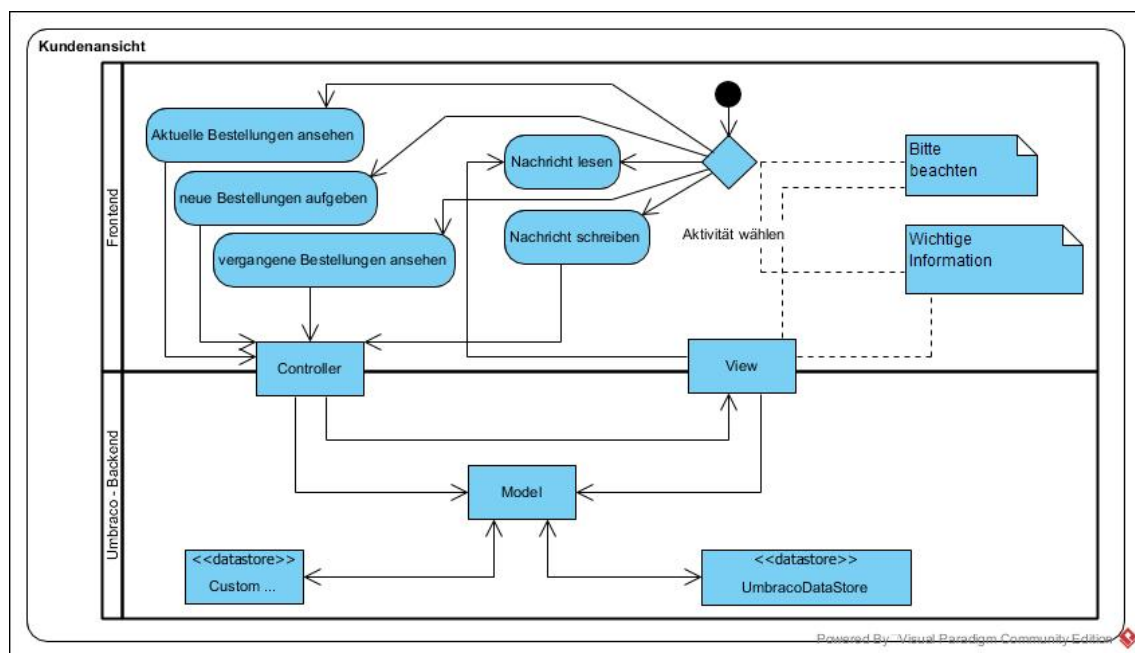


Abbildung 5.7: Funktionalität von Kundenansicht

5.3.3 Auftraggeber-Ansicht

1. Mithilfe vom Umbraco-Member-Section kann der Auftraggeber seine Kunde filtern. Es gibt eingebaute Einstellungen, wie ListView. Durch diese Möglichkeit ist man kann die Kunden im Member-Bereich filtern, nach dem Name suchen oder löschen. Es kann auch zusätzliche Properties erstellt werden. Diese Properties können über extra Attribut im Model-Datei mit dem Frontend verbunden werden. Hier wurden neuen Tabs und Properties erstellt. Als Beispiele wird Tab-"Datenuebersicht" gegeben. Dort befindet sich nur ein Property 'Ört'. Im Tab-Membership ist "Password" auch ein Beispiel, aber man kann mehrere einbauen. Abbildung 5.8 stellt dieses Tab und zugehöriges Property dar.

Wie schon erläutert wurde, ist ein Macro erstellt, durch das die Information zum jeweiligen Kunde zugeordnet ist.

Um die Migration problemlos durchgeführt zu werden, muss man eine neue Zuordnung im Member-Bereich realisieren. Die neuen Properties müssen zu den alten übertragen werden.

Müller

Datenuebersicht Membership Auftraege Properties

Ort Saarbrücken

Abbildung 5.8: Beispiel zur Erstellung eines extra Tab und Property

Attributen von der Member-Datenbank zustimmen. Mehr dafür wird im Unterkapitel "Übersicht" erläutert.

5.3.4 Kommunikation

In diesem Kapitel wird beschrieben, die Kommunikation zwischen dem Auftraggeber und dem Kunde. Als Anforderung wird eine Übersichtliche Kommunikationsmethode mit Filtern (gelesen, nach Kunden suche...) festgestellt.

5.4 Artikelverwaltung

5.4.1 Erfassen, editieren und löschen

In der vorgegebenen Aufgabe muss den Online-Editor mit zwei Kategorien sein - Arrangements und Artikel-Standard. Diese müssen zu den zugehörigen Webseiten gekoppelt werden. Als Anforderung steht, dass Artikel-Verwaltung einfacher verwaltet werden kann und die Artikelseite der Webseite soll direkt mit dem Artikel gekoppelt sein. In der alten Seite ist die Artikel-Verwaltung nicht mit den zugehörigen Seite gekoppelt. Migration ist nicht angefordert.

Das betrachtete Ziel wird über Macros, PetaPoco-Datenbank und Umbraco vorgegebene Funktionen. Zuerst wird Umbraco ListView verwendet, damit die erstellte Artikel sich im Umbraco-Content-Section unter einer gemeinsamen Unteroption befinden. Dazu ist auch vorgegeben, dass den neuen erstellten Artikel direkt editiert werden kann. Abbildung 5.9 zeigt übersichtlicher wie die Artikel-Verwaltung aussieht.

B

Abbildung 5.9: Übersicht der Artikel-Verwaltung

Wenn der Artikel unpublish ist, kann er nicht im Frontend verwendet werden. Wenn die Artikel schon erstellt sind, wird ein Quellcode im Macros programmiert. Diese Macros werden in schon besprochenen Grids integriert, über die der Inhalt der Artikel zum Frontend übertragen wird. Der Inhalt wird nicht nur mit den Webseiten, sondern auch mit dem Shop-Menü gekoppelt. So wird Selbständigkeit des Auftraggebers verbessert. In der nächsten Abbildung wird gezeigt, wie mit der Erstellung von einem Artikel, werden seine Daten sowohl in zugehörige Seite, als auch in der Shop-Menü gekoppelt.

In der Abbildung Wenn der Kunde registriert ist, kann er die von dem Auftraggeber eingegebene Artikel aus dem Shop wählen. Nach der Bestellung werden die Daten in der Datenbank Artikel gespeichert. Gleichzeitig werden diese Daten zusammen mit dem Kunden Daten auch in der Datenbank Aufträge gespeichert.

5.5 Auftragsverwaltung

5.5.1 Übersicht

Das vorgegebene Ziel ist eine Übersicht über die Aufträge und Nachrichten. Diese Ansicht soll in einer eigenen Umbraco-Section umgesetzt werden. Die Artikeldatenbank soll von Access nach SQL transportiert werden. Es muss eine neue Zuordnung zu Umbraco-Member geben.

Das vorgeschlagene Konzept umfasst die Migration von Access Datenbankdatei, Erstellung von neuer Custom-Section.

Um ein neues Section aufgebaut zu werden, braucht man zuerst in einer Klasse-Datei, Methoden von den Bibliotheken `umbraco.businesslogic` und `umbraco.interfaces` aufzurufen. Über `ApplicationMethod` und `IApplicationInterface`. Somit wird die Custom-Section erstellt. Wenn das fertig ist, wird die Übersicht über AngularJS-Controller und HTML erstellt. Diese Dateien werden zu dem API-Controller referenziert. In diesem Controller werden die eigentlichen Funktionalitäten von Custom-Section realisiert. Mithilfe dem API-Controller werden die Daten über Model-Attributen von schon erstellter Datenbank Aufträge zur Verfügung gestanden, die schon im vorherigen Unterkapitel erwähnt wurde. Die Datenbanken werden über PetaPoco erstellt. Allgemein wird die Erstellung vom Custom-Section in folgender Abbildung dargestellt.

Abbildung

Wie es angefordert wurde, soll eine Migration von Access-Datenbank zu Umbraco. Das angestrebte Ziel ist, dass die alten Daten in der neuen zugehörigen Datenbank gespeichert werden. Die neue betrachtete Datenbanken sind Custom-Artikel und Umbraco-Member. Nach tiefer Recherche wird ein Konzept erstellt. Grundsätzlich stehen im Umbraco Member-Section [7] drei Unteroptionen - Members, Member Types und Member Groups. Im Member Type wird die Einstellungen (Properties) des Members aufgebaut. Dort wird eine neue Member Type mit extra Properties erstellt. Folgendes Konzept ist basiert auf der Erwartung, dass die Datenbanken von Access auf CSV-Format konvertiert sind. Das bedeutet, dass die Access-Datenbankformat Datei auf einer Text-Datei umgewandelt wird. Somit kann man die Methode `StreamReader` nutzen. Um das vorgegebene Ziel realisiert zu werden, braucht man ein SurfaceController erstellen, das die Übertragung leitet. Umbraco verfügt mit spezielle Autobannen, durch die möglich ist die Manipulation von festgelegte Sections im Umbraco. Man nennt diese Autobannen `Services`. Sie werden über `ApplicationContext` aufgerufen. In diesem Fall wird `MemberService` genutzt. Damit die Datei gelesen wird, wird die Funktion `StreamReader` verwendet. Somit wird die gelesene Datei in einer Variablen gespeichert. Von dieser Variablen wird mithilfe von dem Befehl `Split` in einem Array gespeichert. Jeder Teil von dem Array wird

5 Konzeption und Implementierung

über MemberService Befehle und Methoden im Umbraco Members gespeichert. Im Anhang A.6 steht das Quellcode, um man besseres Vorbild zu haben. Die Abbildung stellt dar, wie dies Prozess durchgeführt wird.

Abbildung

Wenn die Kundendaten schon übertragen wurden, kann man das selben Prinzip der Migration für die Artikel benutzen. Hier wird als Ziel statt MemberServices - Datenbank ArtikelDatenbank verwendet. Nach der Umwandlung von Access-Datenbankdatei auf CSV-Format, wird diese Datei über die Methode SStreamReader", danach ist die Information in ein Array gespeichert. Über API-Controller werden die Daten in die ArtikelDatenbank übertragen. In Listing 5.3

Listing 5.3: Datenbank Artikel Transfer

```
var setArtikel = new Artikel();
var db = new PetaPoco.Database("umbracoDbDSN");
setArtikel.kundeID = _split[0];
setArtikel.bezeichnung = _split[1];
setArtikel.beschreibung = _split[2];
setArtikel.preis = _split[3];
setArtikel.art = _split[4];
setArtikel.kannWaehelen = m_split[5];

db.Insert(setArtikel);
```

5.5.2 Detailansicht

5.6 E-Mail Verwaltung

5.7 Umsatzverwaltung

Literatur

- [1] 1and1. *Web-Cheker*. 1and1. Juli 2018. URL: <https://www.1and1.com/domain-check>.
- [2] Mary Delamater und Zak Ruvalcaba. *murach's Javascript and jQuery*. Mike Murach und Associates, 2017.
- [3] Jon Duckett. *HTML and CSS Desigt and build websites*. John Wiley und Sons, 2011.
- [4] David Flanagan. *JavaScript*. OReilly, 2011.
- [5] Minko Gechev. *Switching to Angular*. Packt Publishing, 2017.
- [6] April King. *HTTP Observatory Scoring Methodology*. Aug. 2018. URL: <https://github.com/mozilla/http-observatory/blob/master/httpobs/docs/scoring.md>.
- [7] OurUmbraco. *Members*. UmbracoHQ. Sep. 2018. URL: <https://our.umbraco.com/documentation/getting-started/data/members/>.
- [8] SitePoint. *SitePoint*. Juli 2018. URL: <https://www.sitepoint.de/>.
- [9] UmracoHQ. *Umbraco*. UmbracoHQ. Juli 2018. URL: <https://umbraco.com/>.
- [10] Nik Wahlberg und Paul Sterling. *Umbraco User's Guide*. Wiley Publishing, 2011.
- [11] Charles Wyke-Smith. *Stylin with CSS A Disigner's Guide*. New Rider, 2013.

Abbildungsverzeichnis

3.1	Anmeldeformular	7
3.2	Registerformular	8
3.3	Auftragsverwaltung	8
3.4	Kundeansicht	9
3.5	Anmeldung/Bestellung	10
3.6	Kommunikation	10
3.7	AutragLoeschen	11
3.8	auftragEinsehen	12
3.9	Kundeansicht	12
3.10	Kundeansicht	13
3.11	Kundeansicht	14
3.12	Kundeansicht	14
3.13	Kundeansicht	15
3.14	Kun2deansicht	16
3.15	Kundeansicht	17
3.16	Kundeansicht	17
3.17	Egebniss Von 1zu1	18
3.18	Egebniss vom HTTP Observatory	18
3.19	Egebniss TSL Observatory	19
5.1	Umbraco Backoffice	26
5.2	GridsLayout	27
5.3	StylingGrind	27
5.4	neues Konzept: Registrierung	29
5.5	Kundeansicht	30
5.6	Kundeansicht	30
5.7	Funktionalität von Kundenansicht	31
5.8	Kundeansicht	32
5.9	ArtikelVerwaltung	32
A.1	newRegister	45
A.2	MemberDatenuebersicht	50

Tabellenverzeichnis

Listings

Listings

5.1	JavaScript PIN Generator	29
5.2	Macro zum Kundenansicht	31
5.3	Datenbank Artikel Transfer	34
A.1	RegisterModel	43
A.2	RegisterView	43
A.3	PIN-Generator	45
A.4	RegisterConsontroller	45
A.5	Class E-Mail zum Kunde schicken	47
A.6	Migrationsclass	48

Abkürzungsverzeichnis

EDV	Elektronische Datenverarbeitung
WIC	Windows Imaging Components
ASP	Active Server Pages
HTML	Hypertext Markup Language
WWW	World Wide Web
GUI	Graphical User Interface
CSS	Cascading Style Sheets
CMS	Content Management System
HTTP	Hypertext Transfer Protocol
TLS	Transport Layer Security
SSH	Secure Shell
SSL	Secure Sockets Layer
XSS	Cross-Site-Scripting
CSRF	Cross-Site-Request-Forgery
RSA	Rivest–Shamir–Adleman
IIS	Internet Information Services
PIN	Personal-Identification-Number
API	Application Programming Inter-face
MVC	Model View Controller
UI	User-Interface

Anhang

A Anhang -Kundenverwaltung

A.1 Kundenerfassung

Hier kann man besser sehen wie die Umsetzungsvorgabe erfüllt wurde.

Im Model werden die Attributen erstellt, die von den Cotroller und PartialView benutzt werden.

Listing A.1: RegisterModel

```
using System.ComponentModel.DataAnnotations;

namespace newKonzept.Models.Register
{
    public class RegisterModell
    {
        [Required]
        public string Name { get; set; }
        [Required]
        public string Vornane { get; set; }
        [Required]
        public string Ort { get; set; }
        [Required]
        [EmailAddress]
        public string Email { get; set; }
        public string Password { get; set; }
    }
}
```

Im "PartialView"werden vom Model und PIN JavaScript-Funktion-Generator angegeben, die Attribute von Model.

Listing A.2: RegisterView

```
@model newKonzept.Models.Register.RegisterModell
@using newKonzept.Controllers.Register;

@if (TempData["success"] == null)
{
    using (Html.BeginUmbracoForm<CheckIsApprovedController>("
        HandleFormSubmit"))
    {
        using (Html.BeginUmbracoForm<RegisterController>("
            Register"))
        {

            @Html.AntiForgeryToken()
```

```

@Html.ValidationSummary(true)

<fieldset>
<div class="row form-group">
<div class="col-md-6">
<!-- <label for="lname">Last Name</label>
-->
@Html.TextBoxFor(m => m.Name, new { @class
    = "form-control", placeholder = "Name"
})
@Html.ValidationMessageFor(m => m.Name)

</div>
</div>

<div class="row form-group">
<div class="col-md-6">
<!-- <label for="lname">Last Name</label>
-->
@Html.TextBoxFor(m => m.Vornane, new {
    @class = "form-control", placeholder = "
    Vorname" })
@Html.ValidationMessageFor(m => m.Vornane)

</div>
</div>

<div class="row form-group">
<div class="col-md-6">
<!-- <label for="email">Email</label> -->
@Html.TextBoxFor(m => m.Email, new { @class
    = "form-control", placeholder = "Email"
})
@Html.ValidationMessageFor(m => m.Email)
</div>
</div>
<div class="row form-group">
<div class="col-md-6">
<!-- <label for="lname">Last Name</label>
-->
@Html.TextBoxFor(m => m.Ort, new { @class =
    "form-control", placeholder = "Ort" })
@Html.ValidationMessageFor(m => m.Ort)

</div>
</div>
@Html.HiddenFor(m => m.Password, new { id =
    "newInput", Value = "123" })
@Html.ValidationMessageFor(m => m.Password)
@section Scripts{

    <script type="text/javascript" src
        ="~/Scripts/randomNr.js"></
        script>

}

<div class="form-group">

```

```

        <button type="submit" class="btn btn-
            primary">Register</button>
    </div>

    </fieldset>

    }

    }
}
else
{
    <p>Sie haben sich erfolgreich registriert!</p>
}

```

PIN JavaScript-Generator. der PIN wird werden zwischen den Zahlen 1000 und 9999 generiert.

Listing A.3: PIN-Generator

```

function myFunction() {
    document.getElementById('newInput').setAttribute('Value',
        Math.floor((Math.random() * 9000) + 1000));
}window.onload = myFunction;

```

Der Kunde gibt seine Daten ein.

Abbildung A.1: Der Kunde gibt seine Daten ein.

Die eingegebene Daten werden über SurfaceController zum Umbraco Datenbank zu geschickt.

Listing A.4: RegisterConsontroller

```
using System.Web.Mvc;
using Umbraco.Web.Mvc;
using newKonzept.Models.Register;

namespace newKonzept.Controllers.Register
{
    public class RegisterController : SurfaceController
    {
        // GET: Register
        public ActionResult Register(RegisterModell model)
        {
            if (!ModelState.IsValid)
            {
                return CurrentUmbracoPage();
            }

            var memberService = Services.MemberService;

            if (memberService.GetByEmail(model.Email)
                != null)
            {
                ModelState.AddModelError("", "
                    Mietglider/in mit diesem Konto
                    schon existiert!");
                return CurrentUmbracoPage();
            }

            //Schick die Information an Umbraco
            var member = memberService.
                CreateMemberWithIdentity(model.Email,
                    model.Email, model.Name, "neueKunden");
            member.SetValue("ort", model.Ort);
            member.SetValue("password", model.Password)
                ;
            member.IsApproved = false;
            memberService.AssignRole(member.Username, "
                Normal");
            memberService.SavePassword(member, model.
                Password);
            Members.Login(model.Email, model.Password);
            memberService.Save(member);

            return Redirect("/");
        }
    }
}
```

Der Kunde ist mit automatisch generierter PIN im Umbraco Member-Section gespeichert. Jetzt wartet der Kunde auf die Bestätigung.

A.2 Kundenansicht

Listing A.5: Class E-Mail zum Kunde schicken

```
namespace newKonzept.Controllers.Register
{
    public class CheckIsApprovedController : SurfaceController
    {
        // GET: CheckIsApproved

        public ActionResult Index()
        {
            return PartialView("RegisterPartial", new RegisterModell());
        }

        [HttpPost]
        public ActionResult HandleFormSubmit(RegisterModell model)
        {
            void MemberService_Saving(IMemberService sender, SaveEventArgs<
                IMember> e)
            {
                foreach (IMember umbMember in e.SavedEntities)
                {

                    if (!umbMember.IsApproved)
                    {
                        continue;
                    }

                    bool oldValue = ApplicationContext.Current.Services.MemberService.
                        GetById(umbMember.Id).IsApproved;

                    if (oldValue != umbMember.IsApproved)
                    {
                        MailMessage message = new MailMessage();
                        message.To.Add(model.Email);
                        message.Subject = "PIN";
                        message.From = new System.Net.Mail.MailAddress("office@jungekueche.
                            com");
                        message.Body = model.Password;
                        SmtpClient smtp = new SmtpClient();
                        smtp.Send(message);
                    }
                }
            }
            return RedirectToCurrentUmbracoPage();
        }
    }
}
```

A.3 Anhang - Auftraggeber-Ansicht

Listing A.6: Migrationsclass

```
using System;
using System.IO;
using System.Web.Mvc;
using Umbraco.Core.Models;
using Umbraco.Web.Mvc;

namespace newKonzept.Controllers.MemberController
{
    public class MemberController : SurfaceController
    {
        // GET: Member
        public ActionResult ImportMembers()
        {
            string _line = "";
            using (StreamReader sr = new StreamReader(System.Web.HttpContext.
                Current.Server.MapPath("~/DtenbankDatei.csv")))
            {
                do
                {
                    //Liest die Reihe vom CSV-Datei
                    _line = sr.ReadLine();

                    //Prueft, ob die Reihe leer ist.
                    if (!String.IsNullOrEmpty(_line))
                    {
                        //trennt die Reihen mithilfe von ;-Zeichen
                        string[] _splits = _line.Split(';');

                        if (_splits.Length == 8) //Die Zahl hier ist gleich von den Zahl
                            der angegebenen Attribute in der alten Datenbank
                        {
                            //Die Zahlen im _splits[] zeigen die Positionen, in denen sich
                                zugehoeriges Attribut in der altenDatenbank befindet
                            //Estellt neues Member
                            IMember member = Services.MemberService.CreateMember(
                                _splits[3], // Username
                                _splits[2], // Email
                                _splits[6], // Display name
                                "neueKunden" // Member type
                            );

                            //Member ist bestaetigt zu Login
                            member.IsApproved = true;

                            //Prueft, ob zugehoeriges Attribut existiert im Member-Section und
                                dann setzt den Wert ein.
                            if (member.HasProperty("nameDesProperty"))
                                member.SetValue("nameDesProperty", _splits[4]);

                            if (member.HasProperty("nameDesProperty") && !String.IsNullOrEmpty
```

```

        (_splits[5]))
member.SetValue("nameDesProperty", _splits[5]);

if (member.HasProperty("nameDesProperty"))
member.SetValue("nameDesProperty", _splits[0]);

if (member.HasProperty("nameDesProperty"))
member.SetValue("nameDesProperty", _splits[1]);

try
{

//Speichert Member
Services.MemberService.Save(member);

//Speichert PIN des Members
Services.MemberService.SavePassword(member, _splits[7]);

//Korrekte Role zuweisen
Services.MemberService.AssignRole(member.Id, "MemberRole");

}
catch { }
}
} while (!sr.EndOfStream);

}

return null;

}
}
}

```

<input type="text" value="Müller"/>	
<div>Dateneübersicht Membership Auftraege Properties</div>	
Ort	<input type="text" value="Saarbrücken"/>

<input type="text" value="Müller"/>	
<div>Dateneübersicht Membership Auftraege Properties</div>	
<div></div>	

Failed Password Attempts	
--------------------------	--

Is Approved	<input type="checkbox"/>
-------------	--------------------------

Is Locked Out	No
---------------	----

Last Lockout Date	
-------------------	--

Last Login Date	9/28/2018 1:58:30 PM
-----------------	----------------------

Last Password Change Date	9/28/2018 1:58:16 PM
---------------------------	----------------------

Password Answer	
-----------------	--

Password Question	
-------------------	--

Password	<input type="text" value="8748"/>
----------	-----------------------------------

Abbildung A.2: Die Kunde im Members gespeichert.

Kolophon

Dieses Dokument wurde mit der L^AT_EX-Vorlage für Abschlussarbeiten an der htw saar im Bereich Informatik/Mechatronik-Sensortechnik erstellt (Version 2.1). Die Vorlage wurde von Yves Hary und André Miede entwickelt (mit freundlicher Unterstützung von Thomas Kretschmer, Helmut G. Folz und Martina Lehser). Daten: (F)10.95 – (B)426.79135pt – (H)688.5567pt