Kundenzugang als Web Applikation für Online-Marketing Kampagnen

Abbildung 1 Titelbild

Autor: Hubert Thalmann
Hauptexpertin: Ursula Reinhard
Nebenexperte Daniel Jäggli
Fachvorgesetzter: Ignaz Walgis
Ausbildungsbetrieb: Chili Solutions GmbH
Datum: 16. April 2018
Version: 1.0 Finale Version

Teil 1 des IPA Berichts:

Obligatorische Kapitel

Enthält Aufgabenstellung, Projektorganisation, Zeitplan und Journal

Vorwort

Ich befinde mich zurzeit im Praktikumsstadium meiner Ausbildung. Meine Ausbildung besteht aus 2 Jahren Schule und 2 Jahren praktischer Ausbildung in einem Betrieb. Ich absolviere mein Praktikum bei der Chili Solutions GmbH in Zürich, welche im Bereich der Webentwicklung mit .NET aktiv ist.

Um meine 4 jährige Lehre abzuschliessen wird eine 10 tägige Abschlussarbeit durchgeführt. Diese Arbeit wird im Praktikumsbetrieb erfolgen und enthält Elemente, die ich im Rahmen meiner Ausbildung gelernt und vertieft habe, aber auch Elemente die neu für mich sind und in denen ich zuerst Kenntnisse sammeln muss.

Chili bietet Onlinemarketing für ihre Kunden an. Wir erstellen AdWords-Kampagnen und verwalten diese. Damit ein Kunde sehen kann, wie erfolgreich seine Kampagne ist, erhält er jeden Monat einen Report mit den wichtigsten Kennzahlen seiner Kampagne. Diese Prozedur ist bisher von Hand gemacht worden, es wurde dafür ein PDF erstellt und dem Kunden zugestellt. Meine Aufgabe besteht jetzt darin diese Lösung zu modernisieren und zwar in Form einer Webapplikation. Kunden sollen sich einloggen und ihre Reports direkt online ansehen können. Zusätzlich sollen die Reports sauber auf ein A4 Papier ausgedruckt werden können.

Diese Arbeit ist ein wichtiger Schritt zum Abschluss meiner Lehre. Ich sehe die Arbeit als grosse Herausforderungen und freue mich darauf, die Lösung möglichst sauber und schön umzusetzen. Ich arbeite im gewohnten Umfeld und werde zeigen, was ich im Rahmen meiner Ausbildung zum Informatiker alles gelernt habe. Als grosse Herausforderung sehe ich die Zeitplanung. Bisher habe ich noch kein Projekt dieser Grösse umgesetzt und bin gespannt, ob ich die Grösse richtig kalkulieren kann. Ich bin sehr motiviert und freue mich auf diese Arbeit.

Inhaltsverzeichnis

Versionsverlauf

Version	Datum	Änderung
0.1	03.04.18	Erstellung des Dokuments, Management Summary, Projektphase informieren erläutert
0.2	04.04.18	Grobkonzept der Webapplikation fertiggestellt, Systementwurf hinzugefügt, Testfälle definiert, Dokumentation ergänzt
0.3	05.04.18	Planung mit Mockups ergänzt, mit Entscheidungsphase begonnen, Möglichkeiten aufgezeigt und entschieden. Überarbeitung Grobkonzept
0.4	06.04.18	Mit der Realisierungsphase begonnen und dokumentiert.
0.5	09.04.18	Arbeitsjournal Tag 5 ergänzt, heute habe ich nichts dokumentiert, sondern einfach mal programmiert. Die Dokumentation werde ich anschliessend ergänzen.
0.6	10.04.18	Arbeitsjournal Tag 6 geführt, begonnen mit der Dokumentation über die Realisierungsphase 1 Login
0.7	11.04.18	Arbeitsjournal Tag 7 geführt Realisierungsphase 2 und Realisierungsphase 3 dokumentiert und Dokumentation allgemein überarbeitet
0.8	12.04.18	Arbeitsjournal Tag 8 geführt Realisierung weiter Dokumentiert, Tests durchgeführt und Dokumentiert
0.9	13.04.18	Arbeitsjournal Tag 9 geführt Reflexion geschrieben, Danksagung geschrieben und Dokumentation überarbeitet
1.0	16.04.18	Arbeitsjournal Tag 10 geführt, Dokumentation überarbeitet

1.2 Detailbeschreibung[2]

1.2.1 Ausgangslage

Chili führt für Kunden Online-Marketing Kampagnen mit Google-AdWords durch. Bisher wurden die monatlichen Kunden-Reports mit Hilfe von Excel und Word aufbereitet, in ein PDF umgewandelt und den Kunden per E-Mail verschickt. Es soll nun Web-Applikation programmiert werden bei welchem die Kunden einloggen können und sofort Zugriff auf die Kampagnenkennzahlen haben. Der Kunde kann somit die Kennzahlen seinen aktuellen Report abrufen sowie vergangene Reports abrufen. Die Kampagnenentwicklung soll in einer Trend-Grafik visualisiert werden.

1.2.2 Detaillierte Aufgabenstellung

Es soll eine .NET MVC Web Applikation in C#, HTML 5, CSS3 erstellt werden:

- die öffentlich via Internet zugänglich ist über Adresse: customers.chili.ch. Die Adresse darf nicht von öffentlichen Seiten verlinkt sein und darf nicht von Suchmaschinen indexiert werden.
- die Web Applikation soll in Responsive Design programmiert sein. Wobei nur die Ausprägung in Grösse Desktop ausprogrammiert werden soll.
- die im „Look und Feel" der bestehenden Chili Website ist (www.chili.ch). Die Web Applikation soll in den neuesten Versionen IE, Firefox und Chrome sauber dargestellt sein und fehlerfrei funktionieren
- bei dem der Kunde mit Kunden-Nr. und Passwort einloggen kann. SQL-Injection muss verhindert werden. Alle Logins ob erfolgreich oder nicht sollten in der Tabelle „TAB_Adresse_History_Login" historisiert werden. Es soll automatisch nach 15 Minuten ausgelogged werden, wenn keine Benutzer-Interaktion stattgefunden hat.
- der Kunde die aktuellen Kennzahlen des Reports einsehen kann. Zum Vergleich werden die Kennzahlen vergangener Reports immer max. 6 Monate zurück dargestellt.
- der Kunde kann vergangene Reports abrufen. Die Selektion dazu kann über ein Dropdown gemacht werden. Zum Vergleich werden die Kennzahlen vergangener Reports immer max. 6 Monate zurück dargestellt.
- bei dem der Kunde die Reports via Browser auf 1 A4 ausdrucken kann (sauberes Print CSS)
- bei dem der Kunde wieder ausloggen kann
- Es gelten die Programmier-Standards von Chili, Stand 09.06.2016, separates Dokument

Report:

Ein Report enthält folgende Elemente und Kennzahlen

- Titel: Online-Marketing Report Monat Jahr für „Titel" aus Tabelle TAB_Hosting
- Massnahmen der Kampagne im Vormonat in Listenform
- Kommentar zur aktuellen Kampagne
- Tabelle mit Titel „Kampagnenperformance" und folgenden Kennzahlen vertikal: Impressions, Klicks, CTR, Conversions, Conversion-Rate, Kosten/Conv. (CHF), CPC, Kosten (CHF). Vertikal werden die Monate (max. 6 Monate zurück) und die Veränderung zum Vormonat in % dargestellt. Der aktuelle Monat ist visuell erkennbar.
- Massnahmen der Kampagne des aktuellen Vormonates in Listenform
- Ziele der aktuellen Kampagne
- Tabelle mit Titel „Traffic (ganze Site)" mit folgenden Kennzahlen:
- Besuche aktueller Monat und Vormonat zum Vergleich in absoluten Zahlen und Veränderung in Prozent.
- Absprungrate in % aktueller Monat und Vormonat zum Vergleich in absoluten Zahlen und Veränderung in Prozent.

[2] Detailbeschreibung wurde aus PkOrg übernommen

- Durchschnittliche Verweildauer aktueller Monat und Vormonat zum Vergleich in absoluten Zahlen und Veränderung in Prozent.
- Balkendiagramm mit Titel „Zusammensetzung des Traffic aus wichtigsten Besucherquellen". Siehe Vorlage.
- Fusszeile: „chili Onlinemarketing Report – Monat Jahr – Kundenname"

1.2.3 Betroffene Tabellen:

Alle Tabellen bestehen. Es müssen keine neuen erstellt werden.

Tabelle: TAB_Adresse
Felder: NUM_Nummer (Kunden-Nr., Primärschlüssel), Passwort
Zweck: Überprüfen der Login-Daten. Sofern korrekt, können die Marketing-Kampagnen dem Login zugewiesen werden.

Tabelle: TAB_Hosting
Felder: ID (Primärschlüssel), NUM_Kunde (Kunden-Nr., Fremdschlüssel), Titel
Zweck: Alle Einträge mit Typ „Marketing Service" enthalten die Online-Marketing Kampagnen eines Kunden.

Tabelle: TAB_Marketing_Kampagne
Felder: Hosting_ID (Fremdschlüssel), Marketing_Kampagne_ID (Primärschlüssel), Monat, Jahr, Impressions, Klicks, CTR, Conversions, Conversion-Rate, CostPerConversion, CPC, Kosten, Ziele, Kommentar

Zweck: Pro Monat werden die Kennzahlen pro Kunde via Chili-Adress DB erfasst. Hier können die Daten einer Kampagne ausgelesen werden.

Tabelle: TAB_Marketing_Kampagne_Massnahme
Felder: Marketing_Kampagne_ID (Fremdschlüssel), Massnahme
Zweck: Massnahmen einer einzelnen Kampagne werden hier ausgelesen.

Tabelle: TAB_Marketing_Kampagne_Traffic
Felder: Marketing_Kampagne_ID (Fremdschlüssel), Verweise, Suchmaschinen, Sonstige, Absprungrate, DurchschnittVerweildauer, Kommentar
Zweck: Traffic Kennzahlen einer einzelnen Kampagne werden hier ausgelesen.

Tabelle: TAB_Adresse_History_Login
Felder: NUM_Kunde (Kunden-Nr., Fremdschlüssel), ID (Primärschlüssel), Datum, Zeit, Text, IP, UserAgent
Zweck: Hier sollen die Login (erfolgreich und fehlgeschlagen) historisiert werden.

1.2.4 Mittel und Methoden

- Visual Studio 2017, .NET MVC C#
- Visual Studio Team Services
- MS Word
- MS Excel
- SQL Management Studio
- Visio

1.2.5 Vorkenntnisse

- HTML 5, CSS3, JavaScript
- jQuery
- Visual Studio 2017, .NET MVC C#
- Visual Studio Team Services

1.2.6 Vorarbeiten

keine

1.2.7 Neue Lerninhalte

- Diagramme
- Session Handling

1.2.8 Arbeiten in den letzten 6 Monaten

Website: www.chili.ch, HTML 5, CSS3, JavaScript, jQuery. Anbindung an CMS Kentico, C#

Chili intern: Abnahmeformular in HTML, Backend Programmierung in MVC, Razor, Datenbankzugriff mit Entity-Framework

Diverse Responsive-Design Programmierungen in HTML 5, CSS3, jQuery. Anbindungen an CMS Umbraco, Razor

1.3 Zeitraum der IPA

Kalenderwoche	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
14	Ostermontag	4. April Starttermin IPA Ganztags	5. April IPA Ganztags	6. April IPA Ganztags	7: April IPA Ganztagsl
15	9. April IPA Ganztags	10. April IPA Ganztags	11. April IPA Ganztags	12. April IPA Ganztags	13. April IPA Ganztags
16	16. April Abgabetermin IPA Ganztags				

1.4 System und Arbeitsumgebung

Die IPA führe ich an meinem gewohnten Arbeitsplatz bei Chili durch.

Hardware Computer

- Prozessor: Intel Core i7 3770
- RAM : 16 GB
- Betriebssystem Windows 10 (64 Bit)
- Grafikkarte: NVIDIA GeForce GTX 650

Software Computer

- Microsoft Visual Studio 2017 mit reSharper
- Microsoft Office 2013
- SQL Server Management Studio
- Windows Server 2012 auf Serverumgebung
- Google Chrome als Standardbrowser (mit Browsersync VS 2017)
- Visual Studio Team Services

1.5 Vorwissen

Bei Chili als Webagentur hatte ich Einblicke in die verschiedensten Bereiche der Webentwicklung und App-Entwicklung:

Mein Vorwissen als Diagramm:

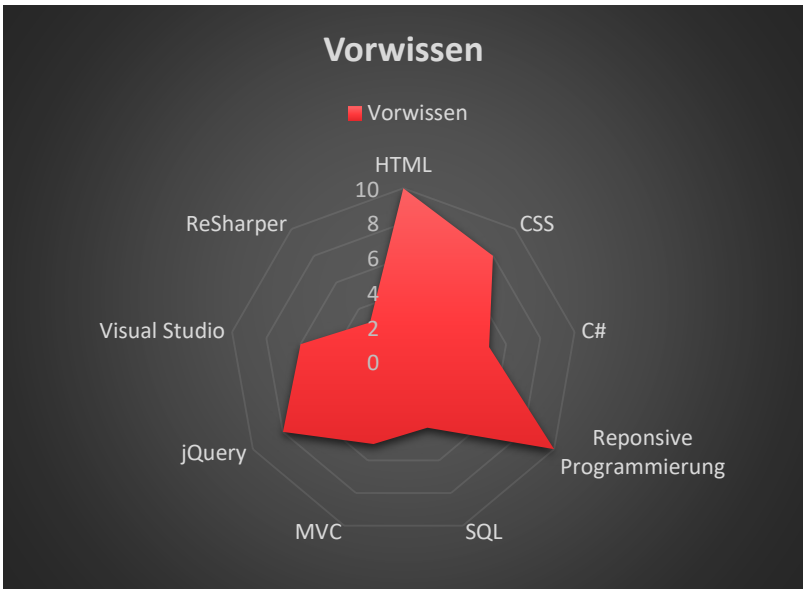


Abbildung 2 Diagramm Vorwissen

1.6 Vorbereitung auf die IPA

Als Vorbereitung auf meine IPA habe ich kleinere Testprojekte realisiert mit MVC. Hauptsächlich habe ich diese Dinge geübt und versucht zu verstehen:

- Datenbankzugriffe
- Entity Framework
- Mehrere Projekte in einer Lösung
- Eigene Models erstellen
- Eigene Controller erstellen
- Eigene Views erstellen

1.7 Firmenstandards

1.7.1 Programmierrichtlinien

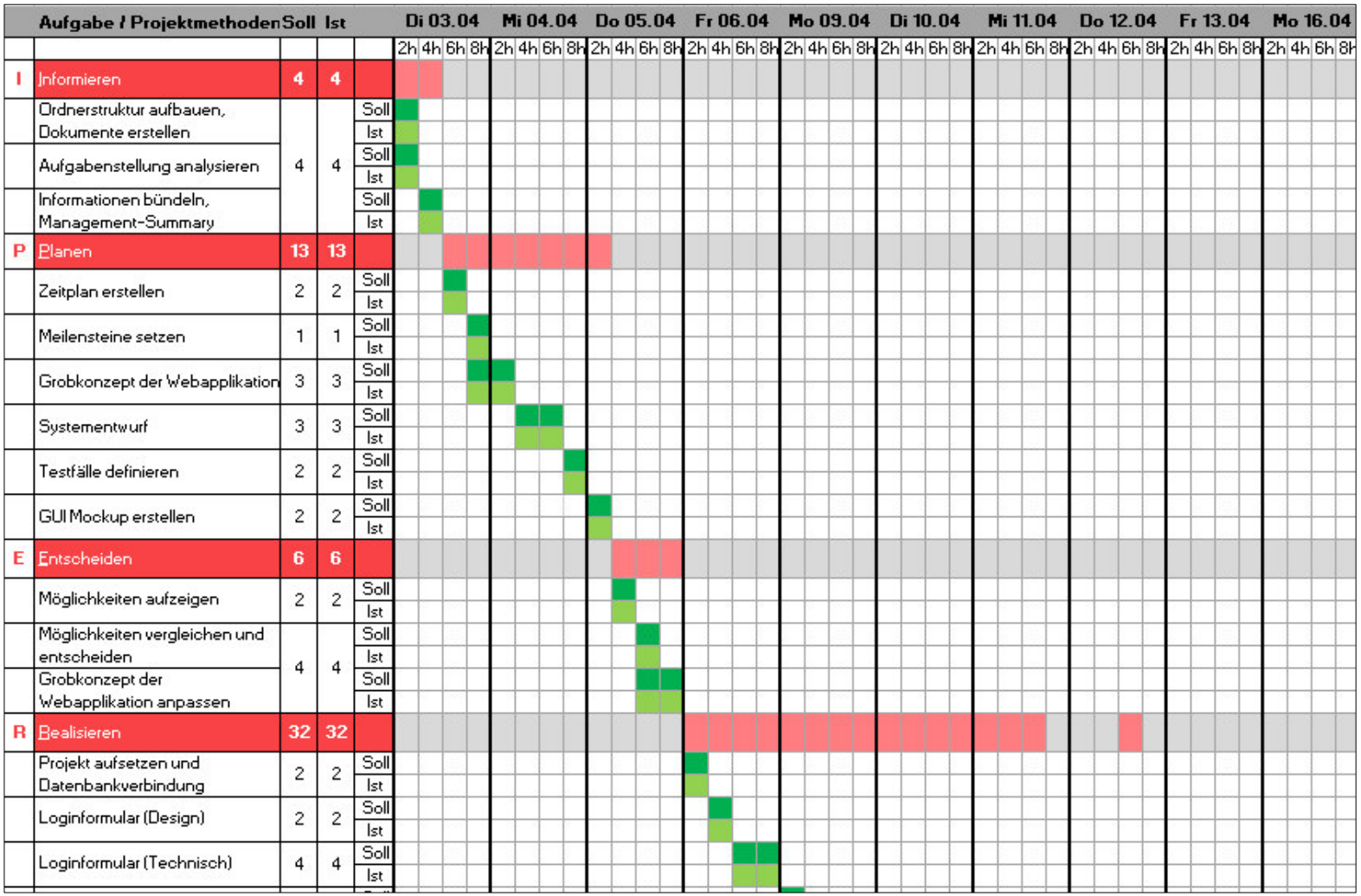
Chili Solutions GmbH hat eigene Programmierichtlinien an die wir uns während der Programmierung halten. Diese sind in der Detailbeschreibung in PKOrg angehangen worden. Die Standards verfolgen folgende Zwecke

- Der erzeugte Code soll leicht lesbar und erweiterbar sein.
- Diese Standards sollen unsere Erfahrungen mit verschiedenen Programmiersprachen und Technologien wiederspiegeln und werden laufend mit neuen Erkenntnissen erweitert.
- Es soll die Entwicklung von Komponenten erleichtern.
- Es soll die Waage halten zwischen zu viel Standard und zu viel Individualismus des einzelnen Programmierers.

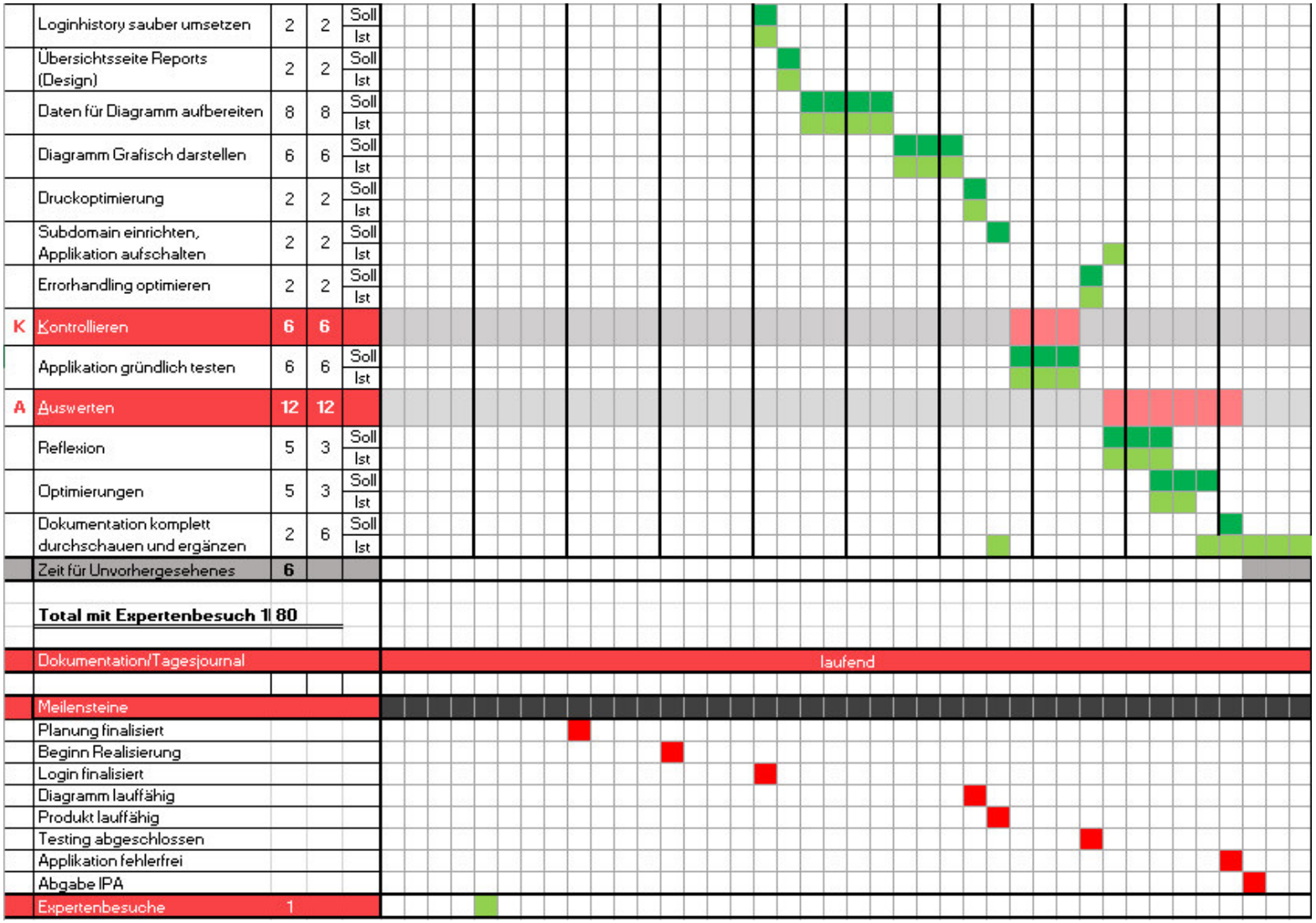
Kommentare: Nebst den Kommentaren innerhalb des Codes wird ein Block mit Kommentaren an wichtigen Stellen in den Code eingefügt. Dieser beinhaltet Basisinformationen zur jeweiligen Datei (Erstelldatum, Ersteller, Zweck etc.).

1.8 Zeitplan

Der Zeitplan wurde zusätzlich als Anhang im PKOrg hochgeladen.



	Aufgabe / Projektmethoden	Soll	Ist		Di 03.04				Mi 04.04				Do 05.04				Fr 06.04				Mo 09.04				Di 10.04				Mi 11.04				Do 12.04				Fr 13.04				Mo 16.04			
					2h	4h	6h	8h	2h	4h	6h	8h	2h	4h	6h	8h	2h	4h	6h	8h	2h	4h	6h	8h	2h	4h	6h	8h	2h	4h	6h	8h	2h	4h	6h	8h	2h	4h	6h	8h	2h	4h	6h	8h
I	Informieren	4	4																																									
	Ordnerstruktur aufbauen, Dokumente erstellen	4	4	Soll																																								
				Ist																																								
	Aufgabenstellung analysieren			Soll																																								
				Ist																																								
	Informationen bündeln, Management-Summary			Soll																																								
				Ist																																								
P	Planen	13	13																																									
	Zeitplan erstellen	2	2	Soll																																								
				Ist																																								
	Meilensteine setzen	1	1	Soll																																								
				Ist																																								
	Grobkonzept der Webapplikation	3	3	Soll																																								
				Ist																																								
	Systementwurf	3	3	Soll																																								
				Ist																																								
	Testfälle definieren	2	2	Soll																																								
				Ist																																								
	GUI Mockup erstellen	2	2	Soll																																								
				Ist																																								
E	Entscheiden	6	6																																									
	Möglichkeiten aufzeigen	2	2	Soll																																								
				Ist																																								
	Möglichkeiten vergleichen und entscheiden	4	4	Soll																																								
				Ist																																								
	Grobkonzept der Webapplikation anpassen			Soll																																								
				Ist																																								
R	Realisieren	32	32																																									
	Projekt aufsetzen und Datenbankverbindung	2	2	Soll																																								
				Ist																																								
	Loginformular (Design)	2	2	Soll																																								
				Ist																																								
	Loginformular (Technisch)	4	4	Soll																																								
				Ist																																								



	Loginhistory sauber umsetzen	2	2	Soll / Ist
	Übersichtsseite Reports (Design)	2	2	Soll / Ist
	Daten für Diagramm aufbereiten	8	8	Soll / Ist
	Diagramm Grafisch darstellen	6	6	Soll / Ist
	Druckoptimierung	2	2	Soll / Ist
	Subdomain einrichten, Applikation aufschalten	2	2	Soll / Ist
	Errorhandling optimieren	2	2	Soll / Ist
K	Kontrollieren	6	6	
	Applikation gründlich testen	6	6	Soll / Ist
A	Auswerten	12	12	
	Reflexion	5	3	Soll / Ist
	Optimierungen	5	3	Soll / Ist
	Dokumentation komplett durchschauen und ergänzen	2	6	Soll / Ist
	Zeit für Unvorhergesehenes	6		
	Total mit Expertenbesuch 1	**80**		
	Dokumentation/Tagesjournal			laufend
	Meilensteine			
	Planung finalisiert			
	Beginn Realisierung			
	Login finalisiert			
	Diagramm lauffähig			
	Produkt lauffähig			
	Testing abgeschlossen			
	Applikation fehlerfrei			
	Abgabe IPA			
	Expertenbesuche	1		

1.9 Expertenbesuche

Datum: Mittwoch, 4. April 2018
Person: Hauptexpertin Ursula Reinhard

Der Expertenbesuch war sehr aufschlussreich. Frau Reinhard hat den Rahmen der IPA aufgezeigt und alle wichtigen Punkte auf einer Präsentation festgehalten. Die wichtigsten Termine wurden aufgezeigt und besprochen. Die Präsentation wird am 2. Mai 2018 in Luzern bei der Leuchter IT Solutions stattfinden.

Zusätzlich wurde der Zeitplan besprochen und Frau Reinhard hat mir einige Tipps zu der Dokumentation gegeben.

1.10 Arbeitsjournal

1.10.1 Tag 1

3. April 2018	Projektphasen: Informieren und Planen
Geplante Arbeiten	**Erledigte Arbeiten**
1. Ordnerstruktur aufbauen, Dokumente erstellen 2. Aufgabenstellung analysieren 3. Informationen bündeln, Management Summary 4. Zeitplan erstellen 5. Meilensteine setzen 6. Grobkonzept (beginnen)	1. Ordnerstruktur aufbauen, Dokumente erstellen 2. Aufgabenstellung analysieren 3. Informationen bündeln, Management Summary 4. Zeitplan erstellen 5. Meilensteine setzen 6. Grobkonzept (beginnen)

Nr	Beschreibung	Probleme	Lösung
1	Ordnerstruktur erstellen, Dokumente erstellen und Versionierung machen	Bei der Formatierung der Dokumentation hatte ich ein paar Schwierigkeiten, wenn man es lange nicht mehr gemacht hat, muss man zuerst wieder herausfinden wie man die Überschriften nummeriert.	Ich habe es gegoogelt und auf der Microsoft Seite eine Lösung[3] gefunden.
2	Aufgabenstellung im PkOrg nochmal genau studiert	-	-
3	Ich habe auf den Informationen im PkOrg das Management Summary geschrieben	-	-
4	Zeitplan mit Excel erstellt	Es ist mir ziemlich schwer gefallen den Zeitaufwand für die Realisierung zu schätzen.	Ich habe die einzelnen Schritte unterteilt und konnte so die Stunden genauer schätzen.
5	Ich habe für mich abgeschätzt welches die wichtigsten Aufgaben sind und dann die Meilensteine gesetzt.	-	-
6	Ich habe zusammen mit Ignaz Walgis die Datenbankstruktur angeschaut, damit ich morgen mit dem Grobkonzept beginnen kann.	-	-

Reflexion:

Ich bin am ersten Tag sehr gut vorwärts gekommen. Ich bin richtig motiviert meine IPA sauber zu Dokumentieren und versuche mich möglichst an den Kriterienkatalog zu halten. Ich merke selber, dass ich gerne schon in der Realisierungsphase wäre, da dass die Phase ist, auf die ich mich am meisten freue. Trotzdem werde ich mich die ersten 3 Tage noch voll auf die Informierungs- und Planungsphase konzentrieren.

[3] Lösung Seitenzahlen unter: https://support.office.com/de-de/article/spaltenbreite-und-zeilenh%C3%B6he-%C3%A4ndern-2c002bd2-3b9c-4561-a99b-cbdcb0f69bc0

1.10.2 Tag 2

4. April 2018	Projektphasen: Planen
Geplante Arbeiten	**Erledigte Arbeiten**
1. Grobkonzept der Webapplikation fertigstellen 2. Systementwurf 3. Testfälle definieren	1. Grobkonzept der Webapplikation fertigstellen 2. Systementwurf 3. Testfälle definieren 4. Doku ergänzt

Nr	Beschreibung	Probleme	Lösung
1	Für alle einzelnen Komponenten ein Lösungsweg definiert, aber ohne den technischen Aspekt sondern nur Grob. Flussdiagramme erstellt um den Programmablauf genau darzustellen.	Die Flussdiagramme haben mich ein bisschen viel Zeit gekostet, da meine Kenntnisse mit Visio nicht überragend sind.	Ich habe im Internet nach Problemlösungen im Visio gesucht und wurde fündig.
2	Systementwurf wurde erstellt, mit Server, Datenbank und der Webapplikation. Die Kommunikation zwischen den einzelnen Komponenten wurde erläutert	Zuvor wurde von mir noch nie eine Systemanalyse erstellt, ich musste mich zuerst informieren.	Ich habe Systemanalysen angeschaut und dann auf mein System übertragen.
3	Ich habe die Anforderungen aus PkOrg genau studiert und dann gebündelt, damit ich daraus nachher ein Testkonzept erstellen konnte.	Nicht alle Anforderungen liessen sich zu einem Input/Output Test konvertieren.	Ich habe 2 Auflistungen gemacht, zum einem die Anforderungen und danach die Testfälle, somit hat man einen klaren Überblick.
4	Ich habe die gesamte Doku nochmal angeschaut und Ergänzungen vorgenommen.		

Reflexion:

Am zweiten Tag bin ich noch voll im Zeitplan. Heute kamen einige unbekannte Aufgaben auf mich zu. Ich hatte zuvor noch nie Testfälle definiert und auch noch nie selber ein Systementwurf gemacht. Schlussendlich bin ich mit meinem Resultat aber zufrieden.

1.10.3 Tag 3

5. April 2018		Projektphasen: Planen und Entscheiden	
Geplante Arbeiten		**Erledigte Arbeiten**	
1. Mockups erstellen und besprechen 2. Technische Möglichkeiten aufzeigen 3. Möglichkeiten vergleichen und entscheiden 4. Grobkonzept der Webapplikation anpassen		1. Mockups erstellen und besprechen 2. Technische Möglichkeiten aufzeigen 3. Möglichkeiten vergleichen und entscheiden 4. Grobkonzept der Webapplikation anpassen	
Nr	**Beschreibung**	**Probleme**	**Lösung**
1	Mockups für die 3 Darstellungen wurden erstellt und mit Geschäftsleitung besprochen	Das erste MockUp Tool was ich benutzt habe, konnte die Datei nicht exportieren ausser man kauft eine Lizenz.	Ich habe es Schlussendlich mit Balsamiq gemacht, da Chili da eine Lizenz besitzt
2	Technische Möglichkeiten wurden aufgezeigt		
3	Ich habe die Möglichkeiten verglichen und dann anhand von Pro und Contra Argumenten entschieden	Es war mir nicht ganz klar wie ich es mit der Code und Kommentarsprache handhaben sollte.	Ich habe es mit meinem Fachvorgesetzten besprochen und dann anhand seiner Angaben entschieden
4	Grobkonzept der Webapplikation wurde mit den Entscheidungen bestückt und ergänzt		

Reflexion

Es war sehr spannend die MockUps zu erstellen, es gingen mir viele neue Ideen durch den Kopf was man alles noch hinzufügen könnte oder was noch cool wäre. Ich habe mir die Sachen notiert, und sobald alle Pflichtaufgaben erfüllt sind, könnte ich noch mit den „Nice to haves“ anfangen.

1.10.4 Tag 4

6. April 2018		Projektphasen: Realisieren	
Geplante Arbeiten		**Erledigte Arbeiten**	
1. Projekt aufsetzen 2. Login designen 3. Login ausprogrammieren		1. Projekt aufsetzen 2. Login designen 3. Login ausprogrammieren	
Nr	**Beschreibung**	**Probleme**	**Lösung**
1	Das Projekt wurde aufgesetzt und die wichtigsten Plugins eingebunden und eine saubere Ordner- und Projektstruktur erstellt		
2	Login desginen, mit HTML und CSS		
3	Login ausprogrammieren, Logik hinzugefügt		

Reflexion

Beim Beginn der Realisierung hatte ich keine Probleme, da ich die meisten Sachen schon mal gemacht habe und deswegen konnte ich sofort loslegen. Trotzdem war es sehr zeitaufwendig, da ich es möglichst sauber und schön umsetzen will.

1.10.5 Tag 5

9. April 2018		Projektphasen: Realisieren	
Geplante Arbeiten		**Erledigte Arbeiten**	
1. Loginhistory sauber umsetzen 2. Übersichtsseite Reports 3. Daten für Diagramm aufbereiten		1. Loginhistory sauber umsetzen 2. Übersichtsseite Reports 3. Daten für Diagramm aufbereiten	
Nr	**Beschreibung**	**Probleme**	**Lösung**
1	Die Loginhistory wurde sauber umgesetzt	IP-Adresse ermitteln	Über das User-Property Wird im Schritt Realisation genauer darauf eingegangen.
2	Übersichtsseite dargestellt		
3	Daten für Diagramm aufbereiten	Es gab viele kleine Probleme die ich aber mittels Google gut lösen konnte. Ich konnte zuerst meinen Webservice nicht über Ajax ansprechen	Das NuGet Web API wurde installiert und richtig konfiguriert, damit ich eine eigene Route für API Calls habe, das ist der saubere Weg und danach hat es geklappt.

Reflexion:

Am fünften Tag habe ich vieles über API-Services in MVC gelernt. Dass man auch eine neue Route definieren kann über die „WebAPI.config“, war mir bisher nicht bewusst. Es ist aber ein schöner weg, weil man die Controlleraufrufe und die API Aufrufe so sauber trennen kann. Es ist immer wieder schön zu sehen, wenn eine Lösung sauber ist und funktioniert.

1.10.6 Tag 6

10. April 2018		Projektphasen: Realisieren	
Geplante Arbeiten		**Erledigte Arbeiten**	
1. Daten für Diagramm aufbereiten 2. Diagramm grafisch darstellen		1. Daten für Diagramm aufbereiten 2. Diagramm grafisch darstellen	
Nr	**Beschreibung**	**Probleme**	**Lösung**
1	Die Daten aus der Datenbank lesen, ins Model schreiben	Model musste erweitert werden, ohne dass das Datenbankmodel betroffen ist, da dieses sich ändern kann und dann überschrieben wird.	Das Datenbankmodel mit einer Teilklasse erweitert. Siehe Realisierung
2	Die Daten mittels API durch Ajax an Javascript geben und dann über das Chart.js darstellen.	Hintergrundfarben richtig darstelllen, damit der aktuelle Monat hervorsticht	Hintegrundfarbe als String-Array genommen statt als String.

Reflexion:

Am sechsten Tag meiner IPA habe ich viele neue Dinge gesehen. Dass man die Datenbankklassen in einer externen Klasse (Teilklasse engl. Partialclass) erweitern kann, wusste ich nicht. Es ist sehr hilfreich, da man somit das Entity-Framework Modell aktualisieren kann und trotzdem die eigenen definierten Properties behält.

Zusätzlich kann man dort auch die Data-Annotations festlegen in einer separaten Klasse. So ist es mir jetzt möglich die generierten Datenbankklassen und meine eigenen Klassenerweiterungen strikt zu trennen.

1.10.7 Tag 7

11. April 2018		Projektphasen: Realisieren und Kontrollieren	
Geplante Arbeiten		**Erledigte Arbeiten**	
1. Diagramm grafisch darstellen 2. Druckoptimierung 3. Applikation aufschalten 4. Applikation gründlich testen		1. Diagramm grafisch darstellen 2. Druckoptimierungen 3. Dokumentation ergänzt 4. Applikation gründlich testen	
Nr	**Beschreibung**	**Probleme**	**Lösung**
1	Diagramm grafisch darstellen	Die Hintergrundfarbe des Diagramms ist bei jeder Säule gleich, so kann ich den aktuellen Monat nicht hervorheben.	Das Datenbankmodel mit einer Teilklasse erweitert. Siehe Realisierung
2	Druckoptimierungen vorgenommen	-	-
3	Applikation aufschalten	Ich habe die Applikation nicht aufgeschaltet, da ich es im Nachhinein sinnvoller finde, die Applikation erst nach dem testen aufzuschalten.	Die Applikation wird erst nach dem IPERKA Schritt „Kontrollieren" aufgeschaltet
4	Applikation wird gründlich getestet	Es gibt im Moment keine Testdaten	Ich erfasse einen Testuser über Chili und trage dort Reports ein über das Access Frontent

Reflexion

Heute musste ich akzeptieren, dass es manchmal sinnvoller ist, etwas um zu planen. Ich werde die Applikation erst nach dem testen auf die Domain aufschalten, da es im Moment noch keinen Sinn macht. Ich kann genauer testen wenn ich es lokal mache. Stattdessen habe ich in dieser Zeit an der Doku weitergeschrieben. Die Dokumentation ist nun wieder auf einem guten Stand und ich kann mich morgen voll dem Testen widmen.

1.10.8 Tag 8

12. April 2018		Projektphasen: Realisieren und Kontrollieren	
Geplante Arbeiten		**Erledigte Arbeiten**	
1. Applikation gründlich testen 2. Applikation aufschalten 3. ErrorHandling optimieren		1. Applikation gründlich testen 2. Applkation aufschalten 3. ErrorHandling optimieren	
Nr	**Beschreibung**	**Probleme**	**Lösung**
1	Applikation gründlich durchtesten, auf allen Browsern, alle möglichen Funktionen	Es gab noch keine realen Testdaten, es waren nur fiktive Daten in der Datenbank	Ich habe einige Daten eines Kunden mit Absprache der Geschäftsleitung in die Datenbank eingetragen.
2	Applikation auf Server aufgeschaltet	-	-
3	Globales Errorhandling implementiert.		

Reflexion

Ich habe nun noch 2 Tage Zeit die ganze Applikation zu Reflexieren und zu optimieren. Ich kann mir nochmal genügend Zeit nehmen um die Dokumentation zu vervollständigen. Es gibt noch einige spannende Sachen in der Realisierung die ich bisher noch nicht dokumentiert habe, das werde ich nun noch nachholen.

1.10.9 Tag 9

13. April 2018		Projektphasen: Auswerten	
Geplante Arbeiten		**Erledigte Arbeiten**	
1. Reflexion 2. Optimierungen (Code und Doku)		1. Reflexion 2. Optimierungen (Code und Doku)	
Nr	**Beschreibung**	**Probleme**	**Lösung**
1	Reflexion geschrieben und ganzes Projekt reflexiert	-	-
2	Programmcodekommentare überarbeitet und ergänzt Dokumentation überarbeitet und ergänzt.	-	-

Reflexion

Die Dokumentation hat noch grosses Verbesserungspotential das ich maximal ausschöpfen möchte. Ich bin motiviert die Dokumentation noch mal zu überarbeiten, damit ich am letzten Tag die Dokumentation gewissenhaft ausdrucken kann.

Der Code ist nun sauber kommentiert, alle Methoden sind klar kommentiert was sie machen. Jedes File hat einen Programmkopf der das Erstelldatum, den Ersteller und den Zweck der Datei angibt. Das macht einen sauberen Eindruck und steigert die Wartbarkeit des Codes.

1.10.10 Tag 10

16. April 2018		Projektphasen: Auswerten	
Geplante Arbeiten		**Erledigte Arbeiten**	
1. Dokumentation komplett durchschauen und ergänzen		1. Dokumentation komplett durchschauen und ergänzen	
Nr	**Beschreibung**	**Probleme**	**Lösung**
1	Dokumentation komplett durchschauen und ergänzen, Bewertungskriterien nochmal genau durchgegangen		

Reflexion

Am letzten Tag meiner IPA habe ich nochmal die gesamte Dokumentation durchgeschaut und habe mir die Bewertungskriterien nochmal angesehen und auf meine Dokumentation bezogen geprüft.

Danach habe ich mit dem Drucken begonnen und habe die Version auf PKOrg hochgeladen.

Teil 2 des IPA Berichts

Projektdokumentation

Enthält die ganze Umsetzung des Projekts vom Schritt Planen bis zum Auswerten

2 Gliederung

Der zweite Teil des Berichts wird strikt nach meiner Projektmanagement-Methode gegliedert. Auf diese gehe ich im Kapitel 3.2.1 ein.

3 Informieren

In der Projektphase „Informieren" werden die Schritte die ich unternommen habe bevor ich mit dem nächsten Schritt „Planung" angefangen habe erläutert.

Für die IPA wird die Projektmanagement Methode IPERKA verwendet, auf die im Punkt 3.2.1 nochmal genauer eingegangen wird.

3.1 Management Summary

Kurzfassung der Situation

Ausgangssituation

Chili führt AdWords Kampagnen für Kunden durch. Damit die Kunden einen Überblick darüber haben, ob und wie erfolgreich ihre Kampagne verläuft, erhalten die Kunden monatlich einen Report. Dieser Report wurde bisher als PDF von Chili erstellt und dann über E-Mail zugeschickt.

Es soll mit dieser IPA eine Webapplikation erstellt werden, die diesen mühsamen manuellen PDF-Schritt erleichtert. Die Daten werden über ein Access Frontend in die SQL Datenbank geschrieben (nicht Teil der IPA) und sollen dann von der Webapplikation ausgelesen und mit einem Diagramm dargestellt werden.

Der Kunde kann die Webapplikation aufrufen, sich einloggen und seine Reports anschauen.

Umsetzung:

Es soll eine moderne Webapplikation auf ASP.NET MVC programmiert werden. Diese Applikation soll unter der Subdomain „customers.chili.ch" laufen und ansprechend aussehen.

Der User hat die Möglichkeit sich einzuloggen und seine AdWords-Berichte anzuschauen.

Die AdWords Berichte werden mittels eines Diagramms dargestellt. Ein AdWords-Bericht enthält für jeden Monat diese Kennzahlen: Verweise, Suchmaschinen, Sonstige, Traffic Vorjahr, Absprungraten Durchschnittliche Verweildauer und Besuche.

Der Kunde hat die Möglichkeit zwischen den verschiedenen Monaten zu wählen und sich diese als Diagramm anzeigen zu lassen.

Zu erwartendes Ergebnis:

- Eine öffentlich aufrufbare Website die auf unserem Server läuft
- Ein funktionierendes Login für unsere Kunden
- Saubere und fehlerfreie Darstellung des Diagramms der AdWords-Daten
- Möglichkeit den AdWords-Bericht auszudrucken mit einem sauberen Print-CSS.
- Benutzerfreundliches Oberflächendesign

3.2 Projektverwaltung

3.2.1 Projektmanagementmethode

IPERKA – Die 6-Schritte-Methode

Als Projektmanagementmethode habe ich mich für IPERKA entscheiden. IPERKA wird auch an den überbetrieblichen Kursen oft angewendet und ich habe damit bereits gearbeitet. IPERKA gliedert sich in 6 Projektphasen die folgend erläutert werden.

Informieren	Was genau ist mein Auftrag?
Planen	Welche Lösungsmöglichkeiten habe ich
Entscheiden	Für welchen Lösungsweg entscheide ich mich
Realisieren	Wie gelingt mir die Umsetzung?
Kontrollieren	Habe ich den Auftrag fachgerecht ausgeführt?
Auswerten	Was gelang mir gut, was weniger?

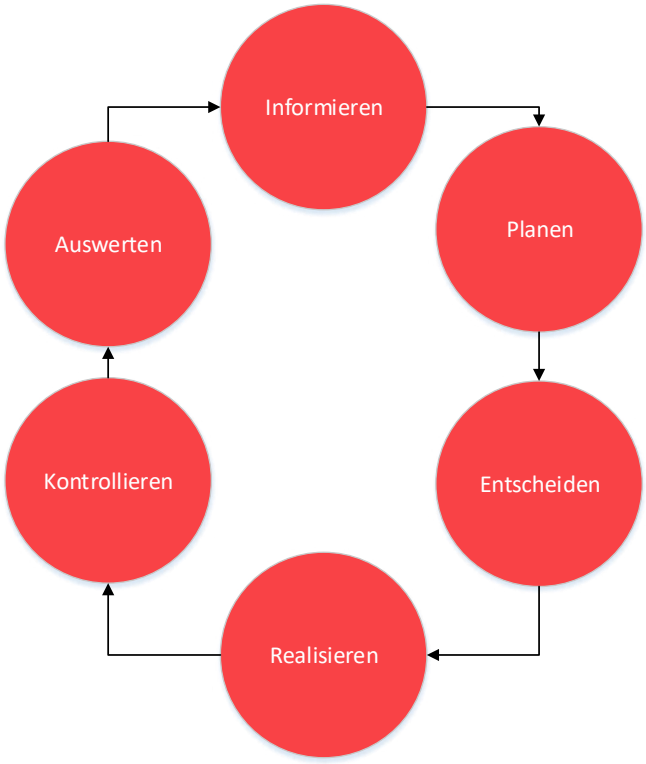


Abbildung 3 Schema IPERKA

Diese 6 Schritte befolgt man bei der IPERKA Methode. Für einen Entwicklungsschritt einer Software macht man immer einen Durchlauf durch diese 6 Schritte. Und für jedes Update das danach kommt, geht man nochmal alle Schritte durch. In meiner IPA wird genau 1 Durchlauf der IPERKA Methode gemacht.

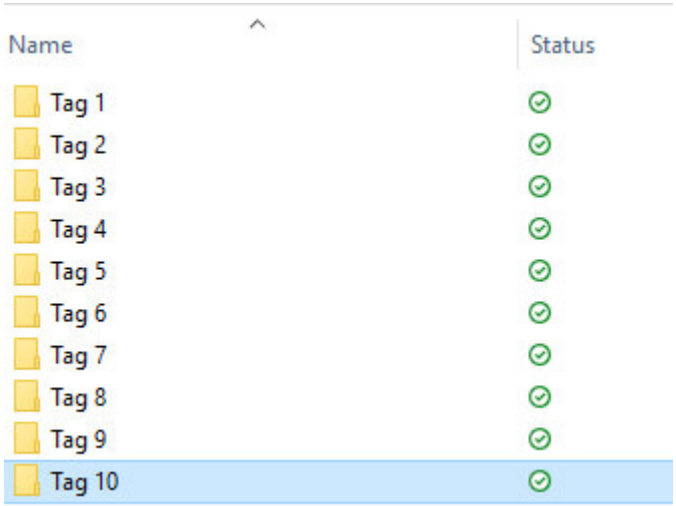
Entscheidungskriterium

Wieso wurde gerade IPERKA als Methode bei der IPA gewählt?

- In den überbetrieblichen Kursen und bei anderen Projekten habe ich jeweils mit IPERKA gearbeitet und kenne mich damit aus.
- Die IPERKA-Methode legt besonderen Wert auf eine genaue Planung, was bei der IPA auch verlangt ist und somit ist es optimal geeignet.
- Mit IPERKA besteht eine klare Struktur durch das ganze Projekt.
- Die einzelnen Projektabläufe sind klar gegliedert und im Zeitplan kann ich die Arbeiten optimal auf die verschiedenen Phasen aufteilen.

3.2.2 Ordnerstruktur und Backups

Die Ordnerstruktur wird auf OneDrive verwaltet. OneDrive ist ein Filehosting Dienst von Microsoft. Die Ordnerstruktur beinhaltet alle wichtigen Dokumente der IPA. Jeden Tag gibt es einen neuen Ordner mit der jeweiligen Version, somit habe ich immer Zugriff auf alte Dokumente, falls mal irgendwas schief läuft.



Ich mache jeden Abend noch ein BackUp als .ZIP und lade es auf Google Drive hoch. So habe ich 2 separate Cloud Dienste (OneDrive und GoogleDrive) und so eine doppelte Sicherung.

3.2.3 Versionsverwaltung der Dokumentation

Nach dem Inhaltsverzeichnis ist ein Versionsjournal zu finden, welches die Änderungen die in der jeweiligen Version gemacht wurden festhält.

3.2.4 Codeverwaltung der Software

Bei Chili sind wir im Moment am Umstellen von VSS (Visual SourceSafe) auf Visual Studio Team Services. Visual Studio Teamservices ist in Visual Studio integriert und ist eine Cloud-Verwaltung mit Git. Man kann alle Änderungen separat veröffentlichen und hat danach eine saubere Versionierung.

In der IPA wird mit Visual Studio Team Services gearbeitet. Seit meiner Ausbildung bei Chili arbeite ich mit Team Services und bin sehr zufrieden, deshalb setze ich bei meiner IPA auch auf Visual Studio Team Services.

Alte Versionen vom Code können jederzeit zurückgeholt und angeschaut werden.

Genaue Erklärung zu **Visual Studio Team Services** im Kapitel 6.1.2.

4 Planen

In dieser Phase findet die gesamte Planung des Projekts statt. Zum einen geht es um das zeitliche Planen und zum anderen um das Planen der technischen Umsetzung

4.1 Zeitplanung

Um einen klaren Überblick über die Zeitreserven zu bekommen, ist es in der IPA vorgeschrieben einen Zeitplan zu führen. Der Zeitplan ist im Kapitel 1.8 im ersten Teil der IPA zu finden.

4.1.1 Strukturierung

Die verschiedenen Aufgaben wurden in die 6 Phasen von IPERKA eingeteilt. Jede Aufgabe hat eine Soll Planung. Eine Soll Planung bedeutet, dass definiert wird wie viele Stunden man für diese Aufgabe aufwenden darf.

Alle Stunden zusammengezählt ergeben 80 Stunden, das ist der Rahmen in dem diese IPA durchgeführt wird.

Die Dokumentation meiner IPA habe ich als „laufend" definiert. Die Dokumentation wird bei jeder Aufgabe nachgeführt und laufend ergänzt. Änderungen in der Dokumentation sind im Versionsverlauf zu finden.

4.1.2 Meilensteine

Die wichtigsten Punkte meiner Arbeit sind im Zeitplan als Meilensteine definiert. Das sind zum einem die IPERKA-Phasen abzuschliessen und wichtige Punkte bei der Realisierung wie zum Beispiel „Login finalisiert" und „Diagramm lauffähig". So ist der Überblick über den Stand der Arbeit stets gewährleistet.

Die geplanten Meilensteine sind einzuhalten. Bei Änderungen wird das in der Dokumentation ersichtlich sein.

4.2 Systementwurf

4.2.1 Zielsystem

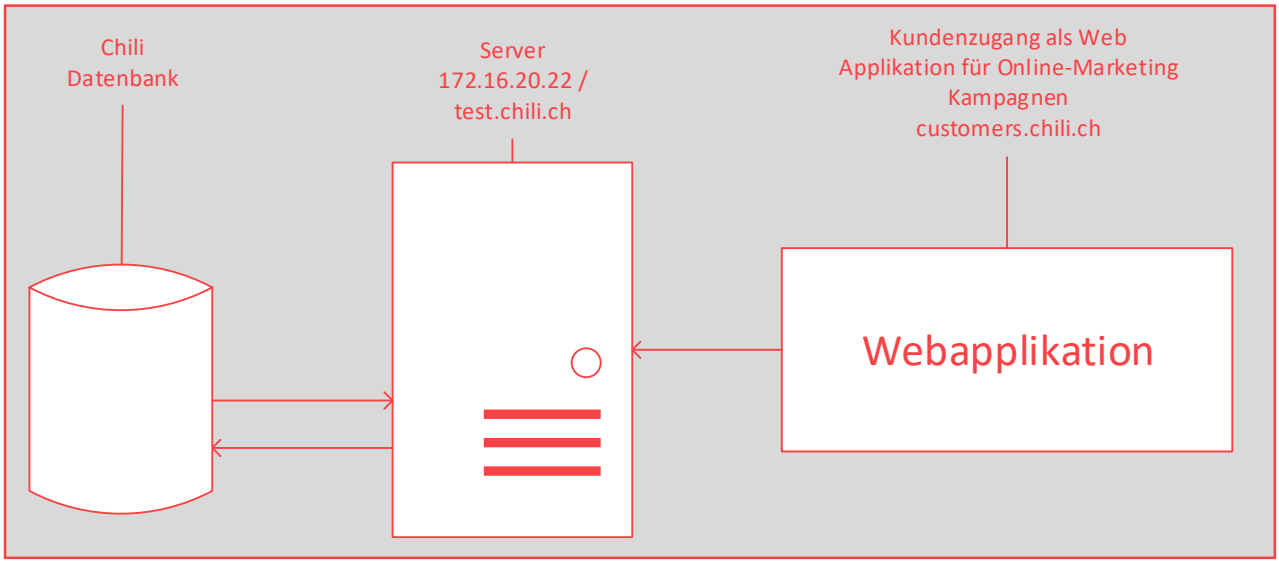


Abbildung 4 Zielsystem

Das Zielsystem ist die Applikation selber. Diese Applikation läuft auf dem Server von Chili. Auf diesem Server gibt es eine Datenbank (im Schema Chili Datenbank). Die Applikation läuft dann unter customers.chili.ch und kann so übers Web aufgerufen werden.

Die Applikation schickt Anfragen an den Server, der Server verbindet sich mit der Datenbank und gibt der Applikation einen Rückgabewert in Form eines oder mehreren Datensätzen.

4.2.2 Grobkonzept

Der Kunde soll sich übers Web einloggen können und kommt dann auf eine Übersichtsseite über seine Berichte. Wenn auf ein Bericht geklickt wird, dann wird man weitergeleitet auf eine Inhaltsseite wo man das Diagramm und alle nötigen Daten zum aktuellen Report sieht. Von dort aus kann man auf andere Reports wechseln oder wieder zurück zu der Gesamtübersicht gehen.

Lösung Loginform

Es wird zwei Inputfelder geben wo Kundenummer und Passwort eingegeben werden. Beim Klick auf den Login Button werden die 2 Felder validiert und dann dem User eine Rückmeldung gegeben.

Bei der Validierung dieser Felder gibt es mehrere Möglichkeiten. Auf diese wird in der Projektphase „Entscheiden" eingegangen und Vor- und Nachteile angeschaut und anhand von diesen entschieden.

Für das Loginformular wird auch ein Flussdiagramm erstellt, dieses ist im Kapitel der Workflows zu finden.

Lösung Loginhistory

Immer wenn der Login Button gedrückt wird, wird ein neuer Datensatz in die Historytabelle geschrieben. Ich muss den User Agent ausfindig machen und dann in die Datenbank schreiben. Der Useragent wird benötigt um die History zurückzuverfolgen und bei eventuellen Unstimmigkeiten den Fehler auf dem richtigen Browser ausfindig zu machen.

Lösung Reportübersichtsseite

Auf der Reportsübersichtsseite werden alle Reports des eingeloggten Kunden aufgelistet. Diese sind nach Datum sortiert, der neueste Report kommt oben links und alle anderen folgen.

Lösung zwischen einzelnen Reports umschalten in einem Report

Der Kunde soll die Möglichkeit haben zwischen den einzelnen Reports ganz simpel hin und her zu wechseln.

Dazu werde ich 2 Pfeile einbauen mit „zum nächsten Monat" und „zum vorherigen Monat". Diese Pfeile erscheinen aber nur dann, wenn auch wirklich ein nächster oder vorheriger Report in der Datenbank vorhanden ist.

Lösung Diagramm

Für das Diagramm wird ein jQuery Plugin verwendet. Welches genau benutzt wird, wird in der nächsten Projektphase „Entscheiden" entschieden. Ich werde einige Diagrammmethoden vergleichen und mich für eine entscheiden.

Lösung Druckfähig

Es wird ein separates Stylesheet verwendet, welches nur im Druckmodus aktiv wird. Unnötige Elemente werden ausgeblendet und Kopf und Fusszeile werden hinzugefügt, damit der Ausdruck sauber aussieht.

Grafische Lösungen

Im Kapitel GUI Konzepte sind die Entwürfe für die grafische Oberfläche der einzelnen Lösungen zu finden.

4.2.3 Workflows

Flussdiagramm Login

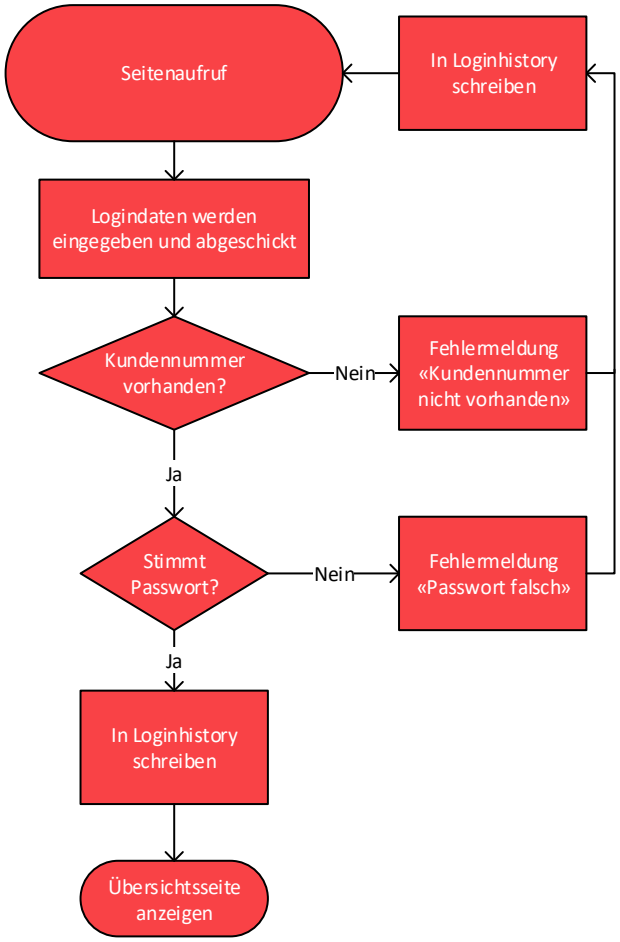


Abbildung 5 Flussdiagramm Login

Flussdiagramm Reportübersichtseite

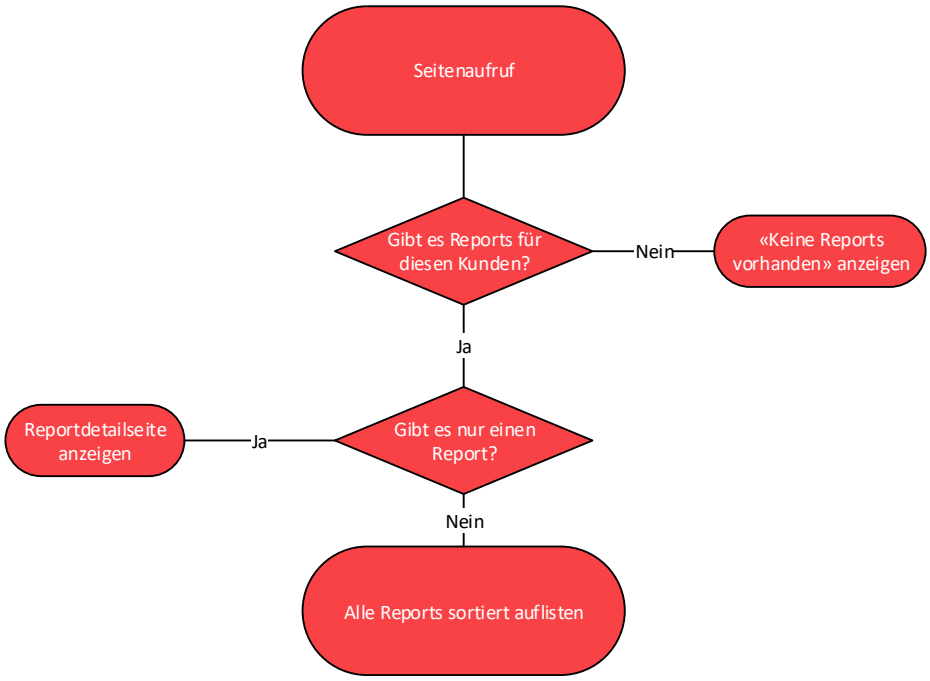


Abbildung 6 Flussdiagramm Reportübersichtseite

Flussdiagramm Reportseite

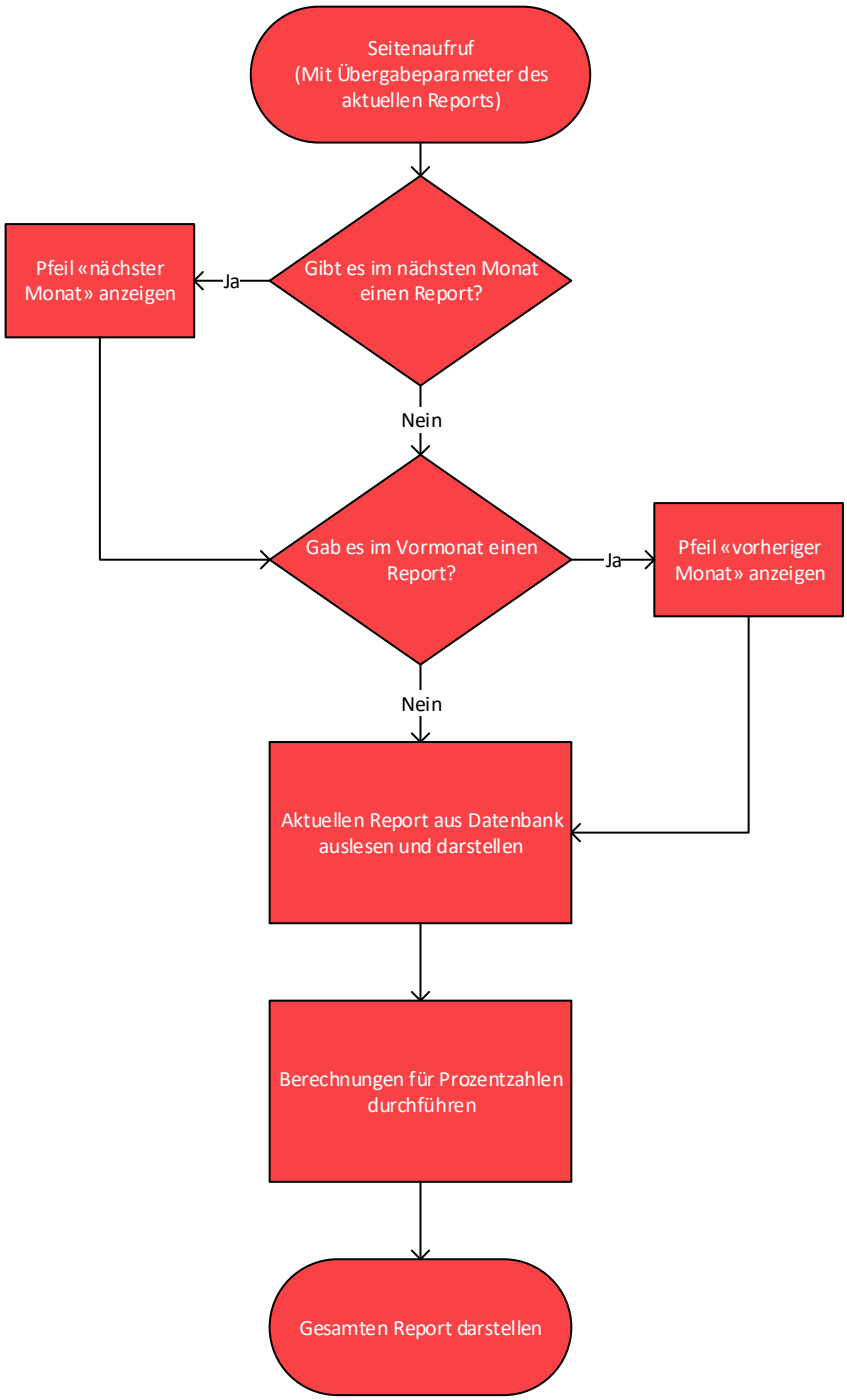


Abbildung 7 Flussdiagramm Reportdetailseite

4.2.4 Datenbankanalyse

Teile der Datenbank bestehen schon lange und sind Teil unseres Chili-CRM. Alle Marketing Tabellen wurden vorgängig vor der IPA von der Geschäftsleitung erstellt und mit Beispieldaten befüllt.

Ein Überblick über die Tabellen mit Primär- und Fremdschlüsseln. (In Visio erstellt für den Überblick)

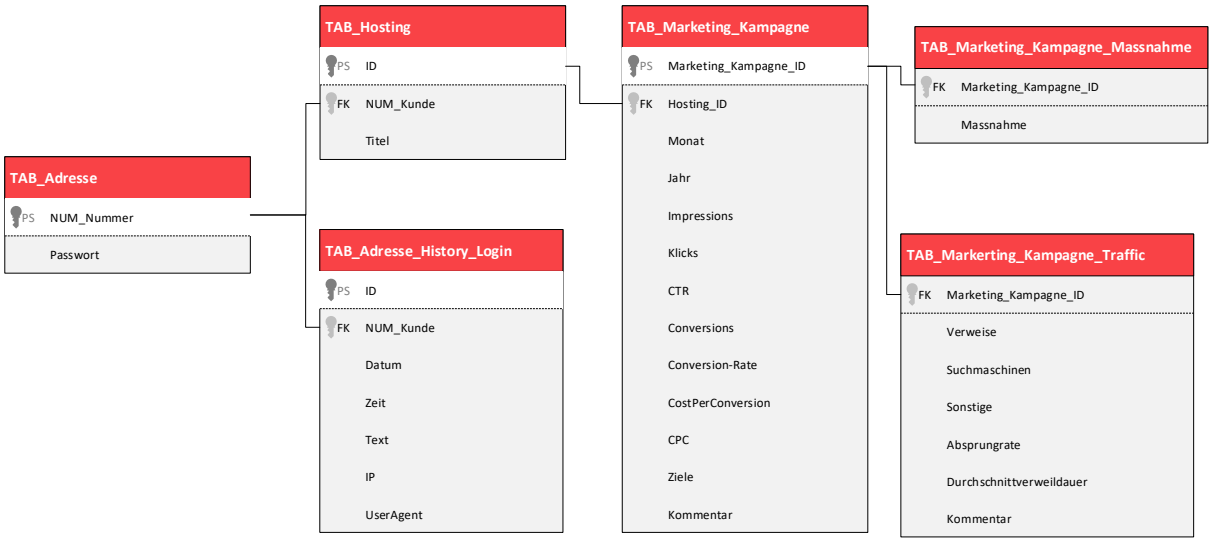


Abbildung 8 Datenbankschema

TAB_Adresse

In dieser Tabelle sind alle unsere Kunden erfasst. Jeder Kunde hat eine Kundennummer und falls vorhanden ein Passwort. Bei der Realisierung muss ich noch regeln, was passiert wenn kein Passwort gesetzt ist.

TAB_Adresse_History_Login

Jedes Mal wenn sich ein Kunde über die Webapplikation versucht einzuloggen, wird ein neuer Eintrag in diese Tabelle hinzugefügt.

Ist über die Kundennummer (NUM_Kunde) mit TAB_Adresse verbunden.

TAB_Hosting

In dieser Tabelle sind Verträge erfasst. Ein Kunde kann mehrere Verträge haben. Für die IPA ist der Vertrag vom Typ „Marketing" interessant, denn genau diese Kunden, die einen solchen Vertrag haben, können sich schlussendlich in die Webapplikation einloggen.

Ist über die Kundennummer (NUM_Kunde) mit TAB_Adresse verbunden.

TAB_Marketing_Kampagne

Ein Marketingvertrag (aus Tabelle TAB_Hosting) kann mehrere Kampagnen beinhalten. Eine Kampagne ist immer ein Monat eines Jahres. Hat der Kunde zum Beispiel einen Jahresvertrag dann wird es 12 Marketing-Kampagnen geben.

Kampagnen sind über die Hosting_ID mit der Tabelle Hosting verbunden.

TAB_Marketing_Kampagne_Massnahme

Eine Kampagne kann mehrere Massnahmen enthalten. Jede Massnahme hat einen Massnahmentext, also eine Änderung die vorgenommen wird, um die laufende Kampagne zu verbessern.

Massnahmen sind über die Marketing_Kampagne_ID mit der Kampagne verbunden.

TAB_Marketing_Kampagne_Traffic

Jede Kampagne hat Traffic, das sind z.B Aufrufe durch Verweise, Suchmaschinen oder Sonstiges. Dieser Traffic wird in dieser Tabelle gespeichert.

Der Traffic ist über die Marketing_Kampagne_ID mit der Kampagne verbunden.

4.2.5 GUI Konzepte

Mockup 1 Startseite / Loginseite

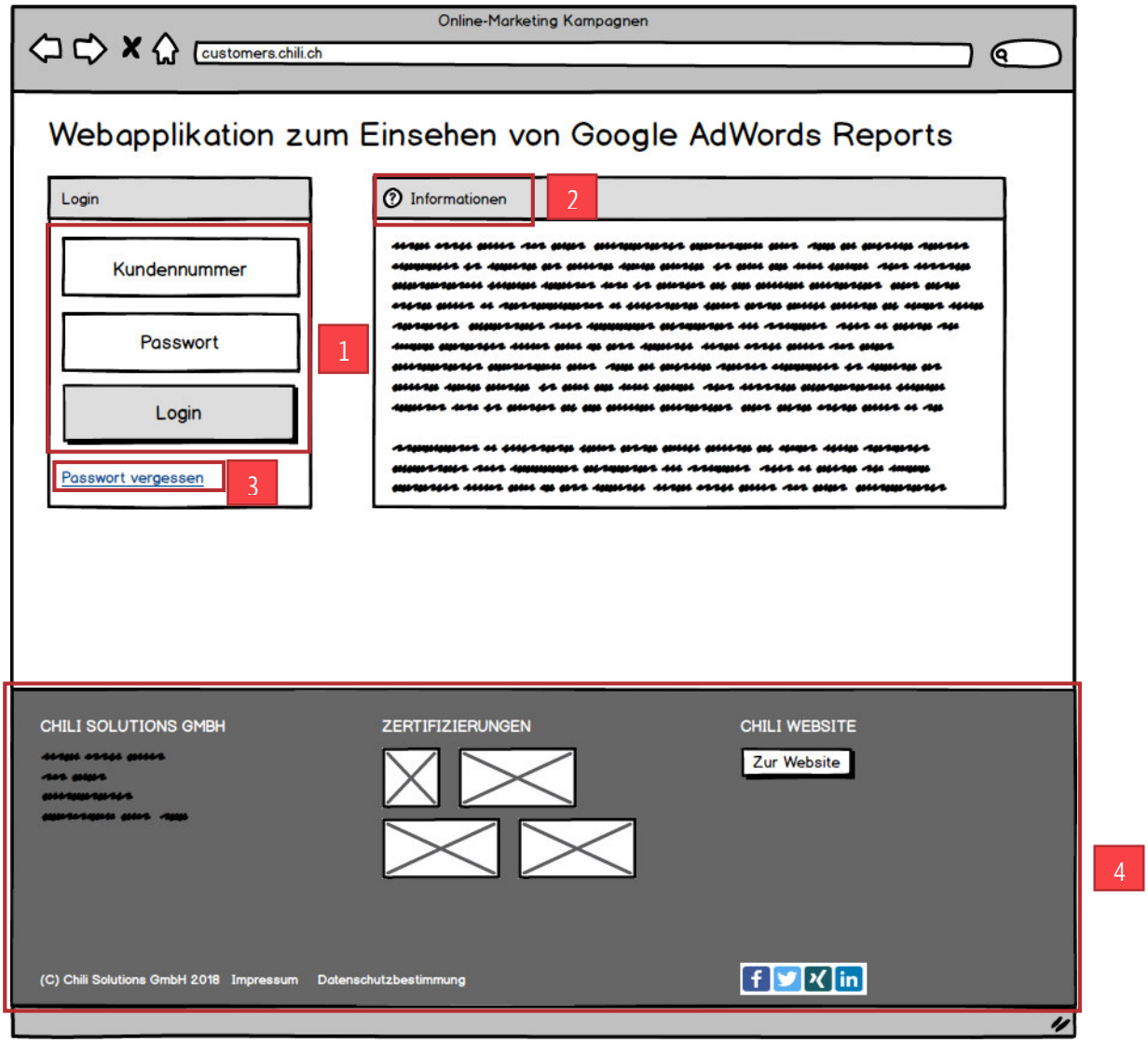


Abbildung 9 Mockup Startseite

Erklärung

1) **Eingabefelder:** Das sind die Eingabefelder für den Kunden und der Login Button, um seine Eingaben prüfen zu lassen. Bei Fehlern werden diese immer unterhalb des Betroffenen Elements angezeigt. Ist also die Kundennummer falsch wird direkt unter „Kundennummer" eine Fehlermeldung ausgegeben.
2) **Informationen (optional):** Hier können Informationen über die Applikation stehen
3) **Passwort vergessen Funktion (optional):** Falls ich noch übrige Zeit habe, würde ich noch eine Passwort vergessen Funktion einbauen
4) **Footer:** Der Footer wird am Layout der Chili Website angelehnt sein. Zusätzlich wird auf der rechten Seite eine Verlinkung zu unserer Unternehmenswebsite zu finden sein.

Mockup 2 Reportübersichtsseite

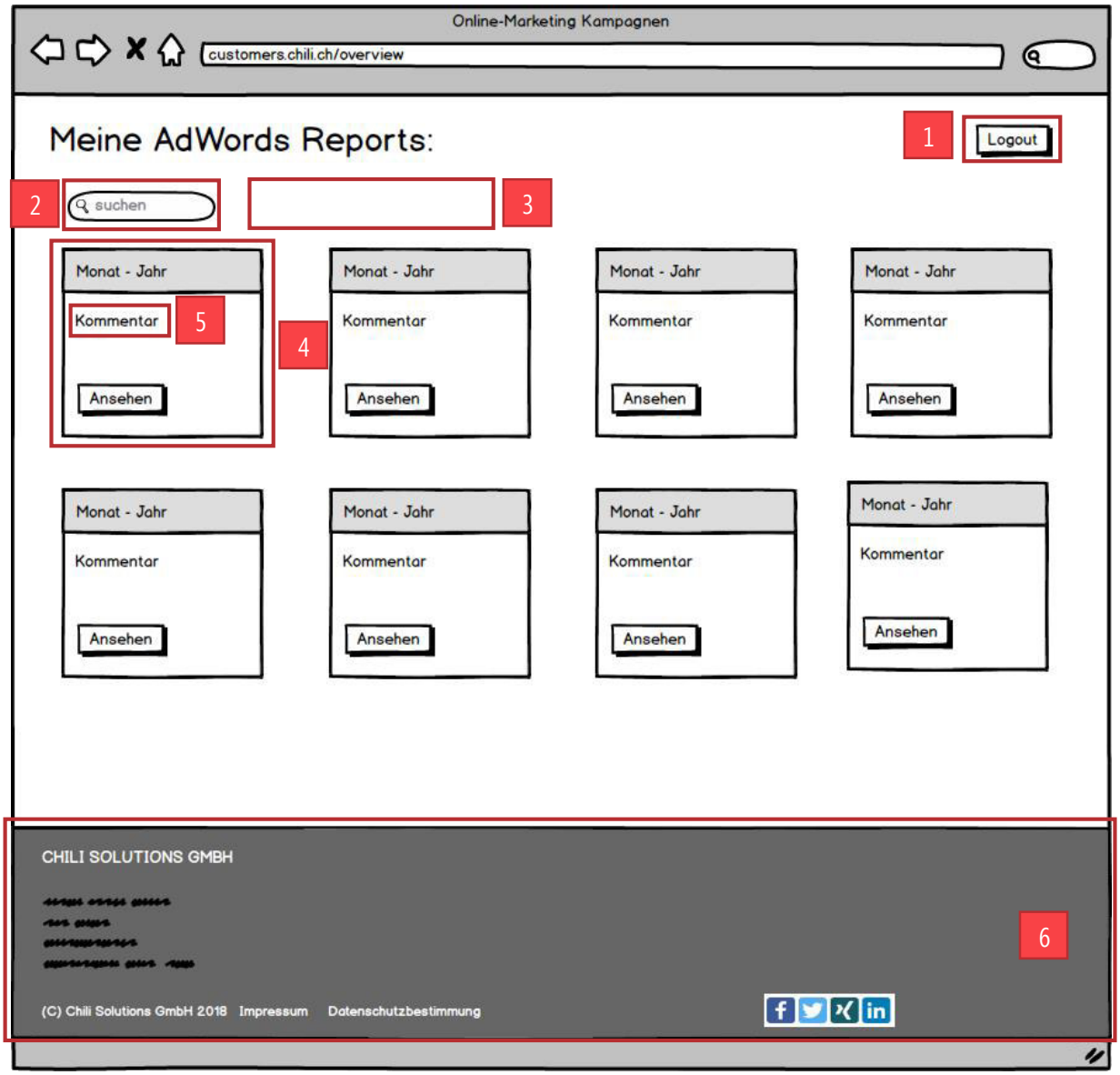


Abbildung 10 Mockup Reportübersichtsseite

Erklärung

1) **Logoutbutton:** Beim Klick auf diesen Button wird die Session beendet und der Kunde ausgeloggt.
2) **Suchfeld:** Im Suchfeld kann der Kunde nach Begriffen suchen wonach dann die einzelnen Reports gefiltert werden. Er kann zum Beispiel „April" eingeben und kann so den Monat April der verschiedenen Jahre vergleichen.
3) **Sortierungsbuttons (optional):** Dieser Platz könnte für Sortierungsbuttons verwendet werden, damit man die Reports aufsteigend/absteigend oder nach sonstigen Kennzahlen sortieren kann.
4) **Reportauflistung:** Es werden alle Reports aufgelistet und nach Datum sortiert. Jeder Report erhält ein eigenes Kästchen.
5) **Kommentar:** Hier kommen die wichtigsten Kennzahlen eines jeden Reports und es wird der Domainname des Reports dargestellt (diese Eingaben können im Suchfeld gefiltert werden)
6) **Footer:** Auf dieser Seite wird der Footer etwas schlänker dargestellt, die Zertifizerung und die Verlinkung fallen weg.

Mockup 3 Reportseite

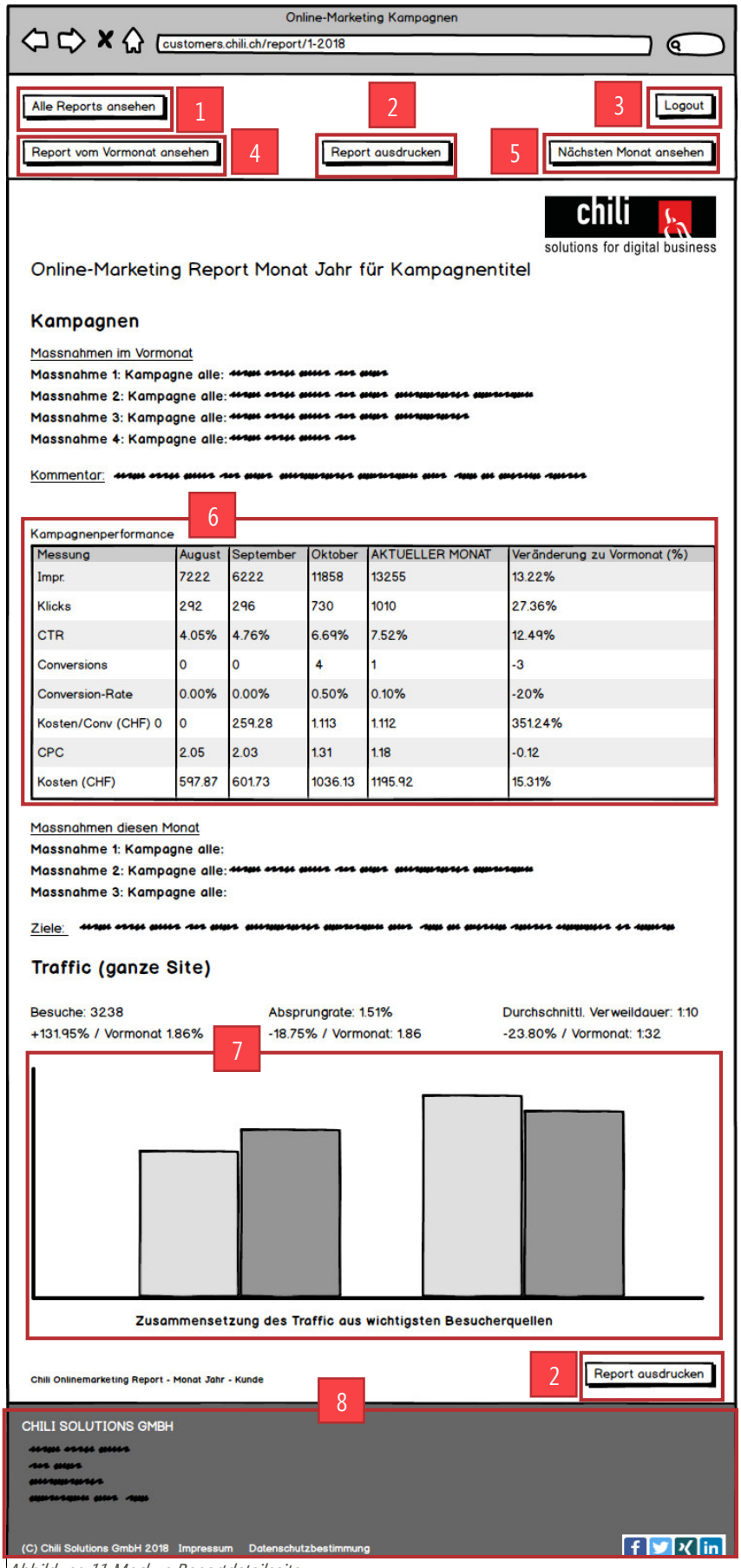


Abbildung 11 Mockup Reportdetailseite

Erklärung zum MockUp auf der Folgeseite.

Erklärung

1) **Alle Reports ansehen:** Beim Klick wird man zur Reportsübersichtsseite (Mockup 2) geleitet.
2) **Report ausdrucken:** Beim Klick auf diesen Button wird die Website gedruckt (mit optimiertem Print.css), damit nur das nötige auf dem Ausdruck ist.
3) **Logoutbutton:** Beim Klick auf diesen Button wird die Session beendet und der Kunde ausgeloggt.
4) **Report vom Vormonat ansehen:** Beim Klick auf diesen Button wird man zum Report des vorherigen Monates weitergeleitet. Dieser Button erscheint aber nur wenn es auch wirklich einen Report im Vormonat gegeben hat.
5) **Report vom nächsten Monat ansehen:** Beim Klick auf diesen Button wird man zum Report des nächsten Monates weitergeleitet. Dieser Button erscheint nur, wenn es im Folgemonat auch wirklich einen Report gibt.
6) **Tabelle mit Kampagnendaten:** In dieser Tabelle werden alle Kampagnendaten dargestellt. Es werden maximal 7 Monate gleichzeitig angezeigt. Der aktuelle Monat soll optisch erkennbar sein und rechts dargestellt sein. Links davon dürfen maximal 6 vergangene Monate angezeigt werden. Sind weniger als 6 vergangene Reports vorhanden, dann werden weniger angezeigt.

   Der Titel einer Tabelle ist immer verlinkt mit dem jeweiligen Report des Monats. Ganz rechts sind die Kennzahlen mit dem Vormonat im Vergleich ersichtlich.
7) **Diagramm über den Traffic:** In diesem Diagramm werden die Traffic-Daten visualisiert. Es werden wie in der Tabelle maximal 7 Monate und mindestens der aktuelle Monat angezeigt. Der aktuelle Monat ist ebenfalls visuell vorgehoben.

   Die einzelnen Werte sind über die Farbe erkennbar und können voneinander unterschieden werden.
   Das Diagramm enthält neben dem Säulendiagramm der Trafficdaten (Suchmaschinen, Verweise, Sonstige) auch noch eine Linie, die den Traffic des Vorjahres darstellt.
8) **Der Footer wird schlänker gehalten als auf der Loginseite.**

4.3 Testkonzept

Hier werden die Testfälle definiert, die bei der Auswertung der Applikation getestet werden. Es macht Sinn diese Testfälle im Voraus zu definieren, damit man bei der Planung und Realisierung besonders auf die Testfallspezifikationen achtet.

Das **Umfeld der Tests** ist die entwickelte Applikation.

4.3.1 Testmethode

Es wird die White-Box-Testmethode benutzt.[4] Es wird eine Ausgangslage angeben und ein zur erwartendes Ergebnis. Diese Testmethode leitet die Testfälle aus dem Programm selbst ab und nicht wie das Black-Box Testing aus der Programmspezifikation. Die Testfälle werden von Hand ausgeführt.

4.3.2 Anforderungsanalyse

Die messbaren Anforderungen wurden aus der Detaillieren Aufgabenstellung Kapitel 1.2.2 erstellt.

Messbare Anforderungen

Da nicht alle diese Anforderungen mit einem Input / Output Verfahren Messbar sind habe ich nun die Messbaren Anforderungen auseinandergenommen und nummeriert aufgelistet:

1. Seite ist öffentlich über die Adresse customers.chili.ch erreichbar
2. Seite darf nicht von Suchmaschinen indexiert werden
3. Applikation soll in den aktuellsten Versionen von IE, Firefox und Chrome sauber dargestellt werden.
4. Der Kunde soll sich mit Kunden-Nr. und Passwort einloggen können.
5. Die Loginhistory soll korrekt geführt werden.
6. SQL Injection muss verhindert werden.
7. Es soll automatisch nach 15 Minuten ausgeloggt werden
8. Der Kunde kann Reports via Browser auf 1 A4 ausdrucken
9. Der Kunde kann sich ausloggen

Messbare Anforderungen Report

10. Titel lautet: Online-Marketing Report *Monat Jahr* für *Kunde*
11. Massnahmen der Kampagne im Vormonat in Listenform sind ersichtlich
12. Kommentar zur aktuellen Kampagne ist ersichtlich
13. Tabelle mit Titel „Kampagnenperformance“ und folgenden Kennzahlen vertikal: Impressions, Klicks, CTR, Conversions, Conversion-Rate, Kosten/Conv. (CHF), CPC, Kosten (CHF). Vertikal werden die Monate (max. 6 Monate zurück) und die Veränderung zum Vormonat in % dargestellt. Der aktuelle Monat ist visuell erkennbar.
14. Massnahmen der Kampagne des aktuellen Vormonates in Listenform ist ersichtlich
15. Ziele der aktuellen Kampagne sind ersichtlich
16. Tabelle mit Titel „Traffic (ganze Site)“ mit folgenden Kennzahlen:
    Besuche aktueller Monat und Vormonat zum Vergleich in absoluten Zahlen und Veränderung in Prozent.
    Absprungrate in % aktueller Monat und Vormonat zum Vergleich in absoluten Zahlen und Veränderung in Prozent.
    Durchschnittliche Verweildauer aktueller Monat und Vormonat zum Vergleich in absoluten Zahlen und Veränderung in Prozent.
17. Balkendiagramm mit Titel „Zusammensetzung des Traffic aus wichtigsten Besucherquellen“ wird korrekt dargestellt
18. Fusszeile: „chili Onlinemarketing Report – Monat Jahr – Kundenname“ ist korrekt dargestellt.

[4] Quelle White-Box-Testmethode: https://de.wikipedia.org/wiki/White-Box-Test

4.3.3 Testfallspezifikationen

Nr.	Input	Output
1	Die Seite customers.chili.ch wird aufgerufen	Das Login Formular wird angezeigt
2	Auf Suchmaschine nach Seite suchen, Metatags überprüfen	Seite darf nicht gelistet sein
3	Auf den neusten Browsern (IE, Firefox und Chrome)testen	Applikation muss fehlerfrei sein
4	Kundennummer und Passwort wird eingegeben und Login Button wird gedrückt	Die Übersichtseite der Reports wird angezeigt
5	Login Button wird gedrückt	History wird korrekt geführt und User Agent ist richtig eingetragen
6	SQL Injection ausprobieren	Kunden Nr. oder Passwort falsch wird angezeigt
7	Keinen Input machen	Nach 15 Minuten ausgeloggt werden
8	Druck-Button wird gedrückt	A4 wird fehlerfrei ausgedruckt
9	Logout-Button wird gedrückt	Kunde wird ausgeloggt, Session wird geleert und Kunde wird weitergeleitet auf die Login Seite.

Spezifikationen Report

Nr.	Input	Output
10	Reportseite wird aufgerufen	Titel wird korrekt angezeigt
11	Reportseite wird aufgerufen	Liste wird korrekt dargestellt und alle Informationen sind vorhanden
12	Reportseite wird aufgerufen	Kommentar wird korrekt angezeigt
13	Reportseite wird aufgerufen	Kennzahlen sind vorhanden und korrekt. Der Aktuelle Monat ist grafisch hervorgehoben
14	Reportseite wird aufgerufen	Massnahmen sind in Listenform ersichtlich und korrekt
15	Reportseite wird aufgerufen	Ziele sind ersichtlich und korrekt dargestellt
16	Reportseite wird aufgerufen	Tabelle wird korrekt dargestellt und enthält die richtigen Daten
17	Reportseite wird aufgerufen	Balkendiagramm wird korrekt dargestellt
18	Reportseite wird aufgerufen	Fusszeile ist vorhanden und korrekt dargestellt

5 Entscheiden

In der Phase „Entscheiden" geht es um die Entscheidung mit welchen Produkte und technischen Massnahmen die IPA umgesetzt werden soll. Es wird auf verschiedene Möglichkeiten eingegangen und danach wird Anhand von Kriterien entscheiden, welche Methode schlussendlich gewählt wird.

5.1 Produkte ohne Entscheidungsspielraum

Einige Produkte sind von oder Aufgabestellung oder von der Infrastruktur her vorgegeben und können nicht selber ausgewählt werden.

Abbildung 12 Visual Studio Logo

Entwicklungsumgebung

Wenn man mit .NET programmiert, kommt man nicht um **Visual Studio** herum. In meiner IPA werde ich Visual Studio 2017 benutzen, da dies die aktuellste Version ist und ich mich damit auskenne. Als Programmierhilfe wird das Tool „ReSharper" benutzt, um den Code möglichst sauber zu halten und die Programmierung zu erleichtern.

Datenbank

Die Chili Datenbank läuft auf MSSQL, ist also ein **Microsoft SQL Server**, der bei uns selber im Betrieb steht und von uns gewartet wird. Also wird im Rahmen der IPA auch auf MSSQL gearbeitet und ich kenne mich damit aus.

Abbildung 13 Microsoft SQL Server

Abbildung 14 Microsoft IIS

Webserver

Während der Entwicklungszeit wird die Applikation auf dem IIS Express auf meinem eigenen Computer laufen (über Visual Studio). Sobald die Applikation veröffentlicht wird, läuft die Software auf einem Windows Server 2012 mit Internet Information Services. (IIS)

Programmiersprache

Die Programmiersprache ist auch schon gegeben. Sie wurde im Detailbeschrieb festgesetzt und ist auch ein Teil der Bewertung (Individuelle Bewertungskriterien). Es wird C# auf ASP.NET MVC programmiert.

Abbildung 15 Microsoft C# .NET

Abbildung 16 Microsoft Logo

Das sind alles Produkte die von **Microsoft** entwickelt wurden. Chili ist eine .NET Firma, es wird möglichst mit .NET programmiert, denn in dieser Sparte ist die nötige Erfahrung vorhanden. Natürlich wird auch auf OpenSource Produkte gesetzt, wenn es Sinn macht.

5.2 Passworthashing

Ausgangslage:

Das Passwort des Kunden wird manuell über ein Access Frontend von der Geschäftsleistung gesetzt. Das Passwort wird momentan in Klartext erfasst und wird so in die Datenbank geschrieben

Möglichkeiten:

1. Passwort gehasht erfassen und gehashte Eingaben mit DB vergleichen
2. Passwort in Klartext in der DB lassen und Klartext Eingaben mit DB vergleichen

Nr.	Pro	Contra
1	• Sicherheit steigt • Heutiger Standard • Datenschutz der Nutzer ist gewährleistet	• Zeitaufwand für Geschäftsleitung • Passwort kann nicht an Kunde per Klartext gesendet werden • Wenn Kunde Passwort vergisst muss ein neues gesetzt werden
2	• Es muss nichts angepasst werden	• Sicherheit

Entscheid mit Begründung:

Ich habe es mit der Geschäftsleitung besprochen und es wird vorerst in der DB als Klartext gespeichert bleiben. (Variante 2) Es könnte nach der IPA noch geändert werden, aber es ist keine Anforderung an meine IPA, dass das Passwort gehasht sein muss.

5.3 Feldervalidierung

Ausgangslage:

In meiner Webapplikation gibt es Input Felder die validiert werden müssen und dem Kunden muss eine Fehlermeldung zurückgegeben werden.

Möglichkeiten:

1. HTML 5 Required Attribut benutzen
2. Eigene Clientseitige Validierung schreiben mit jQuery
3. MVC Data Annotations benutzen

Nr.	Pro	Contra
1	• Schnell implementiert	• Errormessage nicht anpassbar • Nicht unter IE 10 supportet • Kann nur für „benötigt" benutzt werden
2	• Grosse Anzahl von Möglichkeiten • Beliebig Anpassbar • Anpassbarkeit (mittelmässig)	• Zeitaufwand (gross)
3	• Anpassbarkeit (sehr gut) • Sauberer Code • Wird direkt im Model definiert • Erfahrung darin	• Zeitaufwand (mittelmässig)

Entscheidung mit Begründung:

Ich habe mich für die Variante 3 (MVC Data Annotations) entschieden, da ich damit schon Erfahrung habe und es einfach die sauberste Lösung ist. Es ist auch gut, wenn jemand mal etwas am Code anpassen muss, es muss nur das Model angepasst werden.

5.4 Diagramme

Ausgangslage:

In der Webapplikation werden Diagramme benötigt, um Daten aus der Datenbank darzustellen. Dazu gibt es verschiedene Möglichkeiten die ich aufzeigen möchte.

Möglichkeiten:

1. amCharts (jQuery)
2. Chart.js
3. Google Charts
4. Chart Helper Klasse von .NET

Nr.	Pro	Contra
1	• Support gewährleistet • Viele Chartmöglichkeiten • Wird ständig erweitert	• Kostenpflichtig • Keine Animationen
2	• OpenSource • Grosse Community • Schöne Animationen • Gut dokumentiert	
3	• Viele Charts • Gut dokumentiert	• An eine Firma gebunden • Keine Animationen
4	• Serverseitige Erstellung eines Bildes • Kompatibel auf alle Browser	• Nur wenige Chartmöglichkeiten • Designtechnisch eher altmodisch • Keine Animationen

Entscheid mit Begründung

Ich habe mich für Variante 2 entschieden, da dieses Plugin gut dokumentiert ist und schöne Animationen möglich sind. Ausserdem gibt es das Chart dass ich für meine Arbeit brauche, und somit ist es optimal.

5.5 Sprache des Codes und der Kommentare

Ausgangslage

Die Chili Datenbank ist auf Deutsch gehalten. In den Chili Programmierrichtlinien ist folgendes definiert:
„Bevorzugte Sprache ist Englisch. Die Verwendung von anderen Sprachen ist erlaubt und muss im Rahmen des jeweiligen Projektes definiert werden. Es sind aber innerhalb eines Projektes keine Mischformen erlaubt."

Möglichkeiten:

1. Deutscher Code
2. Englischer Code

Nr.	Pro	Contra
1	• Code ist mit Datenbank einheitlich • Somit kein Sprachenmix	• Nicht so international, Code kann nur von deutschsprachigen gewartet werden
2	• Internationalität	• Verstösst gegen Programmierrichtlinien, da Datenbank und Code nicht übereinstimmen würden

Entscheidung mit Begründung

Das Projekt wird in Deutsch codiert und kommentiert. Dann ist es einheitlich mit der Datenbank. Zusätzlich entspricht es so den Programmierrichtlinien der Chili Solutions GmbH. Ich habe das Thema mit dem Fachvorgesetzten kurz besprochen und darf selber entscheiden, welche Sprache gewählt wird. Somit gibt es keine Vorgabe der Geschäftsleitung.

5.6 Useragent erkennen

Ausgangslage:

Bei jedem Loginversuch wird ein Eintrag in die History Tabelle gemacht. Damit ich diesen Eintrag machen kann, benötige ich den User Agent (Betriebssystem, Browser des Nutzers). Dieser lässt sich auf verschiedene Weisen herausfinden.

Möglichkeiten:

- Clientseitiger UserAgent Check (mit Plain jQuery)
- Clientseitiger UserAgent Check (mit Modernizr.js)
- Serverseitiger UserAgent Check (HttpRequest.UserAgent Property)

Nr.	Pro	Contra
1	• Performance	• Clientseitig ist immer von der Sicherheit her schlechter als Serverseitig
2	• Sehr zuverlässiges Plugin	• Grosses Skript das eingebunden wird • Performance
3	• Sicherheit • Geschwindigkeit • Genauigkeit • Leicht wartbar	

Entscheidung mit Begründung:

Ich habe mich für das Serverseitige HttpRequest.UserAgent Property entschieden, da dies die sauberste Lösung ist und auch später einfach gewartet werden kann.

5.7 Sortieren der Reports auf der Übersichtsseite

Ausgangslage

Auf der Reportübersichtsseite sind viele Reports aufgelistet. Es wäre schön wenn man die Möglichkeit hat diese Reports zu filtrieren oder zu sortieren. Es gäbe ein Input Feld und dort könnte man nach Stichwörtern suchen.

Möglichkeiten:

- Sortierung weglassen
- Sortierung mit jQuery selber schreiben
- Sortierung mit jQuery Plugin Isotope

Nr.	Pro	Contra
1	• Kein Aufwand	• Wenn viele Reports vorhanden sind, schlechte Übersicht
2	• Lerneffekt • Sortierung wäre performant und auf das nötigste reduziert	• Zeitaufwand
3	• Vorkenntnisse vorhanden • Schöne Animationen • Viele Möglichkeiten	• Grosses Skript das eingeunden wird • Performance

Entscheidung mit Begründung

Ich habe mich bei der Sortierung für das jQuery Plugin Isotope entschieden, da ich dieses von anderen Projekten kenne und es sehr anpassungsfähig ist und optisch ansprechende Animation beinhaltet.

5.8 Überarbeitetes Grobkonzept

Lösung Loginform

Es wird 2 Felder geben (Kundennummer und Passwort). Zum Abschicken des Formulars gibt es einen „Login" Button.

Die Validierung wird mit **MVC Data Annotations** erfolgen, so können die Fehlermeldungen direkt im Model definiert werden und können einfach angepasst werden.

Lösung Loginhistory

Sobald der Loginbutton gedrückt wird, soll die IP, der UserAgent und ob das Einloggen erfolgreich war in die Datenbank geschrieben.

Den UserAgent werde ich über das **HttpRequest.UserAgent Property** abfragen und dann in die Datenbank schreiben.

Lösung Reportübersichtsseite

Auf der Reportsübersichtsseite wird es eine Sortierung geben. Es gibt ein Textfeld wo man einen Suchbegriff eingeben kann und es wird dann nach Kommentaren und Massnahmen gefiltert. Diese Filtrierung wird mit dem jQuery Plugin **Isotope** erstellt.

Lösung Diagramm

Die Diagramme auf der Reportseite werden mit dem jQuery Plugin **Chart.js** realisiert. Es wird maximal 7 Säulen geben auf einer Reportseite, da dies so in der Detailbeschreibung der IPA vorgegeben ist.

6 Realisierung

In der Realisierungsphase wird Schritt für Schritt dokumentiert wie das Projekt umgesetzt wird. Zwischendurch werden Unterthemen wie „MVC" oder „Visual Studio Team Services" genauer erklärt.

6.1 Projekt aufsetzen

Als erster Schritt wird das Projekt in Visual Studio erstellt.

Erstellung

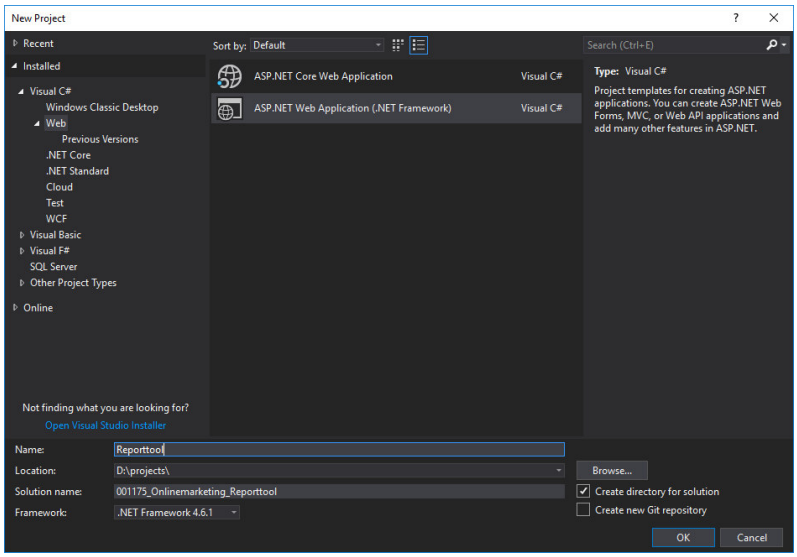


Abbildung 17 Screenshot Applikation erstellen

Es wird eine neue ASP.NET Webapplikation erstellt und zwar mit dem Solutionname 001175_Onlinemarketing_Reporttool

Diese Namensgebung ist durch die Firma gegeben. Projekte haben alle eine identifizierende Projektnummer und diese ist immer 6 stellig. Anschliessend folgt der Projektname.

Der Name selber ist „Reporttool". Das ist der Name der Hautapplikation. Im Verlauf der Realisierung werden 2 weitere Projekte folgen: Klassenbibliothek und Datenzugriff.

Als Projekttemplate kommt eine leere ASP.NET Applikation in Frage mit Referenzen auf das MVC Pattern.

MVC Pattern wird im Kapitel 6.1.1 erläutert.

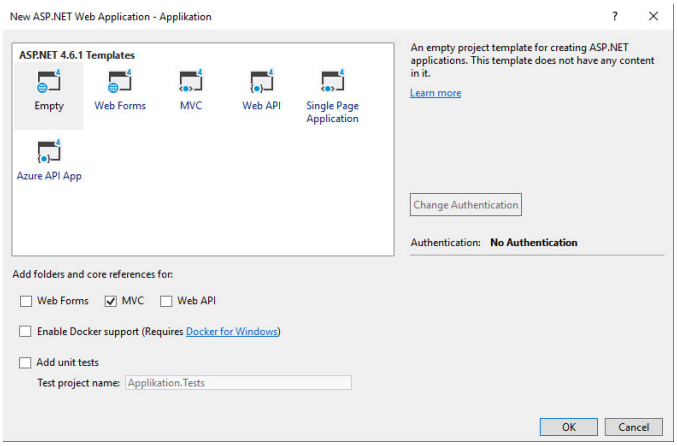


Abbildung 18 Screenshot Template auswählen

Versionsverwaltung einrichten

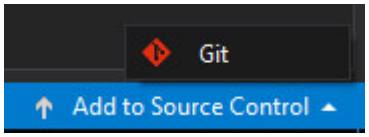


Abbildung 19 Screenshot Source Control

Sobald das Projekt erstellt wurde erscheint unten rechts ein Button „Add to Source Control". Wenn dieser angeklickt wird, wird ein Git Repository erstellt.

Git wird im Kapitel 6.1.2 erklärt.

Repository veröffentlichen

Das soeben erstellte Repository muss nun noch veröffentlicht werden. Im Rahmen der IPA wird mit Visual Studio Team Services gearbeitet.

Visual Studio Team Services wird im Kapitel 6.1.2 erläutert.

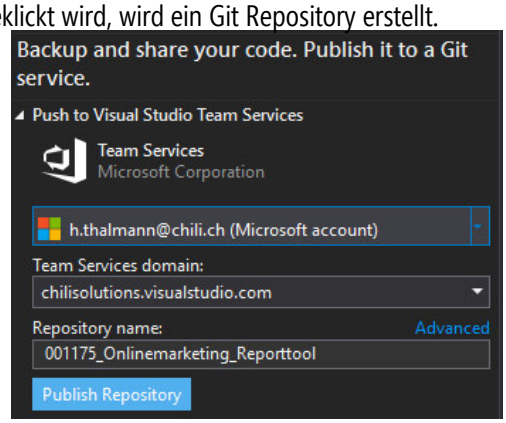


Abbildung 20 Screenshot Publish Repository

6.1.1 MVC

Abbildung 21 .NET MVC Logo

ASP.NET MVC ist ein Framework für die Microsoft .Net Umgebung. MVC steht für Model-View-Controller, ein Pattern, das für die Trennung der Logik, der Daten und der Präsentation verantwortlich ist.

Model: Das Model beinhaltet die Daten. Das Model ist nicht zwingend die Datenbank selbst, es ist eine Klasse, die die Struktur der Daten enthält.

In der IPA werden Models aus der Datenbank generiert mit dem Entity Framework, auf dieses wird in einem späteren Zeitpunkt eingegangen.

View: Die View ist die Anzeige. Sie bildet die Daten aus dem Model sowie Eingabefelder für die Benutzerinteraktionen ab.

Die Views beinhalten bei der IPA den HTML Teil. Ebenfalls wird in den Views die Razor Syntax von C# angewendet, um auf Elemente des Models zuzugreifen.

Controller: Der Controller verbindet den Controller und die View. Er nimmt Übergabewerte an, verarbeitet und berechnet diese, fügt sie ins Model ein und übergibt dann das befüllte Model an die View, wo diese Daten dann dargestellt werden. Im Controller ist die gesamte Logik zu finden. Logik kann mithilfte von Klassen ausgelagert werden.

Durch diese Trennung können auch die Teile wieder eigenständig verwendet werden und es wird ein klarer Überblick geschaffen.

6.1.2 Versionsverwaltung Visual Studio Team Services[5]

Abbildung 22 Visual Studio Team Services Logo

Visual Studio Team Services ist ein Angebot von Microsoft, mit dem sich kollaborativ Anwendungen per Cloud Computing entwickeln lassen. Es erschien 2013 zunächst unter dem Namen Visual Studio Online und als Nachfolger des zuvor von Microsoft angebotenen Team Foundation Service. Enthalten sind Möglichkeiten für das Application Lifecycle Management, darunter Versionskontrolle und Build-Management. Im November 2015 wurde die Umbenennung in Visual Studio Team Services bekannt gegeben. Es handelt sich um eine in der Cloud gehostete Version von Team Foundation Server.

[5] Quelle Team Services: https://de.wikipedia.org/wiki/Visual_Studio

GIT

Visual Studio Team Services agiert mit GIT. GIT ist eine freie Software zur Versionsverwaltung von Dateien.

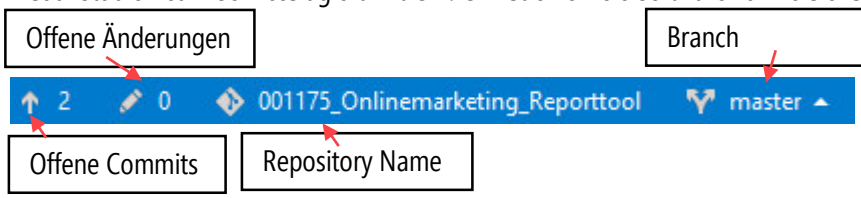
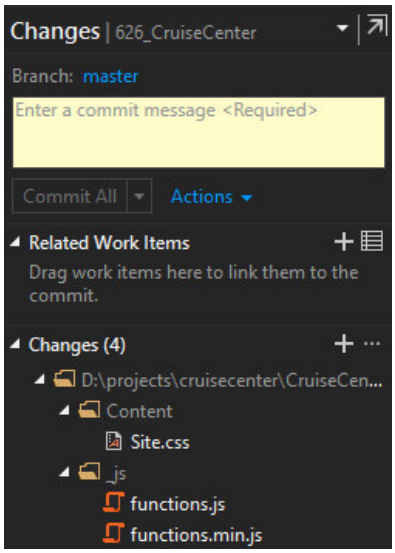


Abbildung 23 Screenshot Git Funktionen in VSTS



Changes

Sobald man Änderungen am Projekt vornimmt, werden diese als Changes dargestellt. Klickt man unten bei Visual Studio auf diese Changes erhält man eine Übersicht der Änderungen.

Nach jeder Änderung die man macht sollte man einen sogenannten „Commit" schreiben und dort festhalten, was man am Programm geändert hat.

Wenn man diesen dann mit dem Cloud-Status synchronisiert (mittels PUSH oder SYNC), werden die 2 Versionen verglichen und dann zusammengeführt (mittels MERGE).

Gibt es Komplikationen mit anderen Codestellen, wenn zum Beipsiel jemand anderes noch am Code arbeitet, dann kann man die Codestellen vergleichen und dann entscheiden welcher Code dass genommen werden soll.

Abbildung 24 Screenshot changes

Wenn man diese Commits immer sauber nach jeder Änderung macht, dann erhält man eine saubere Versionsstruktur wie folgt:

Abbildung 25 Screenshot Commits

Falls dann später neue Leute ins Projekt kommen, können Sie sich gut einlesen und haben einen sauberen Ablauf der Programmierung vor sich.

Gitignore:

Es werden grundsätzlich alle Dateien in einem Ordner beachtet. Mittels einer .gitignore Datei kann man gewisse Dateien von der Versionsverwaltungt auschliessen. Das können zum Beispiel .dll Dateien sein, die sowieso jedesmal neu erstellt werden und nichts mit der eigentlichen Entwicklung zu tun haben.

6.2 Datenbankzugriff

In der IPA wird der Datenbankzugriff mit dem Entity Framework realisiert.

Abbildung 26 Entity Framework Logo

6.2.1 Entity Framewok Erklärung

Es gibt im Entityframework mehrere Modelierungsansätze. In der IPA besteht die Datenbank bereits und deswegen wird im Rahmen der IPA mit dem Database-First Model.

Entity Framework geniert so aus der bestehenden Datenbank ein Model und ein Datenbanken Kontext, welcher die verschiedenen Tabellen aus der Datenbank als Model darstellt. Dies ermöglicht einen Zugriff auf die bestehenden Daten.

Wenn man Änderungen an der Datenbankstruktur vornimmt muss man dann Model im Visual Studio aktualisieren und es wird neu aus der Datenbank ausgelesen. Wenn es aber Änderungen an den Daten in der Datenbank gibt, wird dies natürlich in Echtzeit übertragen.

6.2.2 Entity Framework nutzen

Ich trenne meine Projekte gerne in Unterprojekte. Die Webapplikation ist unter dem Namen „Reporttool“ zu finden. Das Datenbankmodell mit Entity Framework wird im Projekt „Data“ gemacht.

Man fügt also der Projektmappe eine neue „Class Library“ hinzu.

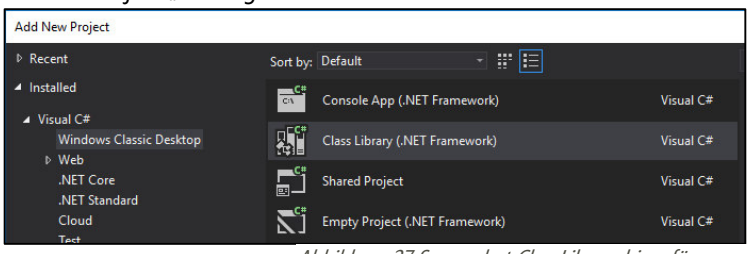
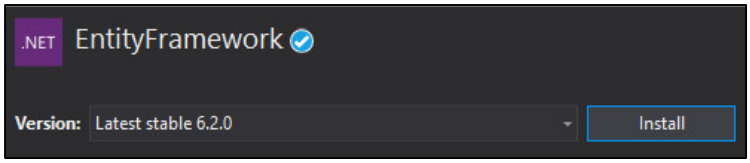
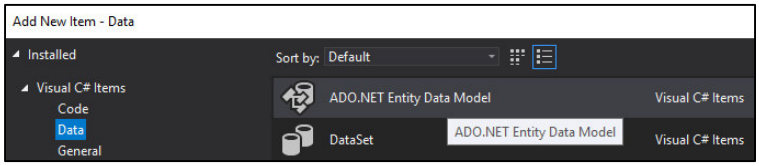


Abbildung 27 Screenshot ClassLibrary hinzufügen



Danach fügt man bei dem soeben erstellten Projekt das NuGet Packet „EntityFramework“ hinzu.

Abbildung 28 Screenshot Entity Framework hinzufügen



Zuletzt wird dann noch ein neues Data Model zum Projekt „Data“ hinzugefügt. Dort wird die Verbindung zur Datenbank aufgebaut.

Abbildung 29 Screenshot Data Model hinzufügen

Im Modus „EF Designer from database“ wird das Database-First Model gewählt. Somit werden die Daten aus der Datenbank zu einem Model generiert.

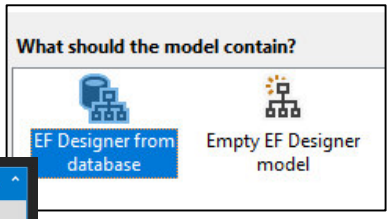


Abbildung 30 Screenshot Modelstruktur

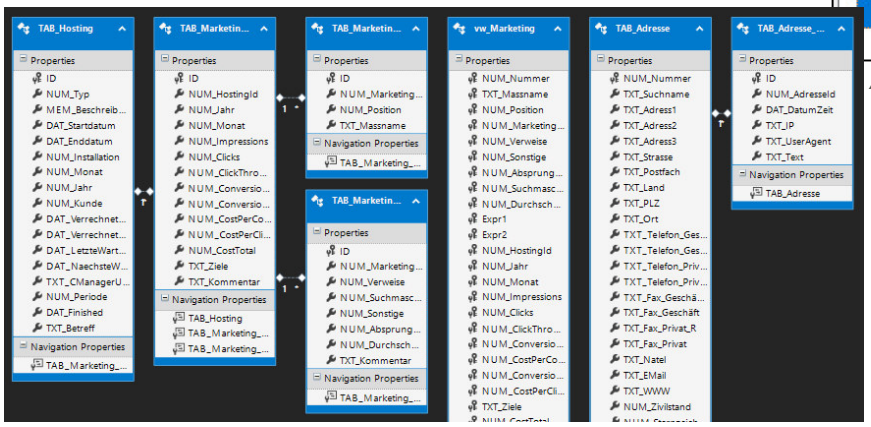


Abbildung 31 Screenshot Datenbankmodel in Visual Studio

Danach gibt man die Verbindungsdaten der Datenbank ein, wählt alle Tabellen und Views aus die man für das Projekt benötigt und klickt dann auf bestätigen.

Das Entity Framework generiert nun alle Models und man erhält eine Übersicht:

6.2.3 Bisherige Ordnerstruktur:

Ein sauber strukturiertes Projekt ist wichtig, um den Überblick zu behalten. Deshalb hier die erste Version der bisherigen Ordnerstruktur.

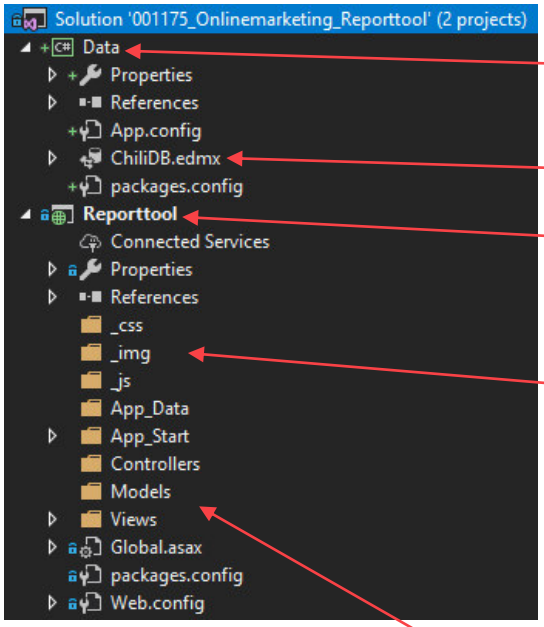


Abbildung 32 Screenshot Ordnerstruktur

Das Projekt welches für den Datenbankzugriff zuständig ist

Das Datenbank-Model

Dieses Projekt beinhaltet die Webapplikation an sich

Diese Ordner beinhalten die Stylesheets, Javascript Files und Bilder

In diesen Ordner sind die 3 Schichten der MVC Modellierung zu finden.

Views:
Ansichten der Daten (CSHTML)

Models
Datenmodellierung (zusätzlich zum Datenbankmodel)

Controllers
Enthalten die Logik, füllen das Model mit Daten und übergeben es an die View

Weitere Schritte

Es können später noch weitere Projekte folgen, es wird sicher noch eine ClassLibrary geben, damit ich redundanten Code auslagern kann. Code auslagern macht Sinn wenn er öfters benutzt wird. Es dient auch dazu die Controller schlank zu halten damit man auch bei viel Logik noch eine Übersicht hat über den Code.

Reflexion Erstellung des Projekts

Ich habe nun viel Zeit in die Erstellung des Projekts gesteckt, finde es aber sinnvoll wenn man das Projekt gleich von Anfang an sauber aufzieht, damit man danach in Ruhe arbeiten kann.

Jetzt geht's weiter an die Programmierung und ich bin sehr motiviert eine saubere und schöne Lösung umzusetzen.

6.3 Phase 1 Login

Nachdem das Projekt nun mal grundsätzlich erstellt ist, wird mit dem Login begonnen. Das Login ist zugleich auch die Startseite der Applikation. Ein MockUp ist im Kapitel 4.2.5 unter „Mockup Startseite / Loginseite" zu finden.

6.3.1 Die HTML Seite

Als erstes baue ich die HTML Seite und werde erst danach mit der Logik die Daten füllen.

Bootstrap

Ich benutze für meine HTML-Templates das CSS-Framework Bootstrap.

Bootstrap teilt die Bildschirmgrösse in Grids ein, also prozentuale Einteilungen. Die neuste Version ist die Version 4 und ist vor kurzem erst heraus gekommen.

Abbildung 33 Bootstrap Logo

Die Website eingeteilt ins Bootstrap Grid

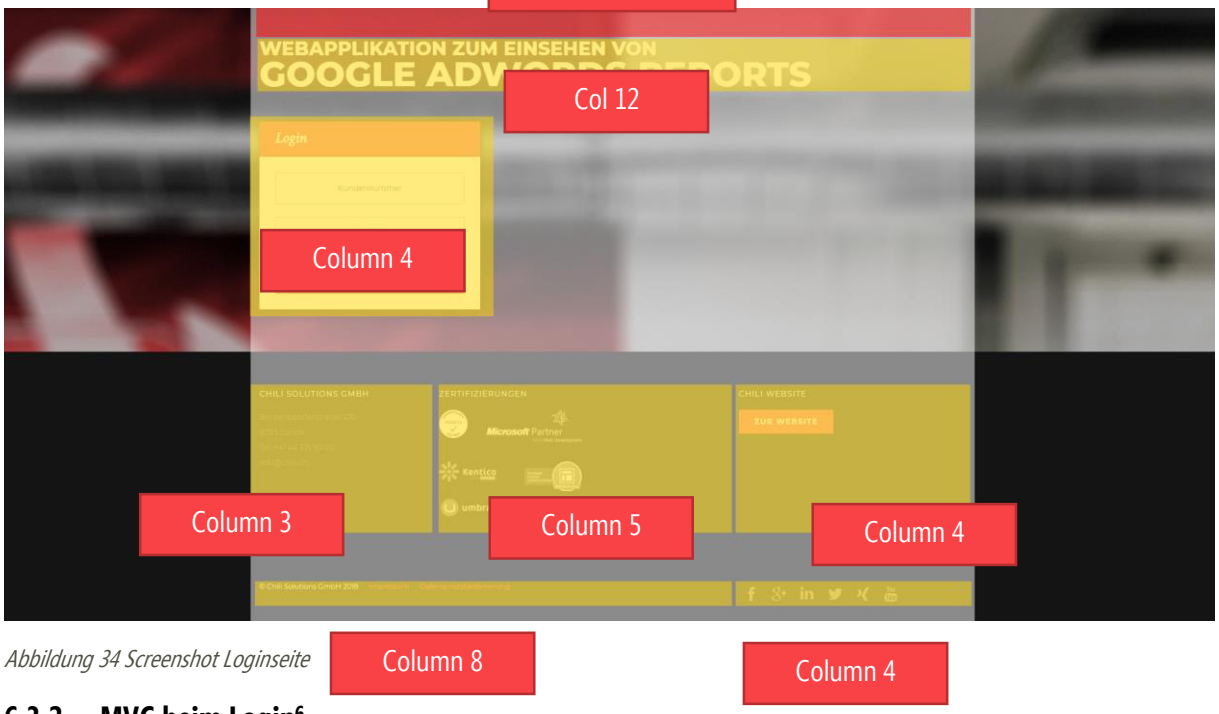


Abbildung 34 Screenshot Loginseite

6.3.2 MVC beim Login[6]

Da ich auf dem MVC-Konzept arbeite, werde ich natürlich auch beim Login die View, das Model und den Controller strikt trennen. Das sieht in meinem Fall etwa so aus:

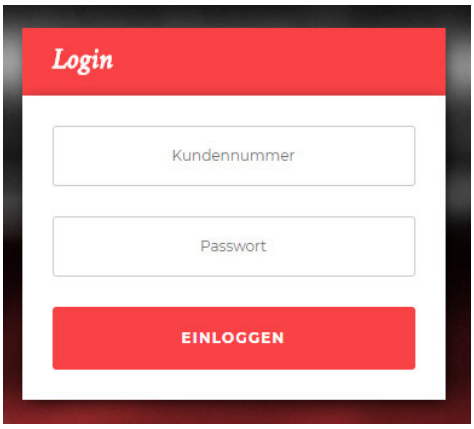
Model: TAB_Adresse.cs
Das Model TAB_Adresse ist ein Model das vom Entity Framework anhand der Tabelle in der Datenbank generiert wird. Es enthält die Kundennummer und das Passwort. Diese 2 Felder sind wichtig für die Überprüfung des Logins.

View: Login.cshtml
Die View enthält die HTML Struktur der Loginseite

Controller: LoginController.cs
Im Controller werden die Daten vom Model überprüft und dann Rückmeldungen gegeben.

[6] Quellcode von Model, View und Controller sind im Anhang zu finden

6.3.3 Das Loginformular



Das Loginformular besteht aus 2 Eingabefeldern und einem Button. Die 2 Eingabefelder sind Pflichtfelder, wenn diese nicht ausgefüllt sind, werden die Eingaben gar nicht erst geprüft, es wird direkt eine Fehlermeldung ausgegeben.

Abbildung 35 Screenshot Loginformular

6.3.4 Die Validierung der Felder

In der Entscheidungsphase habe ich mich bei der Validierung für die Methode der DataAnnotations entschieden.

Data Annotations

Annotations sind in der deutschen Sprachen Anmerkungen. Man kann zu allen Objekten in einem Model solche Anmerkungen schreiben die sich dann auf die Validierung auswirken.

Beispiel:

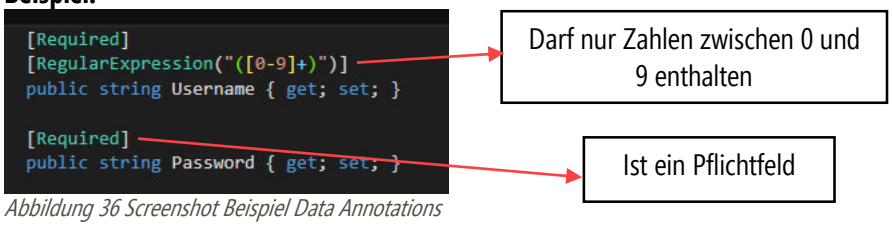


Abbildung 36 Screenshot Beispiel Data Annotations

Schwierigkeit

Ich benötige die DataAnnotations in einem Model (TAB_Adresse.cs). Dieses Model wird von der Datenbank generiert und jedes Mal wenn es eine Änderung am Model gibt (durch Anpassungen an der Datenbankstruktur), wird dieses Model neu generiert.

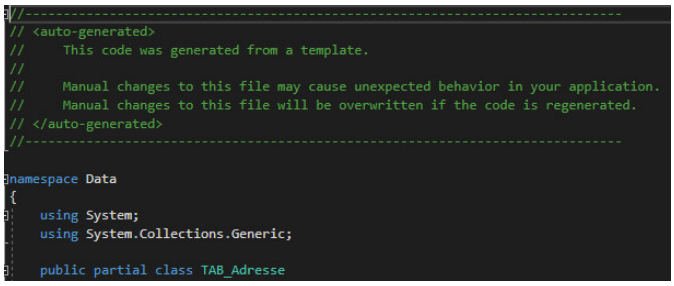


Abbildung 37 Screenshot automatisch generiertes Model

Wenn ich also die Data Annotations in das Model schreiben würde, besteht die Möglichkeit, dass diese später überschrieben werden und dann nicht mehr existieren. Dadurch wäre die Validierung nicht mehr gewährleistet.

Lösung[7]

Teilklassen (engl. PartialClasses) bieten die Möglichkeit Klassen in einem separaten File zu erweitern. Es wird also eine Metaklasse erstellt und dort werden die Data Annotations geschrieben. Diese Metaklasse wird dann über eine Teilklasse dem TAB_Adresse zugewiesen.

Umsetzung

Metaklasse erstellen: (MetaDaten.cs)

```
namespace Reporttool
{
    public class TAB_AdresseMeta
    {
        [Required]
        [Range(0, int.MaxValue)]
        public int NUM_Nummer { get; set; }

        [Required]
        public string TXT_Passwort { get; set; }
    }
}
```

Diese Klasse dient dazu, die Data Annotations zu definieren. Der Klassenname ist „Datenbankmodel" (in diesem Fall TAB_Adresse) und dann hinten Meta. Das muss nicht zwingend so gemacht werden, ist aber übersichtlich.

Abbildung 38 Screenshot Metadaten Klasse

Teilklasse erstellen: (Teilklassen.cs)

```
namespace Data
{
    [MetadataType(typeof(TAB_AdresseMeta))]
    public partial class TAB_Adresse
    {

    }
}
```

Diese Klasse verbindet die Metadaten (TAB_AdresseMeta) mit dem Datenbankmodel TAB_Adresse.

Abbildung 39 Screenshot Teilklasse

Schwierigkeit

Die Fehlermeldungen der Data Annotations sind standardmässig auf Englisch. Man kann aber über das Attribut „ErrorMessage" eine Fehlermeldung mittels eines Strings angeben. Eine schöne Lösung ist es aus meiner Sicht, wenn die Errormeldungen alle in einem separaten Resourcefile gespeichert sind und dann ausgelesen werden.

```
[Required(ErrorMessage = "Das Feld Passwort ist ein Pflichtfeld!")]
public string TXT_Passwort { get; set; }
```

Abbildung 40 Screenshot String als Errormeldung

In diesem Beispiel wird eine ErrorMessage als String definiert.

Wie das mit den Resourcefiles gehandhabt wird, ist in der „Phase 2 Resourcen" dokumentiert. Die Schwierigkeit besteht in der Phase Login darin, dass man als „ErrorMessage" nicht ein ResourceString angeben kann.

Lösung[8]

Der ErrorMessageResourceType muss umdefiniert werden, damit Resourcefiles benutzt werden könnnen.

```
[Required(ErrorMessageResourceType = typeof(Inhalte.Properties.Validierungsmeldungen),
    ErrorMessageResourceName = "passwortPflicht")]
public string TXT_Passwort { get; set; }
```

Abbildung 41 Screenshot Resourcestring als Errormeldung

Der Typ ist jetzt Inhalte.Properties.Validierungsmeldungen. Das Resourcefile „Validierungsmeldungen" enthält alle Meldungen der Validierung. In der Meldung *passwortPflicht* findet sich folgender Wert: **„Es wird ein Passwort benötigt"**

[7] Quelle Lösung https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/database-first-development/enhancing-data-validation
[8] Quelle Lösung https://stackoverflow.com/questions/2451150/dataannotations-and-resources-dont-play-nicely

6.3.5 Überprüfen der Benutzerdaten

Die Überprüfung der Benutzerdaten passiert mit einer Datenbankabfrage. Die Datenbankabfrage passiert aber nur dann wenn die Kundennummer wirklich eine Zahl ist und wenn das Passwort eingegeben wurde. Dann wird abgefragt ob es in der Datenbank(TAB_Adresse) einen Eintrag mit der eingegebenen Kundennummer und dem eingegebenen Passwort gibt. Dies geschieht im LoginController.cs (Siehe Anhang)

```
//Prüft ob es einen Kunden mit dieser Kundennummer und Passwort gibt
var myUser = dbContext.TAB_Adresse.FirstOrDefault(e =>
    e.NUM_Nummer == loginDaten.NUM_Nummer && e.TXT_Passwort == loginDaten.TXT_Passwort);
```

Abbildung 42 Screenshot Überprüfung Benutzerdaten

6.3.6 Session Handling

Session eröffnen

```
//Setzen der Sessionvariabeln
Session["KundenNr"] = myUser.NUM_Nummer.ToString();
Session.Timeout = 15;
```

Abbildung 43 Screenshot Session starten

Nachdem geprüft wurde ob der Kunde existiert, und der Kunde wirklich existiert, wird die Session eröffnet. Es wird eine Sessionvariable „KundenNr" gesetzt und ein Session.TimeOut von 15 Minuten.

Session beenden

```
//Session wird beendet
Session.Clear();
Session.Abandon();
Session.RemoveAll();
```

Abbildung 44 Screenshot Session beenden

Im LogoutController wird die Session beendet. Der LogoutController wird aufgerufen sobald der Kunde auf den Logout Button klickt.

Der Loginhelper

Der Loginhelper prüft ob eine Session aktiv ist.

```
/// <summary>
/// Prüft ob eine Session besteht
/// </summary>
/// <returns>Ob der User eingeloggt ist</returns>
public static bool IstEingeloggt()
{
    return (HttpContext.Current.Session["KundenNr"] != null);
}
```

Abbildung 45 Screenshot Loginhelper

6.3.7 Historisierung

Aufbereiten

```
//Aufbereitung der Daten für das History Logging
var historyDaten = new TAB_Adresse_History_Login
{
    DAT_DatumZeit = DateTime.Now,
    NUM_AdresseId = loginDaten.NUM_Nummer,
    TXT_IP = HttpContext.Request.ServerVariables["REMOTE_ADDR"],
    TXT_UserAgent = Request.UserAgent
};
```

Abbildung 46 Screenshot Historydaten aufbereiten

Für die Historisierung setze ich das aktuelle Datum, die aktuelle Kundennummer (die eingegeben wurde), die Servervariabel „REMOTE_ADDR" für die IP und den Request.UserAgent für den Useragent.
Schwierigkeit: IP ermitteln[9]

Schreiben

```
//History Text Login erfolgreich
historyDaten.TXT_Text = ResourceHelper.GetValidierungsmeldung("loginErfolgreich");
//History in DB schreiben
HistoryHelper.Schreiben(historyDaten);
```

Abbildung 47 Screenshot Historydaten setzen

Diese Daten werden danach mit der jeweiligen Fehlermeldung oder mit dem Vermerk „Login erfolgreich" in die Datenbank geschrieben (Tabelle TAB_Adresse_History_Login)

Der HistoryHelper

Der HistoryHelper hängt die Historydaten an die Historytabelle an.

```
/// <summary>
/// Hängt Daten an die History Tabelle an
/// </summary>
/// <param name="history">Historyobjekt befüllt mit Daten für die Historisierung</param>
public static void Schreiben(TAB_Adresse_History_Login history)
{
    using (var dbContext = new marketingEntities())
    {

        dbContext.TAB_Adresse_History_Login.Add(history);
        dbContext.SaveChanges();
    }
}
```

Abbildung 48 Screenshot Historydaten schreiben

[9] Quelle Lösung: https://stackoverflow.com/questions/1907195/how-to-get-ip-address

6.4 Phase 2 Resourcenverwaltung

Für eine saubere Projektstruktur und eine einfache Verwaltung der Applikation, sollen alle Resourcen wie Texte, Farben im Diagramm oder die Validierungsmeldungen beim Login in Resourcefiles ausgelagert werden.

6.4.1 Resourcefile

Ein Resourcefile ist ein XML-basiertes File, welches Daten enthält. Es gibt immer einen Tag „**Data**" mit einem Namen und darin ist dann die „**Value**" gespeichert.

```
<data name="falschesPasswort" xml:space="preserve">
    <value>Passwort falsch</value>
</data>
<data name="kundennummerPflicht" xml:space="preserve">
    <value>Es wird eine Kundennummer benötigt</value>
</data>
```

Abbildung 49 Screenshot Struktur Resourcefile

Codestruktur eines ResourceFiles

Die Ansicht links, so sieht die XML Strutkur eines Resourcefiles aus

Wenn man das Resourcefile in Visual Studio bearbeitet erhält man aber eine schönere Ansicht:

	Name	Value	Comment
▶	falschesPasswort	Passwort falsch	
	kundennummerPf	Es wird eine Kundennummer benötigt	
	loginErfolgreich	Login erfolgreich	
	nummerFehler	Die Kundenummer darf nur aus Zahlen	
	passwortPflicht	Es wird ein Passwort benötigt	
*			

Abbildung 50 Screenshot Struktur Resourcefile in Visual Studio

6.4.2 Vorteil

Somit sind die Validierungsmeldungen, Farben und Texte einfach anzupassen und zentral verwaltet. Sie können durch die Geschäftsleitung jederzeit mit einem XML-Viewer bearbeitet und angepasst werden, wenn dies gewünscht ist.

Falls das Tool mal mehrsprachig werden würde, könnte man einfach das Ressourcefile kopieren und übersetzen.

6.4.3 Struktur im Projekt

Für die Inhalte werde ich ein neues Projekt innerhalb der Projektmappe erstellen. Das mache ich um eine „**Circular dependency**"[10] zu verhindern. Das passiert dann wenn 2 Projekte sich gegenseitig referenzieren. Hätte ich zum Beispiel die Resourcefiles in der ClassLibrary erfasst und möchte dann vom Model im Projekt „Data" darauf zugreifen, dann geht das nicht, weil die ClassLibrary hat das Projekt „Data" für den Datenbankzugriff referenziert

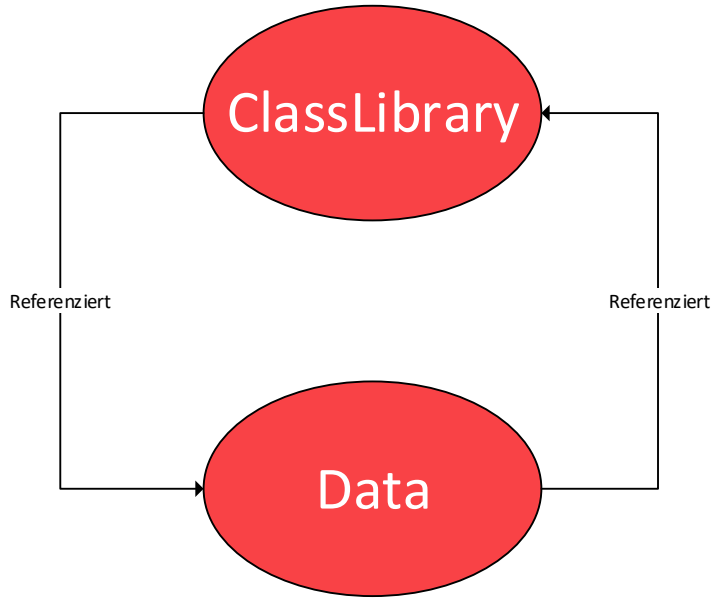
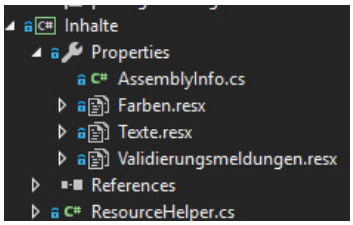


Abbildung 51 Darstellung Circular dependency

[10] Quelle Circular dependency: https://en.wikipedia.org/wiki/Circular_dependency

Die Resourcefiles



In diesem Projekt werden die Ressourcen in 3 einzelne Resourcefiles unterteilt. Das sind zum einem die Farben, die Texte und die Validierungsmeldungen.

Abbildung 52 Screenshot Resourcefiles im Projekt Inhalte

Der ResourceHelper

Damit ich von überall her Zugriff auf diese Daten habe, habe ich den ResourceHelper erstellt. Dieser enthält 3 Methoden die ich von überall aufrufen kann.

```
public class ResourceHelper
{
    /// <summary>
    /// Sucht im Resourcefile "Texte" nach Daten
    /// </summary>
    /// <param name="key">Schlüsselwort nach dem gesucht werden soll</param>
    /// <returns>Den Inhalt dieses Schlüsselworts</returns>
    public static string GetText(string key)
    {
        return Texte.ResourceManager.GetString(key);
    }
```

Abbildung 53 Screenshot Resourcehelper

Zugriff bei den Validierungsmeldungen

Bei den Validierungsmeldungen geschieht der Zugriff direkt ohne den ResourceHelper. Weil man als ErrorMessage keinen Methodenaufruf schreiben kann, wird direkt das Property ausgelesen. Siehe Kapitel 6.3.4 unter Lösung 2.

Reflexion Ressourcenverwaltung

Obwohl die Umsetzung mit den Ressourcen einiges an Zeit benötigt hat, bin ich doch froh es so realisiert zu haben. So kann ich bei allen anderen Umsetzungsschritten Zeit sparen, weil ich schnell und einfach auf die ResourceFiles verweisen kann. Es ist eine saubere Lösung.

6.5 Phase 3 Datenbanksicht

Damit ich die Daten möglichst simpel aufrufen kann, habe ich mich entschieden im SQL Server Management Studio eine View (auf Deutsch Sicht) zu erstellen. In dieser Datenbanksicht sind alle wichtigen Tabellen die ich brauche verbunden und geben mir die Werte zurück, die ich benötige.

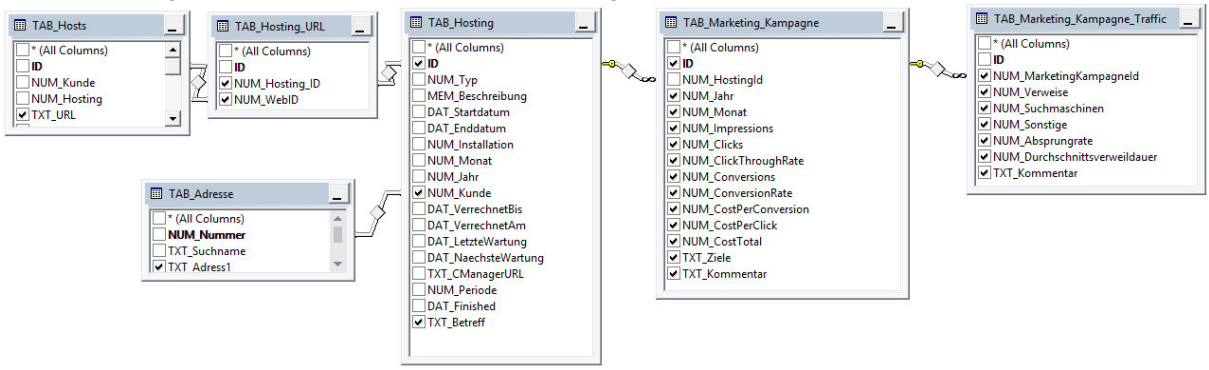


Abbildung 54 Screenshot Datenbanksicht in SQL Management Studio

6.5.1 Verbindungen der Tabellen:

Eine kurze Erläuterung wie die Tabellen miteinander verbunden sind. Ergänzung zur Datenbankanalyse.

TAB_Marketing_Kampagne_Traffic

Hat einen Fremdschlüssel auf **TAB_Marketing_Kamapgne** (NUM_MarketingKampagneId)

TAB_Marketing_Kampagne

Hat einen Fremdschlüssel auf **TAB_Hosting** (NUM_HostingId)

TAB_Hosting

Ist durch den Primärschlüssel ID mit dem Fremdschlüssel von **TAB_Hosting_URL** (NUM_Hosting_ID) verbunden.
Hat einen Fremdschlüssel NUM_Kunde auf **TAB_Adresse** (NUM_Nummer)

TAB_Hosting_URL

Hat eine Fremdschlüssel NUM_WebID auf **TAB_Hosts** (ID)

```
public partial class vw_Marketing
{
    public int Hosting_ID { get; set; }
    public Nullable<int> NUM_Kunde { get; set; }
    public int NUM_MarketingKampagneId { get; set; }
    public int NUM_Verweise { get; set; }
    public int NUM_Sonstige { get; set; }
    public double NUM_Absprungrate { get; set; }
    public int NUM_Suchmaschinen { get; set; }
    public double NUM_Durchschnittsverweildauer { get; set; }
    public string TXT_Kommentar_Traffic { get; set; }
    public int NUM_Jahr { get; set; }
    public int NUM_Monat { get; set; }
    public int NUM_Impressions { get; set; }
    public int NUM_Clicks { get; set; }
    public double NUM_ClickThroughRate { get; set; }
    public int NUM_Conversions { get; set; }
    public double NUM_CostPerConversion { get; set; }
    public double NUM_ConversionRate { get; set; }
    public double NUM_CostPerClick { get; set; }
    public string TXT_Ziele { get; set; }
    public double NUM_CostTotal { get; set; }
    public string TXT_Kommentar { get; set; }
    public string TXT_Passwort { get; set; }
    public int ID { get; set; }
    public string TXT_Betreff { get; set; }
    public string TXT_Adress1 { get; set; }
    public string TXT_URL { get; set; }
    public Nullable<int> NUM_WebID { get; set; }
    public Nullable<int> NUM_Hosting_ID { get; set; }
}
```

Diese Datenbanksicht konnte ich dann anschliessend mit Entity Framework in das Projekt einbinden, so dass ich ein Model bekam, welches diese Datenstruktur beinhaltet. Das ist das Model **vw_Marketing**. „Vw" steht in diesem Fall für View und wird so bei allen Views in der Chili Datenbank gehandhabt.

Abbildung 55 Datenbank generiertes Model vw_Marketing

6.5.2 Erweiterungen für das Model vw_Marketing

Da ich das Model Vw_Marketing gerne in meinem 2 Views Uebersicht.cshtml und ReportDetail.cshtml verwenden will, benötige ich noch ein paar weitere Daten.

Schwierigkeit

Ich kann die Daten mit ihren Datentypen nicht einfach in das Datenbankmodel hineinschreiben, da dieses bei Änderungen in der Datenbank überschrieben werden würde.

Lösung

Ich arbeite wie schon bei der Validierung mit Teilklassen und ergänze die fehlenden Daten einfach in der Teilklasse.

```
public partial class vw_Marketing
{
    public List<vw_Marketing> VorherigeReports { get; set; }
    public vw_Marketing ReportVormonat { get; set; }
    public vw_Marketing ReportFolgemonat { get; set; }
    public vw_Marketing AenderungenVormonat { get; set; }
    public int Besuche { get; set; }
    public int BesucheVorjahr { get; set; }
    public double BesucheProzent { get; set; }
    public double ImpressionenProzent { get; set; }
    public double KlicksProzent { get; set; }
    public List<TAB_Marketing_Kampagne_Massnahme> MassnahmenVormonatListe { get; set; }
    public List<TAB_Marketing_Kampagne_Massnahme> MassnahmenListe { get; set; }
    public bool IstAktuellerReport { get; set; }
}
```

Abbildung 56 Screenshot der Teilklasse vom Datenbankmodell vw_Marketing

6.5.3 Erklärung der einzelnen erweiterten Werte und wie diese befüllt werden

VorherigeReports: Eine Liste die mit mehreren Reports gefüllt werden kann. Wird benötigt, wenn die vorherigen Reports dargestellt werden müssen (beim Diagramm und der Tabelle auf der Reportdetailseite)

Kann mit der Methode GetVorherigeReports im ReportsHelper.cs befüllt werden.
Dieser Methode muss der aktuelle Report und die Anzahl der Monate die man zurückgehen will übergeben werden.

ReportVormonat: Enthält den Report des vorherigen Monats im Bezug auf den aktuellen Monat. Wird benötigt wenn die Daten des Vormonats mit den aktuellen Daten verglichen werden sollen.

Kann mit der Methode GetReportVormonatDurchReportId im ReportsHelper.cs befüllt werden.
Dieser Methode muss die ReportId des aktuellen Reports übergeben werden

ReportFolgemonat: Enthält den Report des Folgemonats im Bezug auf den aktuellen Monat. Wird benötigt für den Button „zum nächsten Monat".

Kann mit der Methode GetReportFolgeMonat im ReportsHelper.cs befüllt werden.
Dieser Methode muss die ReportId des aktuellen Reports übergeben werden

AenderungVormonat: Enthält die Werte, die beim Vergleich von 2 Reports berechnet werden. Wird auf der Reportdetailseite dargestellt (Änderungen zum Vormonat)

Kann mit der Methode VergleicheVormonatMitReport im ReportsHelper.cs befüllt werden.
Dieser Methode muss der aktuelle Report übergeben werden.

Besuche: Enthält den berechneten Wert, wenn man die Suchmaschinenzugriffe, Verweiszugriffe und die sonstigen Zugriffe addiert. Wird auf der Reportdetailseite benutzt.

BesucheVorjahr: Enthält die Anzahl der Besuche des Reports im vorherigen Jahr. Wird im Diagramm benutzt um den Wert (Traffic Vorjahr) darzustellen.

Kann mit der Methode GetBesucheVorjahr im ReportsHelper.cs befüllt werden.
Dieser Methode muss der aktuelle Report übergeben werden.

BesuchеProzent: Enthält den prozentualen Wert, wenn man die Besuche von zwei Reports vergleicht.
ImpressionenProzent: Enthält den prozentualen Wert, wenn man die Impressionen von zwei Reports vergleicht.
KlicksProzent: Enthält den prozentualen Wert, wenn man die Impressionen von zwei Reports vergleicht.

Werden in der Methode VergleicheVormonatMitReport im ReportsHelper.cs befüllt.

MassnahmenVormonatListe: Enthält eine Liste mit allen Massnahmen des Vormonats eines Reports.
MassnahmenListe: Enthält eine Liste mit allen Massnahmen eines Reports.

Können mit der Methode GetMassnahmenDurchReportId im ReportsHelper.cs befüllt werden
Dieser Methode muss die ReportId des Reports übergeben werden, von dem man die Massnahmen als Liste haben möchte.

IstAktuellerReport: Definiert ob der Report der aktuelle Report ist (der Report des aktuellen Monats)

Wird in der Methode GetReportsDurchKundenNummer im ReportsHelper.cs gesetzt.

6.5.4 SQL Query der Datenbanksicht[11]

```
SELECT      dbo.TAB_Hosting.ID AS Hosting_ID, dbo.TAB_Hosts.TXT_URL, dbo.TAB_Hosting.NUM_Kunde,
dbo.TAB_Marketing_Kampagne_Traffic.NUM_MarketingKampagneId, dbo.TAB_Marketing_Kampagne_Traffic.NUM_Verweise,
                  dbo.TAB_Marketing_Kampagne_Traffic.NUM_Sonstige,
dbo.TAB_Marketing_Kampagne_Traffic.NUM_Absprungrate, dbo.TAB_Marketing_Kampagne_Traffic.NUM_Suchmaschinen,
                  dbo.TAB_Marketing_Kampagne_Traffic.NUM_Durchschnittsverweildauer,
dbo.TAB_Marketing_Kampagne_Traffic.TXT_Kommentar AS TXT_Kommentar_Traffic,
dbo.TAB_Marketing_Kampagne.NUM_Jahr,
                  dbo.TAB_Marketing_Kampagne.NUM_Monat, dbo.TAB_Marketing_Kampagne.NUM_Impressions,
dbo.TAB_Marketing_Kampagne.NUM_Clicks, dbo.TAB_Marketing_Kampagne.NUM_ClickThroughRate,
                  dbo.TAB_Marketing_Kampagne.NUM_Conversions, dbo.TAB_Marketing_Kampagne.NUM_CostPerConversion,
dbo.TAB_Marketing_Kampagne.NUM_ConversionRate, dbo.TAB_Marketing_Kampagne.NUM_CostPerClick,
                  dbo.TAB_Marketing_Kampagne.TXT_Ziele, dbo.TAB_Marketing_Kampagne.NUM_CostTotal,
dbo.TAB_Marketing_Kampagne.TXT_Kommentar, dbo.TAB_Adresse.TXT_Passwort, dbo.TAB_Marketing_Kampagne.ID,
                  dbo.TAB_Hosting.TXT_Betreff, dbo.TAB_Adresse.TXT_Adress1, dbo.TAB_Hosting_URL.NUM_WebID,
dbo.TAB_Hosting_URL.NUM_Hosting_ID
FROM        dbo.TAB_Hosts INNER JOIN
                  dbo.TAB_Hosting_URL ON dbo.TAB_Hosts.ID = dbo.TAB_Hosting_URL.NUM_WebID INNER JOIN
                  dbo.TAB_Adresse INNER JOIN
                  dbo.TAB_Hosting INNER JOIN
                  dbo.TAB_Marketing_Kampagne_Traffic INNER JOIN
                  dbo.TAB_Marketing_Kampagne ON dbo.TAB_Marketing_Kampagne_Traffic.NUM_MarketingKampagneId =
dbo.TAB_Marketing_Kampagne.ID ON dbo.TAB_Hosting.ID = dbo.TAB_Marketing_Kampagne.NUM_HostingId ON
                  dbo.TAB_Adresse.NUM_Nummer = dbo.TAB_Hosting.NUM_Kunde ON
dbo.TAB_Hosting_URL.NUM_Hosting_ID = dbo.TAB_Hosting.ID
```

[11] Diese SQL Query wurde vom SQL Server Management Studio beim Erstellen der Datenbanksicht generiert.

6.6 Phase 4 Reportübersichtsseite

6.6.1 Die HTML Seite

Wie schon bei der Loginseite baue ich zuerst die HTML Seite auf und fülle diese erst danach mit richtigen Daten aus der Datenbank.

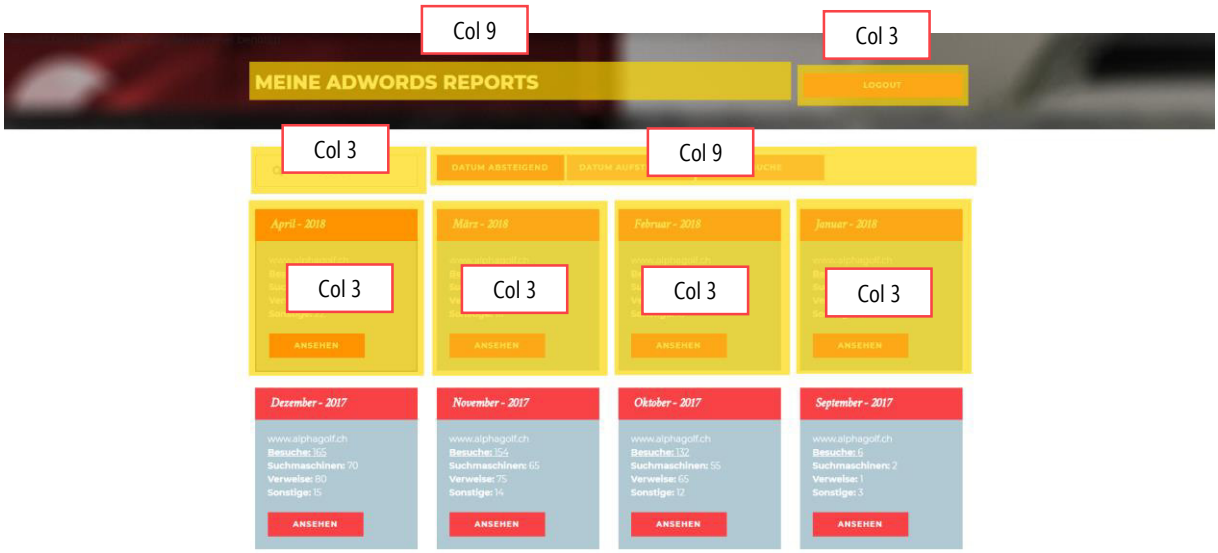


Abbildung 57 Screenshot HTML Reportübersichtsseite

6.6.2 MVC bei der Reportsübersichtsseite

Auch bei der Reportsübersichtsseite werden wie beim Login das Model, die View und der Controller strikt getrennt. Die Definition der 3 Elemente sieht hier wie folgt aus:

Model: vw_Marketing.cs[12]
Das Model vom Typ vw_Marketing. Da auf dieser Seite aber mehrere Reports (vw_Marketing) dargestellt werden müssen ist das Model eine Liste vom Typ <vw_Marketing>. Das wird in der View so definiert:

```
@model List<Data.vw_Marketing>
```

Abbildung 58 Screenshot Modeldefiniton in der Reportsübersichts-View

View: Uebersicht.cshtml
Die View enthält die HTML Struktur der Seite und stellt die Daten aus dem Model mit Razor dar.

Controller: ReportUebersichtController.cs
Der Controller prüft zuerst **ob der Kunde eingeloggt ist**[13], ist dies nicht der Fall wird er zum Login weitergeleitet.

Ist der Kunde eingeloggt, holt der Controller holt Daten (in diesem Fall ist es eine Liste mit Reports) mit Hilfe der Helperklassen aus der Datenbank und übergibt sie an die View.
Ergänzung: Später wurde noch geprüft ob es nur einen Report gibt, dann wird man direkt zur Detailseite geleitet.

```
if (LoginHelper.IstEingeloggt())
{
    var reportsList = ReportsHelper.GetReportsDurchKundenNummer(Convert.ToInt32(Session["KundenNr"]));

    return View("Uebersicht", reportsList);
}
return RedirectToAction("Index", "Login");
```

Abbildung 59 Screenshot ReportUebersichtsController Logik

[12] Dies ist das wichtigste Model im Projekt und wird im Kapitel 6.5 behandelt
[13] Auf der Abbildung 45 im Kapitel 6.3.6 Session Handling ist der Loginhelper noch im Detail zu sehen.

6.6.3 Auflistung der Reports

Die Auflistung der Reports geschieht in der View. Da das Model eine Liste ist mit mehreren Elementen, kann man in der View eine Schleife erstellen, die durch alle Elemente in der Liste hindurchgeht. Das wird mit einer foreach Schleife gemacht.

```
@foreach (var report in Model)
{
    // Hier kommt der Teil der pro Report wiederholt werden soll z.B
    <div>
        <h2>@report.NUM_Monat @report.NUM_Jahr</h2>
        <b>Die wichtigsten Kennzahlen</b>
        <ul>
            <li>Besuche: @report.Besuche</li>
            <li>Klicks: @report.NUM_Clicks</li>
        </ul>
    </div>
}
```

Abbildung 60 Screenshot Beispiel foreach Schleife

Innerhalb der Schleife kann man dann auf alle Werte die im Model gespeichert sind zugreifen.

6.6.4 Initialisierung Isotope

```
//initialisiert das isotope-Grid und definiert die Sortierdaten
$(function () {
    var $grid = $(".grid").isotope({
        itemSelector: '.grid-item',
        percentPosition: true,
        getSortData: {
            datedesc: '.datedesc',
            besuche: '.besuche',
            dateasc: '.dateasc',
        },
        sortBy: 'datedesc',
        sortAscending: {
            datedesc: false,
            dateasc: true,
            besuche: false
        }
    });
```

Klasse des Parents der zu sortierenden/filtrierenden Elemente

Klasse der zu sortierenden/filtrierenden Elemente

Klassen der Daten nach denen sortiert werden soll

Nach diesem Element wird beim Seitenaufruf sortiert

Definiton ob Element aufsteigend oder absteigend sortiert werden soll

Abbildung 61 Screenshot Initialisierung Isotope

6.6.5 Textfiltrierung

In der Entscheidungsphase habe ich mich entschieden, die Filtrierung mit isotope.js[14] vorzunehmen. In diesem Schritt werde ich die Filtrierung mit isotope erklären. Das Ganze passiert Clientseitig mit jQuery.

Die Texteingabe in die Textbox abfangen

Sobald sich der Text im Suchfeld ändert, soll eine Funktion aufgerufen werden, die alle Elemente nach dem Text im Suchfeld filtert.

```
//Beim Ändern des Suchfeldes wird meine Filterfunkiton aufgerufen
$("input.suchen").on('input propertychange paste', function () {
    $grid.isotope({
        filter: function () {
            return textBoxFilter(this);
        }
    });
});
```

Abbildung 62 Screenshot Textboxeingabe Änderung

[14] Website isotope.js: https://isotope.metafizzy.co/

Die Filterfunktion

```
function textBoxFilter(report) {
    var eingabeText = $("input.suchen").val().toLowerCase();

    // Text der mit der Eingabe verglichen werden soll wird ausgelesen
    var kennzahlen = $(report).find('.filterValue').text().toLowerCase();
    var titel = $(report).find('.filterTitle').text().toLowerCase();

    //true zurückgeben und Element anzuzeigen, false zum verstecken
    var elementAnzeigen = (beinhaltetWort(kennzahlen, eingabeText) || beinhaltetWort(titel, eingabeText));
    return elementAnzeigen;
}

//testet ob das Wort innerhalb eines anderen Strings beinhaltet ist
function beinhaltetWort(myString, myWord) {
    return myString.indexOf(myWord) > -1;
}
```

Abbildung 64 Screenshot FilterFunktion mit beinhaltetWort Funktion

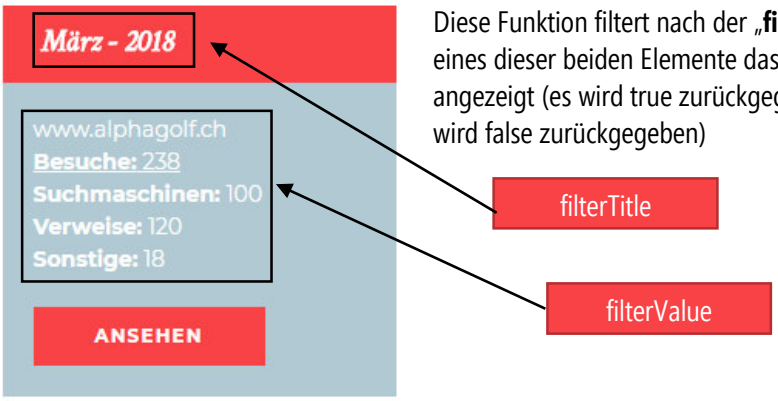


Abbildung 63 Screenshot Ansicht eines Reports

Diese Funktion filtert nach der „**filterValue**“ und nach „**filterTitle**“. Wenn eines dieser beiden Elemente das Wort im Eingabefeld beinhaltet, wird es angezeigt (es wird true zurückgegeben) sonst wird es ausgeblendet (es wird false zurückgegeben)

Die zweite Funktion „beinhaltetWort“ prüft von 2 Strings ob der eine den anderen beinhaltet und so wird festgestellt, ob es angezeigt wird oder nicht.

6.6.6 Sortierung der Reports

Für das Sortieren gibt es von Isotope eine vorgefertigte Funktion. Man muss nur beim Button, welcher die Reports sortieren soll ein Data-Attribut setzen. Wenn die Sortierdaten beim Initialisieren korrekt gesetzt wurden (Siehe Abbildung 61) dann ist man mit der Sortierung schon fertig.

```
<div class="sort-by-button-group">
    <div class="customButton sortBtn aktiv" data-sort-value="datedesc">Datum absteigend</div>
    <div class="customButton sortBtn" data-sort-value="dateasc">Datum aufsteigend</div>
    <div class="customButton sortBtn" data-sort-value="besuche">Besuche</div>
</div>
```

6.6.7 Problematik Monat als Zahl

An einigen Stellen im Projekt, benötige ich den Monat als Text. 1 soll zu Januar werden, 2 soll zu Februar werden. Dies ist zum Beispiel auf der Abbildung 63 im Titel erforderlich. Das ist der perfekte Anwendungsfall für ein **Enum**, Da die Monate immer gleich bleiben. Zum Beispiel 5 wird immer Mai bleiben.

Ein Enum ist eine Aufzählung von Werten die von überall her angesprochen werden können und eindeutig sind. Das Enum werde ich im Teilprojekt „Inhalte“ erstellen und kann so von überall darauf zugreifen.

Enum Monate Deklarierung

```
namespace Inhalte
{
    public enum Monate
    {
        Januar = 1,
        Februar = 2,
        März = 3,
        April = 4,
        Mai = 5,
        Juni = 6,
        Juli = 7,
        August = 8,
        September = 9,
        Oktober = 10,
        November = 11,
        Dezember = 12
    }
}
```

Abbildung 65 Screenshot enum Monate

Enum Monate Aufruf

Im ReportsHelper.cs wurde eine Methode geschrieben die den Monat als Ganzzahl entgegennimmt und dann den Monat als String zurückggibt.

```
/// <summary>
/// Konvertiert den Monat als Int zu einem String
/// 1 zu Januar
/// </summary>
/// <param name="monat">Monat als Int (1-12)</param>
/// <returns>Monat als String</returns>
public static string GetMonatDurchZahl(int monat)
{
    return Enum.GetName(typeof(Inhalte.Monate), monat);
}
```

Abbildung 66 Screenshot Methode Monat zu String

Alternative Möglichkeiten

Im Verlauf der IPA sind mir noch weitere Möglichkeiten, wie ich diese Problemstellung hätte lösen können, aufgefallen. Ich liste diese hier nun kurz auf:

1) Ich hätte ein Resourcefile erstellen können mit den 12 Monaten, wie ich es mit den anderen Resourcefiles auch gemacht habe.

2) Es gibt von .NET eine Möglichkeit ein DateTime.Monat zu einem String zu konvertieren, wenn man dazu noch die aktuelle Sprache mitgibt.

```
public static string GetMonatDurchZahl(int monat)
{
    return new DateTime(2015, monat, 1).ToString("MMMM", CultureInfo.CreateSpecificCulture("de"));
}
```

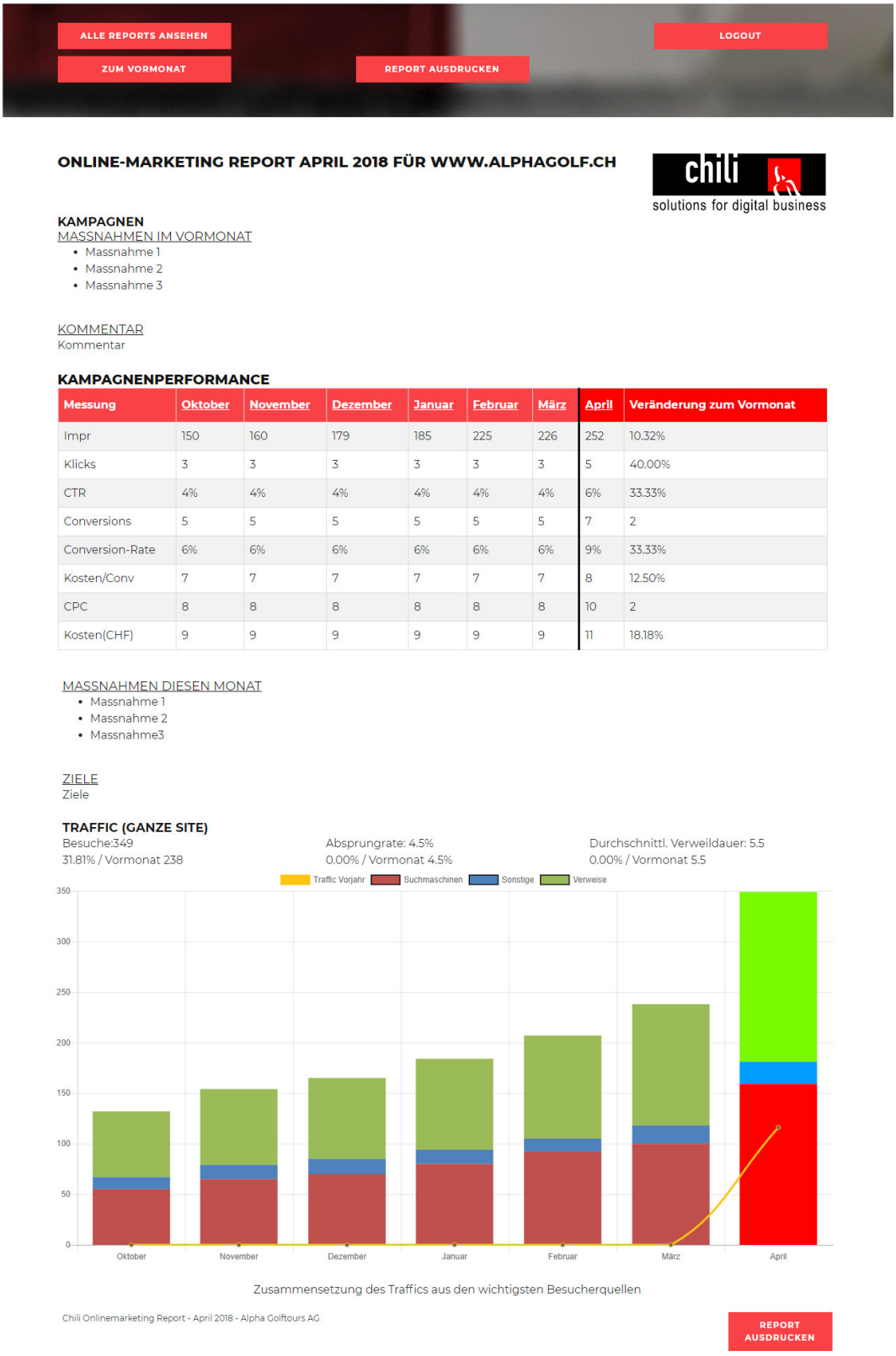
Abbildung 67 Screenshot alternative Methode zum Enum

Die Funktion hätte so ausgesehen. Ich habe mich dann aber gegen diese Methode und für das Enum entschieden, da hier jedesmal ein neues DateTime Object erstellt werden muss und dann von diesem der Monat konvertiert wird. Das ist ein unnötiger Umweg, der zwar funktioniert, die Methode mit dem Enum macht aber aus meiner Sicht mehr Sinn.

6.7 Phase 5 Reportdetailseite

6.7.1 Die HTML Seite

Wie schon bei den 2 anderen Seiten die erstellt wurden beginne ich auch bei der Reportdetailseite mit der HTML Seite. Dabei werden von Hand geschriebene Testdaten verwendet, die noch nicht aus der Datenbank ausgelesen werden.



Messung	Oktober	November	Dezember	Januar	Februar	März	April	Veränderung zum Vormonat
Impr	150	160	179	185	225	226	252	10.32%
Klicks	3	3	3	3	3	3	5	40.00%
CTR	4%	4%	4%	4%	4%	4%	6%	33.33%
Conversions	5	5	5	5	5	5	7	2
Conversion-Rate	6%	6%	6%	6%	6%	6%	9%	33.33%
Kosten/Conv	7	7	7	7	7	7	8	12.50%
CPC	8	8	8	8	8	8	10	2
Kosten(CHF)	9	9	9	9	9	9	11	18.18%

Abbildung 68 Screenshot HTML Reportdetailseite

6.7.2 MVC in der Reportdetailseite

Auch bei der letzten Seite werde ich mich wieder strikt an das Model, View und Controller Schema halten.

Model: vw_Marketing.cs
Das Model ist vom Typ vw_Marketing, also die Datenbanksicht die alle Daten enthält. Diesmal ist es aber keine Liste dieses Typs, wie bei der Reportübersichtsseite, sondern wirklich nur der Typ vw_Marketing. Das ist so, weil nur ein Report dargestellt wird und nicht mehrere.

Das Model enthält zwar noch eine Liste vom Typ vw_Marketing und zwar sind dass die vorherigen Reports die zum einem in der Tabelle zum Zug kommen und im Diagramm. Mehr dazu ist untenstehend bei der Datenverteilungsübersicht zu finden.

View: Reportdetail.cshtml
Hier werden alle benötigten Daten dargestellt. Wie diese genau dargestellt werden, dazu komme ich weiter unten. Auf dieser Seite ist das ein wenig komplexer als bisher, da noch einiges berechnet werden muss und es ein paar spannende Schleifen gibt.

Controller: ReportDetailController.cs
Der Controller prüft zuerst wieder ob der Kunde eingeloggt ist.

Falls der Kunde eingeloggt wird werden 7 Methoden aufgerufen. Diese 7 Methoden dienen dazu das Model so vorzubereiten, dass in der View nur noch Daten ausgegeben werden und keine mehr mutiert. Was diese 7 Methoden im Detail machen, wird später erläutert.

```
public ActionResult Report(int id)
{
    if (LoginHelper.IstEingeloggt())
    {
        int reportId = id;

        var model = ReportsHelper.GetReportDurchReportId(reportId);
        model.VorherigeReports = ReportsHelper.GetVorherigeReports(model, 6);
        model.ReportVormonat = ReportsHelper.GetReportVormonatDurchReportId(reportId);
        model.ReportFolgemonat = ReportsHelper.GetReportFolgeMonatDurchReportId(reportId);
        model.AenderungenVormonat = ReportsHelper.VergleicheVormonatMitReport(model);
        model.MassnahmenListe = ReportsHelper.GetMassnahmenDurchReportId(model.ID);
        model.MassnahmenVormonatListe = ReportsHelper.GetMassnahmenDurchReportId(model.ReportVormonat.ID);

        return View("ReportDetail", model);
    }
    return RedirectToAction("Index", "Login");
}
```

Abbildung 69 Screenshot Reportdetailcontroller

6.7.3 Datenverteilungsübersicht

Das Model das an die View übergeben wird, ist vom Typ vw_Marketing.[15] Diese Darstellung zeigt auf, wo welche Werte zum Einsatz kommen. Das vw_Marketing Model wird in der Darstellung mit „Model." angesprochen.

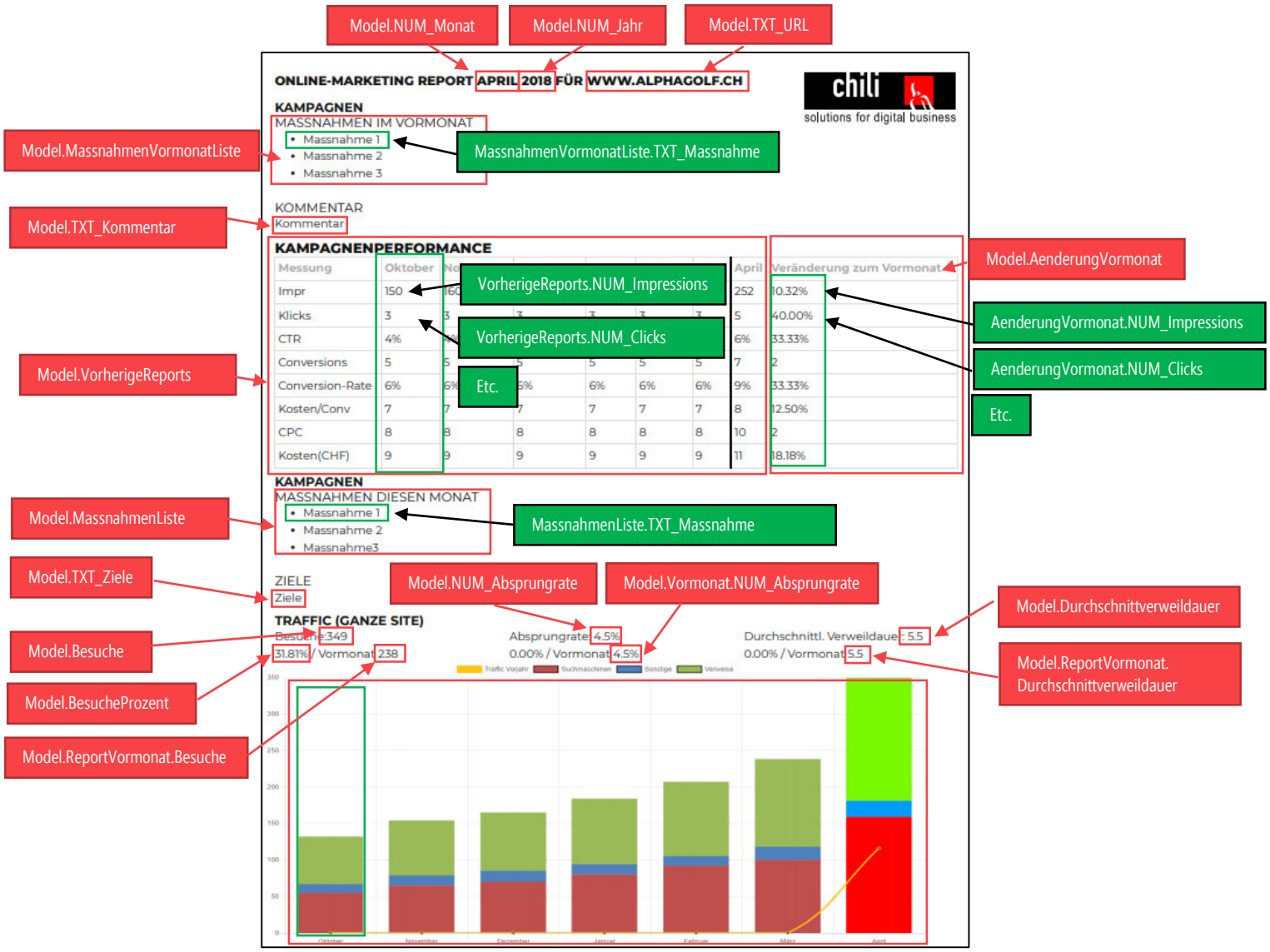


Abbildung 70 Datenverteilungsansicht

Diagramm wird über einen API-Call gemacht. Die Modellstruktur wird eine eigene sein, die an Chart.js angepasst ist. **Modelname: DiagrammDaten.cs**

mehr dazu unter Kapitel 6.7.5

[15] Das Model vw_Marketing.cs und die Teilklassen.cs sind im Anhang ersichtlich. Dort sind alle Werte aufgelistet

6.7.4 Daten beschaffen[16]

Die Daten werden über den ReportsHelper beschafft. Der Aufruf auf den Reporthelper geschieht im ReportDetailController.

```
1  var model = ReportsHelper.GetReportDurchReportId(reportId);
2  model.VorherigeReports = ReportsHelper.GetVorherigeReports(model, 6);
3  model.ReportVormonat = ReportsHelper.GetReportVormonatDurchReportId(reportId);
4  model.ReportFolgemonat = ReportsHelper.GetReportFolgeMonatDurchReportId(reportId);
5  model.AenderungenVormonat = ReportsHelper.VergleicheVormonatMitReport(model);
6  model.MassnahmenListe = ReportsHelper.GetMassnahmenDurchReportId(model.ID);
-- model.MassnahmenVormonatListe = ReportsHelper.GetMassnahmenDurchReportId(model.ReportVormonat.ID);
```

Abbildung 71 Methodenaufrufe im ReportDetailController

Das sind 7 Aufrufe auf 6 verschiedene Methoden. Die Methoden werden folgend aufgelistet mit den wichtigsten Eigenschaften. Wenn bei einer Methode etwas speziell war, oder ich etwas Neues gelernt habe, wird dies ersichtlich sein.

1: GetReportDurchReportId

```
/// <summary>
/// Datenbankabfrage eines Reports durch die ReportId
/// </summary>
/// <param name="reportId">Identifizierende ID eines Reports</param>
/// <returns>ein einziger Report</returns>
```

Abbildung 72 Screenshot des Kommentars GetReportDurchReportId

2: GetVorherigeReports

```
/// <summary>
/// Sucht nach vorhergehenden Reports eines übergebenen Reports
/// </summary>
/// <param name="aktuellerReport">Der Report, von dem aus vorhergehende Reports gesucht werden sollen</param>
/// <param name="anzahlMonateZurueck">Anzahl vorhergehende Reports die zurückgegeben werden sollen</param>
/// <returns>Eine Liste mit vorhergehenden Reports</returns>
```

Abbildung 73 Screenshot des Kommentars GetVorherigeReports

Schwierigkeit:

Um den vorherigen Report zu finden, muss ich den Monat um 1 verringern. Ist der aktuelle Monat im August wird die Zahl 8 um 7 verringert und der Report im Juli wird abgefragt.

Ist der aktuelle Report aber im Monat Januar würde ich mit dieser Methode den Monat 0 abfragen, den es nicht gibt. Also wird geprüft ob der Monat = 1 ist und dann wird der Monat 12 also Dezember gesetzt.

3: GetReportVormonatDurchReportID

```
/// <summary>
/// Fragt den Vormonat eines Reports ab
/// </summary>
/// <param name="reportId">Identifizierende ID eines Report, von dem man den vorherigen Report sucht</param>
/// <returns>Report des Folgemonatss eines Reports</returns>
```

Abbildung 74 Screenshot des Kommentars GetReportVormonatDurchReportId

Schwierigkeit: dieselbe wie bei 2

[16] Ergänzung zum Kapitel 6.5.3

4: GetReportFolgeMonatDurchReportId

```
/// <summary>
/// Fragt den Vormonat eines Reports ab
/// </summary>
/// <param name="reportId">Identifizierende ID eines Report, von dem man den nachfolgenden Report sucht</param>
/// <returns>Report des Vormonats eines Reports</returns>
```

Abbildung 75 Screenshot des Kommentars GetReportFolgeMonatDurchReportId

Schwierigkeit:

Das Gegenteil von Punkt 2 und 3. Um den Report des Folgemonats zu bekommen, erhöhe ich den Monat immer um 1. Wenn man aber im Dezember steht, wäre der Folgemonat der 13, aber es gibt keinen 13. Monat.

Deshalb habe ich geprüft ob der Monat 12 ist und falls ja, wird das Jahr um 1 erhöht und der Monat auf 1 gesetzt. Aus Dezember 2017 wird Januar 2018.

5: VergleicheVormonatMitReport

```
/// <summary>
/// Vergleicht den übergebenen Report mit dem eigenen Report.ReportVormonat
/// </summary>
/// <param name="report">Report der vergleicht werden soll</param>
/// <returns>Ein Model mit den verglichenen Daten</returns>
```

Abbildung 76 Screenshot des Kommentars VergleicheVormonatMitReport

Hier wird der aktuelle Monat mit dem Vormonat verglichen. Daraus ergeben sich teilweise prozentuale Zahlen und teilweise auch Ganzzahlen. Diese werden in dieser Methode berechnet.

Schwierigkeit:

Teilweise musste ich zwei Ganzzahlen (INT) dividieren. Das geht aber nur wenn mal eine davon zu einem Double Konvertiert. Das habe ich zuerst nicht realisiert und mich gefragt wieso das Ergebnis falsch war.

6: GetMassnahmenDurchReportId

```
/// <summary>
/// Sucht nach Massnahmen eines Reports
/// </summary>
/// <param name="reportId">Identifizierende ID eines Reports</param>
/// <returns>Alle Massnahmen eines Reports</returns>
```

Abbildung 77 Screenshot des Kommentars GetMassnahmenDurchReportId

Diese Methode wird einerseits für den aktuellenMonat (report) und für den vorherigen Monat (report.reportVormonat) aufgerufen.

6.8 Diagramm und der API Call

So zum Ende der Realisierung wird auf die Knacknuss dieser Projektarbeit eingegangen. Das war aus meiner Sicht wirklich das Diagramm. Das Ziel war es, das Diagramm möglichst sauber zu lösen.

Dieses Kapitel teilt sich in 2 Teile ein. Der Clientseitige Teil mit dem Ajax Post und dem Zeichnen desDiagramms und in den Serverseitigen Teil mit dem Aufbereiten der Daten.

6.8.1 Serverseitiger Teil

```
public class DiagrammDaten
{
    public string[] Labels { get; set; }
    public DataSet[] Datasets { get; set; }
}

public class DataSet
{
    public string Label { get; set; }
    public int[] Data { get; set; }
    public string[] BackgroundColor { get; set; }
    public string BorderColor { get; set; }
    public string Type { get; set; }
    public bool Fill { get; set; }
}
```

Abbildung 78 Screenshot DiagrammDaten Model

Es wurde ein neues Model erstellt .Das Model repräsentiert die Struktur, die das Chart dann Clientseitig als JSON erwartet.

Labels sind die Texte im Chart unter den Säulen stehen. Zu jedem Label gibt es ein **DataSet**.

Dieses **DataSet** besteht aus:
Label (erscheint wenn man über den Teil der Säule hovert),
Data: die Daten selber
BackgroundColor (Hintergrundfarbe)

Type: Diagrammtyp (in unserem Fall Bar für Säule, Line für Liniendiagramm)

Fill: Boolean ob die Linie gefüllt sein soll (nur benötigt wenn Type = „Line“ ist)

Wie ich auf diese Werte gekommen bin:

Ich habe mir die JSON Struktur in jQuery genau angesehen. Danach habe ich interpretiert, dass Labels ein Stringarray ist und das datasets aus mehreren Objekten besteht.

Diese Objekte haben wieder eigene Werte wie **Type**, **Label**, **data**, **borderColor** und **fill**.

Aufgrund dieser Kenntnisse habe ich dann das Model Serverseitig erstellt.

```
data: {
    labels: ["January", "February", "March", "April", "May", "June", "July"],
    datasets: [{
        type: 'line',
        label: 'Traffic Vorjahr',
        data: [65, 10, 80, 81, 56, 85, 40],
        borderColor: 'black',
        fill: false
    }, {
        type: 'bar',
        label: 'Verweise',
        backgroundColor: "black",
        data: [65, 0, 80, 81, 56, 85, 40]
    }
```

Abbildung 79 Screenshot JSON Struktur clientseitig

Wenn diese Diagrammdaten nun korrekt mit Daten befüllt werden, muss nur noch das DiagrammDatenObjekt an den Client übergeben werden und dort richtig interpretiert werden.

Daten befüllen:

Ich habe einen Helper geschrieben: **DiagrammHelper**.
In diesem Helper werden die letzten Reports (maximal 7) ausgelesen und die Daten in die verschiedenen Arrays mit oder for Schleife gesetzt. Der Helper **DiagrammHelper.cs** ist im Anhang zu finden.

6.8.2 API richtig einrichten

Damit im IPA-Projekt korrekt mit APIs gearbeitet wird, habe ich mich informiert. Ich habe das NuGet **Microsoft.AspNet.WebApi** zu meinem Projekt hinzugefügt. Durch dieses ist es mir möglich gewesen ein eigenes Routing für API Calls zu definieren. Das geschieht in der **WebApiConfig.cs** und sieht folgendermassen aus:

```
config.Routes.MapHttpRoute(
    name: "DefaultApi",
    routeTemplate: "api/{controller}/{id}",
    defaults: new { id = RouteParameter.Optional }
);
```

Abbildung 80 Screenshot WebApiConfig Route

Wir gehen davon aus das der ApiController **DiagrammController** heisst:
Die Api wird also über **/api/Diagramm/id** aufgerufen.
So ist jetzt eine strikte Trennung von normalen Controllern und API Controllern möglich.

Global.asax muss die Route noch registrieren:

```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();
    WebApiConfig.Register(GlobalConfiguration.Configuration);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
}
```

Im Diagrammcontroller passiert nichts anderes als der Aufruf auf die DiagrammHelperKlasse und der Rückgabewert wird dann an den Client zurückgegeben. Den Clientseitigen Call werde ich gleich erläutern.

6.8.3 Clientseitiger Teil

Aufruf sobald das Dokument fertig ist (Document Ready)
Sobald das Dokument fertig geladen ist, wird die API aufgerufen.

```
//Kürzel für Dokument-Bereit Funktion
$(function () {
    DiagrammApiCall();
});
```

Abbildung 81 Screenshot Dokument bereit Funktion

Der Ajax Call
Hier wird die Serverseitige API aufgerufen und die aktuelle reportId aus der View übergeben.

```
function DiagrammApiCall() {
    var id = $("#currentId").text();
    $.ajax({
        type: "POST",
        url: "/api/Diagramm/GetDiagrammDaten",
        content: "application/json; charset=utf-8",
        dataType: "json",
        data: "=" + id,
        success: function (diagrammdaten) {
            zeichneDiagramm(diagrammdaten);
        }
    });
}
```

Rückmeldung vom Server

Abbildung 82 Screenshot AJAX Call

Diagramm zeichnen
Schlussendlich wird die Rückmeldung vom Server noch in die Konfiguration des Diagramms eingefügt

```
function zeichneDiagramm(daten) {
    var konfiguration = {
        type: 'bar',
        data: daten,
        options:...
    };
    var chartElement = document.getElementById("trafficDiagramm").getContext("2d");
    new Chart(chartElement, konfiguration);
}
```

Rückmeldung vom Server

Abbildung 83 Screenshot Diagramm zeichnen

6.9 Abschluss Realisierung

6.9.1 Globale Fehlerbehandlung

Die globalen Events (die über die ganze Applikation gehen) werden in .NET in der Global.asax.cs verwaltet. Zum Beispiel die Registrierung der Pfadmappingdateien ist auch im Global.asax zu finden. Es gibt dort einen Event für globale Fehler. Man kann diese abfangen und dann ausprogrammieren was geschehen soll.

In meinem Fall wird der Fehler einfach ausgegeben und ein Link um wieder zurück zu der Applikation zu kommen.

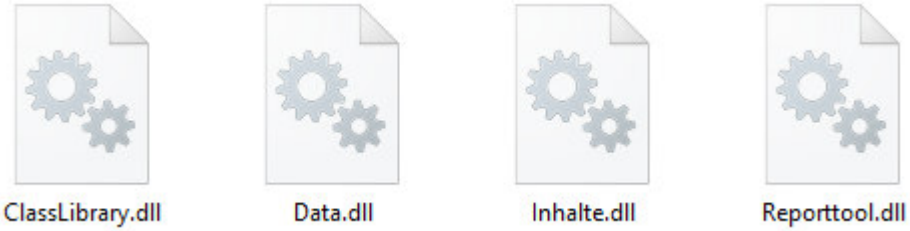
```
// Fängt globale Fehlermeldungen ab
protected void Application_Error(object sender, EventArgs e)
{
    var exc = Server.GetLastError();
    var stringBuilder = new StringBuilder("<b>")
        .Append(ResourceHelper.GetText("fehlermeldungTitel"))
        .Append("</b> <br />")
        .Append(exc.Message)
        .Append("<br /><br />")
        .Append(exc.StackTrace)
        .Append("<br /><br />")
        .Append("<a href='/'>Zurück</a>");

    //Gibt den Fehler aus, damit der Admin den Fehler beheben kann.
    Response.Write(stringBuilder);
    Server.ClearError();
}
```

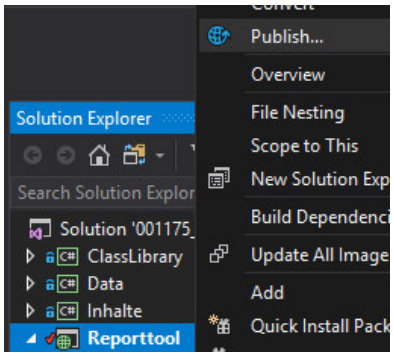
Abbildung 84 Screenshot globale Fehlerbehandlung

6.9.2 Projekt veröffentlichen

Bei einer .NET Applikation gibt es mehrere Stadien eines Projekts. Es gibt die unkompilierte Version, die man lokal benutzt um zu entwickeln und es gibt die kompilierte Version die auf der Liveumgebung benutzt wird. In der kompilierten Version ist der Quellcode nicht ersichtlich. Das dient zur Sicherheit , dass niemand den Quellcode einsehen kann. Die Klassen werden im bin Ordner zu DLL Dateien kompiliert. (Dynamic Link Library)[17]. Diese dll Files enthalten dann den ausführbaren Code, sind aber nicht lesbar.



Diese kompilierte Version erhält man wenn man das Projekt über Visual Studio veröffentlicht. Man macht einen Rechtsklick auf die Applikation und wählt „Publish"



Die veröffentlichten Daten kopiert man dann auf den Server wo es laufen soll.

Dort wird ein neuer IIS Eintrag gemacht damit die Applikation an eine Domain gebunden wird (customers.chili.ch)

Die Subdomain customers.chili.ch muss dann noch mit einem DNS-Eintrag an die IP des Servers gebunden werden.

Danach ist die Applikation lauffähig.

Abbildung 85 Screenshot Publish mit Visual Studio

[17] Informationen über DLL Dateien: https://de.wikipedia.org/wiki/Dynamic_Link_Library

7 Kontrollieren

Während der ganzen Realisierung wurde die Applikation ab und zu getestet und Verbesserungen bereits im Vorfeld vorgenommen.

Im Kapitel kontrollieren wird die Applikation final kontrolliert. In erster Linie heisst das die Applikation wird getestet. Die Testfälle werden duchgegangen und in einem Testprotokoll festgehalten.

7.1 Das Testing

7.1.1 Testprotkoll

Die definierten Testfälle aus Phase 4.3 Testkonzept werden hier durchgetestet. Fehlgeschlagene Tests werden überprüft und wenn möglich bereinigt und danach erneut getestet.

Testfall-Nr.	Resultat	Bemerkung
1		Die Seite wird korrekt angezeigt. Man kommt direkt auf die Loginseite
2		Die Seite wird nicht gelistet, da die Meta Robots Tags gesetzt sind (NO-INDEX, NO-FOLLOW)
3		Applikation ist auf allen Browsern fehlerfrei. Bei der Progammierung wurde darauf geachtet, dass die verwendeten Techniken mit allen Browsern kompatibel sind.
4		Das Login funktioniert wie definiert. Man kommt auf die Übersichtsseite.
5		Die IP die geloggt wird, wenn man sich lokal anmeldet ist „::1“, das ist wenig aussagekräftig und sollte optimiert werden.
6		Wenn man versucht über das Login eine SQL Injection zu machen, werden die Eingaben abgefangen und man erhält eine Fehlermeldung „Passwort falsch“.
7		Die Session wird nach 15 Minuten beendet.
8		A4 wird fehlerfrei ausgedruckt
9		Die Session wird beendet und man kommt zurück zur Loginseite
10		Titel wird korrekt angezeigt. Anstatt für den Kundennamen haben wir uns für den URL_Namen entschieden
11		Liste mit Massnahmen wird korrekt dargestellt
12		Kommentar wird korrekt dargestellt
13		Kennzahlen sind vorhanden und korrekt
14		Massnahmen sind ersichtlich.
15		Ziele werden korrekt dargestellt
16		Tabelle wird korrekt dargestellt und enthält die vorgegebenen Werte.
17		Balkendiagramm wird korrekt und leserlich dargestellt
18		Fusszeile ist korrekt und wird dargestellt

7.2 Fehlgeschlagene Testfälle

7.2.1 Testfall 5

Problem:

Wenn der lokale User ist wird die IP ::1 gespeichert. Das ist unverständlich.

Lösung:

Ich werde diesen Testfall abfangen und dann ::1 durch localhost ersetzen.

Test:

5		Die IP die geloggt wird, wenn man sich lokal anmeldet ist nun localhost.

8 Auswerten

In der letzten Phase auswerten, geht es darum das Projekt und die Vorgehensweise im Projekt kritisch zu hinterfragen. Die Schwierigkeiten und das dadurch gelernte werden dargestellt.

8.1 Hindernisse im Projekt und die Erkenntnis daraus

Während der Umsetzung der praktischen Arbeit bin ich ein paar Mal an Hindernisse gestossen. Ich habe immer versucht das Problem dann aus einer anderen Perspektive zu betrachten. Oftmals konnte so eine Lösung gefunden werden.

8.1.1 Auswahl des Models

Am Anfang wollte ich eigene Models schreiben und diese mit den Models der Datenbank füllen. Im Verlauf der Arbeit ist mir dann aufgefallen, dass ich direkt die Datenbankmodelle benutzen kann und diese mit Teilklassen erweitern kann. Auf diese Erkenntnis bin ich stolz und das ist eine Möglichkeit, die ich in Zukunft sicher oft verwenden werde in weiteren Projekten.

Erkentnisse daraus

Wie man mit Teilklassen arbeitet und so Models erweitern kann

8.1.2 Saubere Validierung mit DataAnnotations

So war es für mich auch neu die Data Annotations nicht direkt ins Model zu schreiben sondern diese in Metaklassen auszulagern und mit Teilklassen dann zu integrieren. Das ist eine sehr saubere Methode und ergibt eine schöne Projektstruktur

Erkentnisse daraus

Mit Metaklassen Data Annotations auslagern und über Teilklassen danach integrieren.

8.1.3 Die Daten in der Datenbank richtig auslesen in einer Datenbankview

Das erstellen und verwalten der Datenbanksicht im SQL Management Studio hat mich einiges an Nerven gekostet. Womit war dieser Fremdschlüssel nochmal verknüpft? Gibt es irgendwo noch eine Verbindungstabelle zu diesen Werten? Eine kleine Anpassung an der Datenbanksicht und plötzlich spuckt diese keine Daten mehr aus. Problem suchen, Problem finden und danach das Problem beseitigen. So habe ich oftmals einiges an Zeit verloren, die ich vielleicht anderswo besser hätte verwenden können

Erkenntnisse daraus

Ich kenne mich nun besser mit der internen Datenbank von Chili aus. Ich habe meine Kenntnisse im SQL Management Studio verbessert.

8.1.4 Userdaten abfangen und Auswerten

Die IP und den UserAgent des Kunden abzufragen war für mich neu. Ich habe mehrere Möglichkeiten gesucht und mich dann für eine Entschieden.

Erkenntnisse daraus

Umgang mit dem Request Property gelernt.

8.1.5 Globales Errorhandling

Ein sehr wichtiger Aspekt, den ich aber bisher in meinen Projekten oftmals nicht beachtet habe ist das globale Errorhandling. Man geht halt selber oft davon aus, dass es keine weiteren Fehler mehr gibt, als die die man selber gefunden hat. Aber es kann halt immer sein dass irgendwo etwas übersehen wurde. Dafür gibt es das Globale Errohandling, welches alle übrigen Fehlermeldungen im Code abdeckt. Sinnvoll wäre es zum Beispiel die Fehlermeldungen zu loggen, so hat man immer eine Übersicht ob die Applikation sauber läuft oder ob Fehler auftreten.

Erkentnisse daraus

Globales Errorhandling ist sehr wichtig. Umgang mit dem globalen Errorhandling.

8.1.6 Arbeiten mit WebAPI

Ich habe schon vor meiner IPA teilweise Webmethoden geschrieben. Das habe ich aber in normalen Controllern gemacht und dann diese einfach über AJAX angesprochen. Der saubere und schnellere Weg geht über das WebAPI NuGet von .NET.

Erkentnisse daraus

Ich habe im Rahmen meiner IPA gelernt wie man in .NET eine saubere WebAPI erstellt und damit arbeitet.

8.1.7 Umgang mit Resourcefiles

Der Umgang mit Resourcefiles war mir vor meiner IPA nur am Rande bekannt. Ich bin sehr froh, dass ich mal selber ein Projekt mit Resourcefiles aufgesetzt habe.

Erkenntnisse daraus

Resourcefiles erstellen und Werte auslesen, zusätzlich gelernt wie man Daten aus einem Resourcefile bei den Data Annotations benutzen kann.

8.1.8 Die saubere Projektstruktur

Bei diesem Hindernis bin ich selber nicht ganz unschuldig. Es war mir persönlich ein grosses Anliegen dass ich mein Abschlussprojekt in der Struktur des Programms sauber gestalte. Oftmals habe ich Sachen wieder gelöscht, anderswo erstellt und dann trotzdem wieder verschoben. Ich habe ausgelagert wo ich nur irgendwie konnte und bin jetzt am Ende doch sehr zufrieden mit der Projektstruktur, wie sie jetzt steht.

Denn nur wenn die bisherige Struktur sauber ist, ist man bei zukünftigen Änderungen motiviert, diese auch sauber umzusetzen. Zu oft habe ich es im Geschäft erlebt, dass irgendwo Änderungen nur so schnell und unsauber gemacht wurden und man danach den halben Code wieder auffrischen musste, damit der Code wirklich wartbar ist.

Erkenntnisse daraus

Eine saubere Projektstruktur ist sehr zeitaufwendig. Ich würde aber behaupten es lohnt sich, da der Code so schön wartbar ist und in Zukunft erweitert werden kann. Auf diese Erweiterungsmöglichkeiten gehe ich übrigens gleich im nächsten Kapitel ein.

8.2 Mögliche Erweiterungen ausserhalb des Projekts

Einige Male habe ich es im Projekt erlebt, dass ich sagen musste: Dieses Feature wäre wirklich noch ziemlich praktisch und sehr nützlich doch es sprengt einfach den Rahmen meiner Abschlussarbeit. Folgend eine kurze Auflistung dieser Erweiterungen:

8.2.1 Passworthashing

Dieser Punkt wurde mit der Geschäftsleitung besprochen und wird später folgen. Denn ich finde es wichtig dass Passwörter verschlüsselt sind. Man weiss ja nie wann Daten aus der Datenbank gestohlen werden. Dann wären alle Passwörter einfach frei einsehbar. Das kann man mit dem Passworthashing verhindern.

8.2.2 Passwort zurücksetzen Funktion

Im MockUp habe ich es zwar dargestellt, habe aber dann doch sagen müssen dass ich den Fokus besser auf die vorgeschriebene Anforderungen setzen will. Jetzt ist es halt einfach so, dass ein Kunde sich bei uns über Mail melden muss, wenn er seine Zugangsdaten vergisst.

8.2.3 AdWords API benutzen

Das Reporttool ist ganz nützlich, doch ein grosser Aufwand wird es sein die ganzen Daten der Kunden in die Datenbank einzutragen. Die AdWords Berichte bestehen im Moment einfach in Dokumentform.

Mit dem AdWords API könnte man bestimmt irgendwie die Daten aus dem AdWords auslesen und dann in unsere Datenbank schreiben.

Ich habe mich nicht tiefer mit diesem Thema befasst, da es den Rahmen meiner IPA um ein vielfaches sprengen würde, doch wäre es sicher eine Überlegung wert.

8.2.4 Zwei beliebige Reports vergleichen

Eine weitere coole Möglichkeit die man dem Tool noch hinzufügen könnte ist eine konfigurierbare Vergleichsmöglichkeit. Man könnte mehrere Reports auswählen und die Prozentualen Unterschiede darstellen lassen.

Dies wäre sicher für den einen oder anderen Kunden interessant, da er dann sein aufgewendetes Budget und die daraus resultierenden Nutzerzahlen noch genauer im Überblick hätte.

8.2.5 Automatische E-Mail Benachrichtigungen

Jeden Monat werden neue Berichte von uns in der Datenbank erfasst. Ein weiteres spannendes Feature wäre eine automatische E-Mail Benachrichtigung für den Kunden, sobald ein neuer Report aufgeschaltet wurde.

So könnte Zeit gespart werden und es macht einen professionellen Eindruck, wenn solche Lösungen erstellt werden.

8.3 Vergleich bisherige Lösung / neue Lösung

Die bisherige Lösung war so geregelt, dass die Reports selber in Word und in Excel erstellt wurden und dann dem Kunden als PDF zugesandt wurden.

Der Vorteil der neuen Lösung ist sicher, dass einiges an Zeit eingespart werden kann. Für den Kunden besteht eine bessere und dynamischere Übersicht. Anstatt sich durch mehrere Reports in PDF Form zu kämpfen, kann der Kunde nun bequem via Buttonklick zwischen den einzelnen Reports hin und her wechseln und kann sich so seine wichtigsten Kennzahlen ansehen.

Ein weiter Vorteil ist das Filtrieren des Diagramms. Es können zum Beispiel die Verweise ausgeblendet werden, wenn man nur die Suchmaschinen und den Sonstigen Traffic vergleichen will.

Ich denke dass wir die neue Lösung gut einsetzen können und sie bei unseren Kunden Gefallen finden wird.

8.4 Reflexion der Projektmethode

IPERKA hat sich als nützliche Projektmanagementmethode erwiesen. Mir ist erst bei diesem Projekt wieder bewusst geworden, wie wichtig und nützlich die Planungsphase ist. Ich hätte nie gedacht dass ich so schnell mit der Realisierung durchkomme. Das war nur möglich, weil ich im Vorfeld alles genau geplant habe. Mit der Einteilung der Aufgaben im Zeitplan wusste man jeden Tag genau was man am diesem Tag macht und was das Tagesziel ist.

Man musste sich nie fragen, was könnte man jetzt noch machen. Ich war durchgehend voll beschäftigt. Die Zeitplanung hat sich als sehr genau erwiesen, ich habe zwar die Dokumentation teilweise etwas herausgeschoben und musste am Ende noch einiges ergänzen, doch ich konnte mich gut an die Vorgaben halten.

Für mich war es teilweise sehr schwierig weiterzumachen, wenn ein Problem bestand, das ich nicht lösen konnte. Ich bin ein Problemlöser und will immer zuerst das Problem lösen und erst dann mit dem Rest der Arbeit fortfahren. Doch so funktioniert es wärhend der IPA nicht. Ich musste mich dann überwinden zuerst andere Sachen zu erledigen. Dann konnte ich mich später, wenn die Zeit es zugelassen hat, mit dem Problem befassen.

Fazit

Im Grossen und Ganzen bin ich sehr zufrieden mit dem Ergebnis meiner Arbeit. Ich würde vieles wieder genauso machen wie ich es in meiner IPA gemacht habe. Ich werde wahrscheinlich noch oft in meiner IPA Dokumentation irgendwas nachschauen und abschauen.

Schlusswort

Nach zehn Tagen individueller praktischer Arbeit blicke ich auf eine lehrreiche, spannende aber auch sehr intensive Zeit zurück. Ich konnte mein gesamtes gesammeltes Wissen aus 2 Jahren Schulzeit und 20 Monaten Praktikum ausschöpfen und zeigen was ich kann. Ich habe immer versucht Techniken zu wählen auf denen ich schon Erfahrung habe um mein bisheriges Wissen zu festigen und ein befriedigendes Ergebnis zu erzielen.

Diese praktische Arbeit ist das grosse Ziel der Ausbildung zum Informatiker und man denkt schon seit dem ersten Lehrjahr an diese Arbeit. Diese Arbeit jetzt hinter mir zu haben, ist eine grosse Erleichterung und doch war es eine schöne Zeit mal zehn Tage am Stück ungestört und alleine an einem Projekt zu arbeiten. Natürlich arbeite ich auch sehr gerne im Team und freue mich wieder auf vielseitige Aufträge durch Chili.

Beim Durchführen dieser Arbeit wurde mir bewusst, wie riesengross der Aufwand ist so ein mittelgrosses Projekt alleine zu realisieren. Wenn man wirklich von der Aufgabenstellung, über das MockUp, hin zum Frontend bis und mit dem Backend alles selber macht ist das ein sehr grosser Aufwand. Jetzt bin ich stolz auf das Ergebnis, welches voll funktionsfähig ist und alle Anforderungen durch Chili erfüllt.

Ich hoffe dass das Tool bald im Tagesgeschäft eingesetzt wird und unsere Kunden genau so viel Spass damit haben, wie ich bei der Entwicklung hatte.

Zusätzlich freue ich mich auch eventuelle Erweiterungen und Verbesserungen, denn eine Software ist nie wirklich fertig, es gibt immer Verbesserungs- und Erweiterungsmöglichkeiten.

Die Arbeit hat mir gezeigt, wie viel Spass und wie viel Abwechslung man doch als Informatiker in der Softwareentwicklung hat. Ich werde in Zukunft sicher auf diesen Berufszweig weiterarbeiten und mich laufend weiterbilden, damit ich den Anschluss nicht verpasse.

Persönliche Bilanz

Mit der Umsetzung des Tools und dem Funktionsumfang bin ich sehr zufrieden. Es funktioniert soweit fehlerlos und alle Bedingungen der Geschäftsleitung sind erfüllt.

Schlusssatz

Meine individuelle praktische Arbeit hat mir auf meinem beruflichen Weg als Informatiker viele neue Kenntnisse gezeigt und ist ein Meilenstein in meinem Lebenslauf.

Danksagung

Ich bedanke mich herzlich bei den beteiligten meiner IPA. Ein grosser Dank geht an meinen Fachvorgesetzen Ignaz Walgis, der mich bei der ganzen Arbeit immer unterstützt hat und mich optimal auf das Projekt vorbereitet hat. Ich möchte mich auch speziell bei meiner Hauptexpertin Ursula Reinhard bedanken, ich wurde sehr gut über den Ablauf der IPA informiert und hat mir auch den ein oder anderen Tipp gegeben.

Zudem möchte ich mich bei den Mitarbeitern bedanken die mir mit Tipps zur IPA zur Seite gestanden sind und die mich zusätzlich zum Ablauf informiert haben. Diesen Mitarbeitern verdanke ich auch ein Grossteil der Erfahrung die ich im Praktikum bei der Chili Solutions GmbH in Zürich gesammelt habe.

Ein Dank geht an die ganze Firma Chili Solutions GmbH für die Möglichkeit das Praktikum hier zu absolvieren und die Möglichkeit meine IPA in der Firma durchzuführen.

Danke vielmals!
Hubert Thalmann

Glossar/Wortverzeichnis

Bezeichnung	Bedeutung
.NET	Sammelbegriff für mehrere von Microsoft herausgegebene Software-Plattformen
API	Programmierschnittstelle, genauer Schnittstelle zur Anwendungsprogrammierung
CRM	Customer relationship management, Kundenverwaltung
CSS	Cascading Stylesheet: definiert das Styling einer Website
Datenbank	Eine Datenbank ist ein System zur elektronischen Datenverwaltung. In Ihr können die Tabellen, Spalten und auch Beziehungen verwaltet werden
DLL	Kompilierte Datei die den ausführbaren Quellcode enthält
Entity Framework	Framework von Microsoft für objektrelationale Abbildung (Verbindung zwischen einer objektorientierten Programmiersprache und einer relationalen Datenbank).
GUI	Graphical User Interface, grafische Benutzeroberfläche
HTML	Hypertext Markup Langugage: definiert die Struktur einer Website
IIS	Internet Information Services (Webserver/Dienstplattform, kann .NET Applikationen ausführen)
IPA	Individuelle praktische Arbeit
IPERKA	Benutzer Projektmanagementmethode
JS	Javascript, eine Sprache um clientseitig Seiten zu manipulieren
Methode	Anderes Wort für Funktion. Funktionen in der Programmierung
NuGet Packet	Zusatzpakete für Visual Studio
SQL	Structured Query Language, Sprache um Datenbank zu bedienen
VSTS	Visual Studio Team Services

Quellverzeichnisse

Abbildungsverzeichnis

Bilderverzeichnis mit Bezug auf Abbildung

Screenshots werden im Fremdbildverzeichnis bewusst weggelassen. Diese wurden alle von mir selber erstellt und eingefügt.

Abbildung Nr.	Quelle
1	https://pixabay.com/en/financial-analytics-blur-business-2860753/
2	Hubert Thalmann in Word
3	Hubert Thalmann in Visio
4	Hubert Thalmann in Visio
5	Hubert Thalmann in Visio
6	Hubert Thalmann in Visio
7	Hubert Thalmann in Visio
8	Hubert Thalmann in Visio
9	Hubert Thalmann in Balsamiq
10	Hubert Thalmann in Balsamiq
11	Hubert Thalmann in Balsamiq
12	http://www.codestring.co.uk/site/wp-content/uploads/2016/07/VisualStudio.png
13	https://cdn.windowsreport.com/wp-content/uploads/2017/12/MSSQL-server.png
14	https://www.gigshost.net/technology/iis-web-server-hosting.php
15	http://fonow.com/Images/2017-08-30/fix/2/66372-20170204173339573-58644925.jpg
16	https://blogs.microsoft.com/uploads/2012/08/8867.Microsoft_5F00_Logo_2D00_for_2D00_screen.jpg
21	http://takeupskills.com/wp-content/uploads/2017/02/asp.net-MVC.png
22	https://www.visualstudio.com/wp-content/uploads/2017/06/Visual-Studio-Team-Services.png
26	https://cdn-images-1.medium.com/max/1344/1*J8khcB_mGxI_TTVTaTUqsg.png
33	http://www.sanjaywebdesigner.com/articles/wp-content/uploads/2017/11/bootstrap-4-courses-768x432.png
51	Hubert Thalmann in Visio
70	Hubert Thalmann in Word

Informationsquellen

Fussnote Nr.	Quelle
1	https://pkorg.ch/
2	https://pkorg.ch/
3	https://support.office.com/de-de/article/spaltenbreite-und-zeilenh%C3%B6he-%C3%A4ndern-2c002bd2-3b9c-4561-a99b-cbdcb0f69bc0
4	https://de.wikipedia.org/wiki/White-Box-Test
5	https://de.wikipedia.org/wiki/Visual_Studio
7	https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/database-first-development/enhancing-data-validation
8	https://stackoverflow.com/questions/2451150/dataannotations-and-resources-dont-play-nicely
9	https://stackoverflow.com/questions/1907195/how-to-get-ip-address
10	https://en.wikipedia.org/wiki/Circular_dependency
17	https://de.wikipedia.org/wiki/Dynamic_Link_Library

Anhang

Quellcode

Global.asax.cs

```
// ============================================================
// Erstelldatum:        06.04.2018
// Ersteller:           Chili/hth
// Zweck:               Verwaltet die Globalen Events
//                      wird automatisch generiert
// ============================================================
using System;
using System.Text;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Routing;
using Inhalte;
namespace Reporttool
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            WebApiConfig.Register(GlobalConfiguration.Configuration);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
        }
        // Fängt globale Fehlermeldungen ab
        protected void Application_Error(object sender, EventArgs e)
        {
            var exc = Server.GetLastError();
            var stringBuilder = new StringBuilder("<b>")
                .Append(ResourceHelper.GetText("fehlermeldungTitel"))
                .Append("</b> <br />")
                .Append(exc.Message)
                .Append("<br /><br />")
                .Append(exc.StackTrace)
                .Append("<br /><br />")
                .Append("<a href='/'>Zurück</a>");
            //Gibt den Fehler aus, damit der Admin den Fehler beheben kann.
            Response.Write(stringBuilder);
            Server.ClearError();
        }  }  } //← Klammern sind aus Platzgründen in der Doku so gesetzt
```

RouteConfig.cs

```
// ============================================================
// Erstelldatum:        06.04.2018
// Ersteller:           Chili/hth
// Zweck:               Verwaltet die Routen der Controller
//                      Wird automatisch generiert
// ============================================================
using System.Web.Mvc;
using System.Web.Routing;

namespace Reporttool
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            //Route für die Standardcontroller wird definiert
            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Login", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

WebApiConfig.cs

```
// ============================================================
// Erstelldatum:        10.04.2018
// Ersteller:           Chili/hth
// Zweck:               Verwaltet die Routen der APIs
//                      Wird automatisch generiert
// ============================================================
using System.Web.Http;
using Newtonsoft.Json.Serialization;
namespace Reporttool
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            config.Formatters.JsonFormatter.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();
            //Route für die API wird definiert
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

diagramm.js

```
/* ============================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Enthält die jQuery Logik für das Diagramm
//                      Macht den API Call auf den Webservice
============================================================ */
/// <reference path="lib/jquery.min.js" />

//Kürzel für Dokument-Bereit Funktion
$(function () {
    DiagrammApiCall();
});

function zeichneDiagramm(daten) {
    var konfiguration = {
        type: 'bar',
        data: daten,
        options: {
            scales: {
                xAxes: [
                    {
                        stacked: true,
                        ticks: {
                            fontSize: 18
                        }
                    }
                ],
                yAxes: [
                    {
                        stacked: true,
                        ticks: {
                            fontSize: 18
                        }
                    }
                ]
            },
            legend: {
                display: true,
```

```
                labels: {
                    fontSize: 18
                }
            }
        }
    };
    var chartElement = document.getElementById("trafficDiagramm").getContext("2d");
    new Chart(chartElement, konfiguration);
}

function DiagrammApiCall() {
    var id = $("#currentId").text();
    $.ajax({
        type: "POST",
        url: "/api/Diagramm/GetDiagrammDaten",
        content: "application/json; charset=utf-8",
        dataType: "json",
        data: "=" + id,
        success: function (diagrammdaten) {
            zeichneDiagramm(diagrammdaten);
        },
        error: function () {
        }
    });
}
```

filter.js

```
/* ======================================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Enthält die jQuery Logik für das Filtrieren
//                      auf der Übersichtsseite
====================================================================== */
/// <reference path="lib/jquery.min.js" />

//initialisiert das isotope-Grid und definiert die Sortierdaten
$(function () {
    var $grid = $(".grid").isotope({
        itemSelector: '.grid-item',
        percentPosition: true,
        getSortData: {
            datedesc: '.datedesc',
            besuche: '.besuche',
            dateasc: '.dateasc',
        },
        sortBy: 'datedesc',
        sortAscending: {
            datedesc: false,
            dateasc: true,
            besuche: false
        }
    });

    //Vorgefertigte Funktion von Isotope zum sortieren
    $('.sort-by-button-group').on('click', '.sortBtn', function () {
        var sortValue = $(this).attr('data-sort-value');
        $grid.isotope({ sortBy: sortValue });
    });

    //Setzt die aktiv Klasse, wenn ein Button geklickt wird und entfernt bei allen anderen die Klasse
    $('.sort-by-button-group').each(function (i, buttonGroup) {
        var $buttonGroup = $(buttonGroup);
        $buttonGroup.on('click', '.sortBtn', function () {
            $buttonGroup.find(".aktiv").removeClass('aktiv');
            $(this).addClass('aktiv');
```

```
        });
    });

    //Beim Ändern des Suchfeldes wird meine Filterfunkiton aufgerufen
    $("input.suchen").on('input propertychange paste', function () {
        $grid.isotope({
            filter: function () {
                return textBoxFilter(this);
            }
        });
    });

    function textBoxFilter(report) {
        var eingabeText = $("input.suchen").val().toLowerCase();

        // Text der mit der Eingabe verglichen werden soll wird ausgelesen
        var kennzahlen = $(report).find('.filterValue').text().toLowerCase();
        var titel = $(report).find('.filterTitle').text().toLowerCase();

        //true zurückgeben und Element anzuzeigen, false zum verstecken
        var elementAnzeigen = (beinhaltetWort(kennzahlen, eingabeText) || beinhaltetWort(titel, eingabeText));
        return elementAnzeigen;
    }

    //testet ob das Wort innerhalb eines anderen Strings beinhaltet ist
    function beinhaltetWort(myString, myWord) {
        return myString.indexOf(myWord) > -1;
    }
});
```

reportdetail.js

```
/* ======================================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Ordnet die Tabelle in die richtige Reihenfolge
====================================================================== */
$(function () {
    $("table").each(function () {
        var $this = $(this);
        var newrows = [];
        $this.find("tr").each(function () {
            var i = 0;
            $(this).find("td,th").each(function () {
                i++;
                if (newrows[i] === undefined) {
                    newrows[i] = $("<tr></tr>");
                }
                newrows[i].append($(this));
            });
        });
        $this.find("tr").remove();
        $.each(newrows, function () {
            $this.append(this);
        });
    });
});
```

LoginController.cs

```
// ============================================================
// Erstelldatum:        06.04.2018
// Ersteller:           Chili/hth
// Zweck:               Verwaltet alle Aktionen auf der
//                      Loginseite
// ============================================================
using System;
using System.Web.Mvc;
using Data;
using System.Linq;
using ClassLibrary;
using Inhalte;

namespace Reporttool.Controllers
{
    public class LoginController : Controller
    {
        // GET: Login
        public ActionResult Index()
        {
            return View("Login");
        }

        [HttpPost]
        public ActionResult Login(TAB_Adresse loginDaten)
        {
            //Aufbereitung der Daten für das History Logging
            var historyDaten = new TAB_Adresse_History_Login
            {
                DAT_DatumZeit = DateTime.Now,
                NUM_AdresseId = loginDaten.NUM_Nummer,
                TXT_IP = Request.UserHostAddress,
            TXT_UserAgent = Request.UserAgent
            };
            if (ModelState.IsValid)
            {
                using (var dbContext = new marketingEntities())
                {
```

```
            //Prüft ob es einen Kunden mit dieser Kundennummer und Passwort gibt
            var myUser = dbContext.TAB_Adresse.FirstOrDefault(e =>
                e.NUM_Nummer == loginDaten.NUM_Nummer && e.TXT_Passwort == loginDaten.TXT_Passwort);

            if (myUser != null)
            {
                //Setzen der Sessionvariabeln
                Session["KundenNr"] = myUser.NUM_Nummer.ToString();
                Session.Timeout = 15;

                //History Text Login erfolgreich
                historyDaten.TXT_Text = ResourceHelper.GetValidierungsmeldung("loginErfolgreich");
                //History in DB schreiben
                HistoryHelper.Schreiben(historyDaten);

                //Weiterleitung zu Report-Übersicht
                return RedirectToAction("Index", "ReportUebersicht");
            }

            ViewBag.Error = ResourceHelper.GetValidierungsmeldung("falschesPasswort");

            //History Text Passwort falsch
            historyDaten.TXT_Text = ResourceHelper.GetValidierungsmeldung("falschesPasswort");
            //History in DB schreiben
            HistoryHelper.Schreiben(historyDaten);
            return View("Login");
        }

    }

    //Workaround für Validierungsproblem wenn der Input keine Nummer ist, ist ein bekannter Bug in MVC 5
    //Quelle: http://www.iminfo.in/post/change-message-the-value-is-not-valid-for-number-mvc-workarond

    if (ModelState["NUM_Nummer"].Errors.Count == 1 && ModelState["NUM_Nummer"].Errors[0].ErrorMessage
            .Contains("is not valid for NUM_Nummer"))
    {
        ModelState["NUM_Nummer"].Errors.Clear();
        ModelState["NUM_Nummer"].Errors.Add(ResourceHelper.GetValidierungsmeldung("nummerFehler"));
    }
```

```
            //History Text Nummerfehler
            //History in DB schreiben
            historyDaten.TXT_Text = ResourceHelper.GetValidierungsmeldung("nummerFehler");
            HistoryHelper.Schreiben(historyDaten);
            return View("Login");
        }


    }

}
```

LogoutController

```
// ================================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Loggt den Kunden aus, beendet die Session
// ================================================================
using System.Web.Mvc;

namespace Reporttool.Controllers
{
    public class LogoutController : Controller
    {
        public ActionResult Index()
        {
            //Session wird beendet
            Session.Clear();
            Session.Abandon();
            Session.RemoveAll();
            return RedirectToAction("Index","Login");
        }
    }
}
```

ReportDetailController.cs

```
// ============================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Befüllt das Model mit Daten um sie danach
//                      auf der Reportdetailseite anzuzeigen
// ============================================================

using System;
using System.Web.Mvc;
using ClassLibrary;

namespace Reporttool.Controllers
{
    public class ReportDetailController : Controller
    {
        public ActionResult Index()
        {
            return RedirectToAction("Report");
        }
        public ActionResult Report(int id)
        {
            //Prüft ob User eingeloggt ist
            if (LoginHelper.IstEingeloggt())
            {
                int reportId = id;
                //Holt den Report, gibt den optionalen KundenNr Parameter mit zur Prüfung ob der Report zum Kunde gehört.
                var model = ReportsHelper.GetReportDurchReportId(reportId, Convert.ToInt32(Session["KundenNr"]));

                //Prüft ob der Report wirklich von diesem Kunde ist, falls nein wird man auf die Übersichtsseite weitergeleitet.
                if (model.NUM_Kunde == null) return RedirectToAction("Index", "ReportUebersicht");

                //Holt die Reports der vergangegen Monate
                model.VorherigeReports = ReportsHelper.GetVorherigeReports(model, 6);

                //Holt den Report des Vormonats
                model.ReportVormonat = ReportsHelper.GetReportVormonatDurchReportId(reportId);

                //Holt den Report des Folgemonats
                model.ReportFolgemonat = ReportsHelper.GetReportFolgeMonatDurchReportId(reportId);
```

```
                //Berechnet die Änderungen vom aktiven Monat zum Vormonat
                model.AenderungenVormonat = ReportsHelper.VergleicheVormonatMitReport(model);

                //Holt die aktuelle Massnahmenliste
                model.MassnahmenListe = ReportsHelper.GetMassnahmenDurchReportId(model.ID);

                //Holt die Massnahmenliste vom Vormonat
                model.MassnahmenVormonatListe = ReportsHelper.GetMassnahmenDurchReportId(model.ReportVormonat.ID);

                //Gibt die Daten an die View und zeigt den ReportDetail Bericht an
                return View("ReportDetail", model);
            }

            //Wird zum Login Screen weitergeleitet
            return RedirectToAction("Index", "Login");
        }
    }
}
```

ReportUebersichtController.cs

```
// ============================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Befüllt das Model mit Daten um sie danach
//                      auf der Reportübersichtsseite anzuzeigen
// ============================================================
using System;
using System.Web.Mvc;
using ClassLibrary;

namespace Reporttool.Controllers
{
    public class ReportUebersichtController : Controller
    {
        public ActionResult Index()
        {
            //Prüft ob der Kunde eingeloggt ist
            if (LoginHelper.IstEingeloggt())
            {
                //Holt eine Liste aller Reports eines Kunden
                var reportsList = ReportsHelper.GetReportsDurchKundenNummer(Convert.ToInt32(Session["KundenNr"]));

                //Prüft ob mehr als ein Report gefunden wurde
                if (reportsList.Count > 1)
                {
                    //Gibt die Liste an die View weiter und zeigt die View an
                    return View("Uebersicht", reportsList);
                }

                //Leitet direkt auf die Reportdetailseite weiter
                return RedirectToAction("Report", "ReportDetail", reportsList[0].ID);
            }
            //Leitet den Kunden weiter zur Loginansicht
            return RedirectToAction("Index", "Login");
        }
    }
}
```

Master.cshtml

```
@*  ============================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Beinhaltet den Rahmen der Website
// ============================================================*@
@using Inhalte

@{
    Layout = null;
}
<!DOCTYPE html>
<html moznomarginboxes mozdisallowselectionprint>
<head>
    <meta name="robots" content="noindex, nofollow">
    <meta name="viewport" content="width=device-width" />
    <link href="https://fonts.googleapis.com/css?family=Crimson+Text:400i|Montserrat:400,700,800" rel="stylesheet">

    <link href="~/_css/lib/bootstrap.min.css" rel="stylesheet" />
    <link href="~/_css/style.css" rel="stylesheet" />
    <link href="~/_css/print.css" rel="stylesheet" media="print" />
    <title>@ResourceHelper.GetText("globalerSeitenTitel")</title>
</head>
<body class="@ViewBag.BodyClass">
    @RenderBody()

    <footer class="paddingBottomSmall noprint">
        <div class="container">
            <div class="row">
                <div class="col-3">
                    <h2>Chili Solutions GmbH</h2>
                    <address>
                        Birmensdorferstrasse 470<br />
                        8055 Zürich<br />
                        Tel. +41 44 315 90 00<br />
                        info@chili.ch<br />
                    </address>
                </div>
                <div class="col-5 zertifizierungen">
```

```
                    <h2>Zertifizierungen</h2>
                    <img src="~/_img/certificates.png" />
                </div>
                <div class="col-4 website">
                    <h2>Chili Website</h2>
                    <a href="https://www.chili.ch" target="_blank">
                        <div class="customButton">
                            Zur Website
                        </div>
                    </a>
                </div>
            </div>
            <div class="row paddingTopBig">
                <div class="col-8 copyright">
                    <ul>
                        <li>© Chili Solutions GmbH 2018</li>
                        <li><a target="_blank" href="https://www.chili.ch/impressum">Impressum</a></li>
                        <li><a target="_blank" href="https://www.chili.ch/datenschutzbestimmung/">Datenschutzbestimmung</a></li>
                    </ul>
                </div>
                <div class="col-4 socialmedia">
                    <a target="_blank" href="https://www.facebook.com/chiliwebagentur/"><img alt="socialicon"
src="~/_img/socialmedia/sm_facebook.svg" /></a>
                    <a target="_blank" href="https://plus.google.com/+ChiliSolutionsGmbHZ%C3%BCrich"><img alt="socialicon"
src="~/_img/socialmedia/sm_googleplus.svg" /></a>
                    <a target="_blank" href="https://www.linkedin.com/company/chili-solutions-gmbh"><img alt="socialicon"
src="~/_img/socialmedia/sm_linkedin.svg" /></a>
                    <a target="_blank" href="https://twitter.com/chili_info"><img alt="socialicon"
src="~/_img/socialmedia/sm_twitter.svg" /></a>
                    <a taget="_blank" href="https://www.youtube.com/user/chiliwebagentur"><img alt="socialicon"
src="~/_img/socialmedia/sm_youtube.svg" /></a>
                </div>
            </div>
        </div>
    </footer>

    <script src="~/_js/lib/jquery.min.js"></script>
    @RenderSection("OptionaleSkripts", false)
</body>
</html>
```

Login.cshtml

```
@*// ================================================================
// Erstelldatum:        06.04.2018
// Ersteller:           Chili/hth
// Zweck:               Loginseite des Reporttools
// ================================================================*@
@using Inhalte
@model Data.TAB_Adresse
@{
    ViewBag.Title = "Login";
    Layout = "~/Views/Shared/Master.cshtml";
}
<div class="container paddingTopMedium minHeight">
    <div class="row">
        <div class="col-12">
            <h2>@ResourceHelper.GetText("loginSeitenTitelKlein")</h2>
            <h1>@ResourceHelper.GetText("loginSeitenTitelGross")</h1>
        </div>
        <div class="col-4 paddingTopMedium">
            <div class="titleBox">@ResourceHelper.GetText("loginBoxTitel")</div>
            <div class="formWrapper">
                @using (Html.BeginForm("Login", "Login", FormMethod.Post))
                {
                    @Html.TextBoxFor(m => m.NUM_Nummer, new { placeholder = "Kundennummer", data_val_number =
ResourceHelper.GetValidierungsmeldung("nummerFehler") })
                    <div class="validationContainer">
                        @Html.ValidationMessageFor(m => m.NUM_Nummer)
                    </div>
                    @Html.PasswordFor(m => m.TXT_Passwort, new { placeholder = "Passwort" })
                    <div class="validationContainer">
                        @Html.ValidationMessageFor(m => m.TXT_Passwort)
                        <span>@ViewBag.Error</span>
                    </div>
                    <input class="loginButton" type="submit" value='@ResourceHelper.GetText("loginButtonText")' />
                }
            </div>
        </div>
    </div>
</div>
```

ReportDetail.cshtml

```
@*  ============================================================
    // Erstelldatum:        09.04.2018
    // Ersteller:           Chili/hth
    // Zweck:               Detailübersicht eines Reports
    // ============================================================*@
@model Data.vw_Marketing
@using ClassLibrary
@using Inhalte

@{
    ViewBag.Title = "Uebersicht";
    ViewBag.BodyClass = "footerweg";
    Layout = "~/Views/Shared/Master.cshtml";
}
@section OptionaleSkripts
{
    <script src="~/_js/lib/chart.min.js"></script>
    <script src="~/_js/diagramm.js"></script>
    <script src="~/_js/reportdetail.js"></script>
}
<div class="text-hide" id="currentId">
    @Model.ID
</div>


<div class="fusszeile">
    @ResourceHelper.GetText("printFusszeileText") - @ReportsHelper.GetMonatDurchZahl(Model.NUM_Monat) @Model.NUM_Jahr -
@Model.TXT_Adress1
</div>
<div class="container noprint">
    <div class="row inhaltsseite">
        <div class="col-12 col-sm-6 ">
            <a class="customButton" href="/Reportuebersicht">@ResourceHelper.GetText("alleReportsText")</a>
        </div>
        <div class="col-12 col-sm-6 text-right">
            <a class="customButton fright" href="/Logout">@ResourceHelper.GetText("logoutText")</a>
        </div>
        <div class="col-12 col-sm-4 ">
```

```
            @if (@Model.ReportVormonat.ID != 0)
            {
            <a class="customButton" href="/ReportDetail/Report/@Model.ReportVormonat.ID">@ResourceHelper.GetText("vormonatText")</a>
            }

        </div>
        <div class="col-12 col-sm-4 text-center ">
            <a class="customButton middle" href="javascript:window.print();">@ResourceHelper.GetText("druckenText")</a>
        </div>
        <div class="col-12 col-sm-4 text-right">
            @if (@Model.ReportFolgemonat.ID != 0)
            {
            <a class="customButton fright"
href="/ReportDetail/Report/@Model.ReportFolgemonat.ID">@ResourceHelper.GetText("folgemonatText")</a>
            }

        </div>
    </div>
</div>
<div class="reportdetail">
    <div class="container">
        <div class="row">
            <div class="col-9">
                <h2>@ResourceHelper.GetText("titelText") @ReportsHelper.GetMonatDurchZahl(Model.NUM_Monat) @Model.NUM_Jahr für
@Model.TXT_URL</h2>
            </div>
            <div class="col-3">
                <img class="bildVolleBreite" src="~/_img/logo_chili.svg" />
            </div>
        </div>
        <div class="row printfix">
            <div class="col-12">
                <h3>Kampagnen</h3>
                <h4>Massnahmen im Vormonat</h4>
                <ul>
                    @{
                        foreach (var massnahme in Model.MassnahmenVormonatListe)
                        {
                            <li>@massnahme.TXT_Massname</li>
                        }
```

```
                }

            </ul>
            <h4 class="paddingTopSmall">Kommentar</h4>
            @Model.TXT_Kommentar
        </div>
    </div>
    <div class="row paddingTopSmall">
        <div class="col-12">
            <h5>Kampagnenperformance</h5>
            <table class="tabelleVolleBreite">
                <tr>
                    <th>@ResourceHelper.GetText("messungText")</th>
                    <td>Impr</td>
                    <td>Klicks</td>
                    <td>CTR</td>
                    <td>Conversions</td>
                    <td>Conversion-Rate</td>
                    <td>Kosten/Conv</td>
                    <td>CPC</td>
                    <td class="active">Kosten(CHF)</td>
                </tr>
                @foreach (var report in Model.VorherigeReports)
                {

                    <tr>
                        <th><a href="/ReportDetail/Report/@report.ID">@ReportsHelper.GetMonatDurchZahl(report.NUM_Monat)</a></th>
                        <td>@report.NUM_Impressions</td>
                        <td>@report.NUM_Clicks</td>
                        <td>@report.NUM_ClickThroughRate%</td>
                        <td>@report.NUM_Conversions</td>
                        <td>@report.NUM_ConversionRate%</td>
                        <td>@report.NUM_CostPerConversion</td>
                        <td>@report.NUM_CostPerClick</td>
                        <td>@report.NUM_CostTotal</td>
                    </tr>
                }

                <tr>
                    <th>@ResourceHelper.GetText("aenderungZumVormonatText")</th>
```

```
                    <td>@Model.AenderungenVormonat.ImpressionenProzent.ToString("0.00")%</td>
                    <td>@Model.AenderungenVormonat.KlicksProzent.ToString("0.00")%</td>
                    <td>@Model.AenderungenVormonat.NUM_ClickThroughRate.ToString("0.00")%</td>
                    <td>@Model.AenderungenVormonat.NUM_Conversions</td>
                    <td>@Model.AenderungenVormonat.NUM_ConversionRate.ToString("0.00")%</td>
                    <td>@Model.AenderungenVormonat.NUM_CostPerConversion.ToString("0.00")%</td>
                    <td>@Model.AenderungenVormonat.NUM_CostPerClick</td>
                    <td>@Model.AenderungenVormonat.NUM_CostTotal.ToString("0.00")%</td>

                </tr>
            </table>
        </div>
    </div>
    <div class="row paddingTopSmall">
        <div class="col-12">
            <h3>Kampagnen</h3>
            <h4>Massnahmen diesen Monat</h4>
            <ul>
                @{
                    foreach (var massnahme in Model.MassnahmenListe)
                    {
                        <li>@massnahme.TXT_Massname</li>
                    }
                }
            </ul>
            <h4 class="paddingTopSmall">Ziele</h4>
            @Model.TXT_Ziele
        </div>
        <div class="col-12 paddingTopSmall">
            <h3>Traffic (ganze Site)</h3>
        </div>
        <div class="col-4">
            Besuche: @Model.Besuche<br />
            @Model.AenderungenVormonat.BesucheProzent.ToString("0.00")% / Vormonat:  @Model.ReportVormonat.Besuche
        </div>
        <div class="col-4">
            Absprungrate: @Model.NUM_Absprungrate%<br />
            @Model.AenderungenVormonat.NUM_Absprungrate.ToString("0.00")% / Vormonat: @Model.ReportVormonat.NUM_Absprungrate%
        </div>
        <div class="col-4">
```

```
                Durchschn. Verweildauer: @Model.NUM_Durchschnittsverweildauer (mins)<br />
                @Model.AenderungenVormonat.NUM_Durchschnittsverweildauer.ToString("0.00")% / Vormonat:
@Model.ReportVormonat.NUM_Durchschnittsverweildauer
            </div>
        </div>
        <div class="row">
            <canvas id="trafficDiagramm"></canvas>
        </div>
        <div class="row paddingTopSmall">
            <div class="col-12 text-center">
                @ResourceHelper.GetText("unterDiagrammText")
            </div>
        </div>
        <div class="row fusszeiledesktop">
            <div class="col-8 noprint">
                @ResourceHelper.GetText("printFusszeileText") - @ReportsHelper.GetMonatDurchZahl(Model.NUM_Monat) @Model.NUM_Jahr -
@Model.TXT_Adress1
            </div>
            <div class="col-4 ">
                <a class="customButton fright noprint" href="javascript:window.print();">@ResourceHelper.GetText("druckenText")</a>
            </div>
        </div>
    </div>
</div>
```

Uebersicht.cshtml

```
@*  ============================================================
    // Erstelldatum:        09.04.2018
    // Ersteller:           Chili/hth
    // Zweck:               Übersichtsseite über alle Reports
    //                      des Kunden
    // ============================================================*@
@using ClassLibrary
@using Inhalte
@model List<Data.vw_Marketing>
@{
    ViewBag.Title = "Uebersicht";
    ViewBag.BodyClass = "footerweg";
    Layout = "~/Views/Shared/Master.cshtml";
}
@section OptionaleSkripts
{
    <script src="~/_js/lib/isotope.min.js"></script>
    <script src="~/_js/filter.js"></script>
}

<div class="container">
    <div class="row inhaltsseite">
        <div class="col-12 col-sm-6 ">
            <h2>Meine Adwords Reports</h2>
        </div>
        <div class="col-12 col-sm-6">
            <a class="fright customButton" href="/Logout">@ResourceHelper.GetText("logoutText")</a>
        </div>
    </div>
</div>
<div class="inhalt">
    <div class="container">
        <div class="row">
            <div class="col-12 col-sm-4 col-md-3">

                <input type="text" placeholder="suchen" class='@ResourceHelper.GetText("suchenText")' />
            </div>
            <div class="col-12 col-sm-8 col-md-9">
```

```
                <div class="sort-by-button-group">
                    <div class="customButton sortBtn aktiv" data-sort-value="datedesc">Datum absteigend</div>
                    <div class="customButton sortBtn" data-sort-value="dateasc">Datum aufsteigend</div>
                    <div class="customButton sortBtn" data-sort-value="besuche">Besuche</div>
                </div>
            </div>
        </div>
        <div class="row reportuebersicht grid">

            @foreach (var report in Model)
            {
                var classname = report.IstAktuellerReport ? "active" : "";

                <div class="col-12 col-sm-6 col-md-4 col-lg-3 grid-item @classname">
                    <div class="text-hide datedesc dateasc">@report.NUM_Jahr@report.NUM_Monat.ToString("00")</div>
                    <div class="animierungsDiv">
                        <a href="/ReportDetail/Report/@report.ID">
                            <div class="titleBox filterTitle">@ReportsHelper.GetMonatDurchZahl(report.NUM_Monat) -
@report.NUM_Jahr</div>
                            <div class="report">
                                <div class="filterValue">
                                    <span>@report.TXT_URL</span><br />
                                    <span class="underline"><span class="bold">Besuche:</span> <span
class="besuche">@report.Besuche</span></span><br />
                                    <span class="bold">Suchmaschinen:</span> @report.NUM_Suchmaschinen<br />
                                    <span class="bold">Verweise:</span> @report.NUM_Verweise<br />
                                    <span class="bold">Sonstige:</span> @report.NUM_Sonstige<br />
                                </div>

                                <div class="customButton">@ResourceHelper.GetText("ansehenText")</div>

                            </div>
                        </a>
                    </div>
                </div>
            }
        </div>
    </div>
</div>
```

DiagrammHelper.cs

```
// ============================================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Hilft die Diagrammdaten zu verwalten
//                      Bereitet Diagrammdaten auf, damit sie dargestellt werden können
// ============================================================================
using System.Linq;
using Data;
using Data.Models;
using Inhalte;

namespace ClassLibrary
{
    public static class DiagrammHelper
    {
        /// <summary>
        /// Bereitet die Daten für das Diagramm so vor, dass Sie die gleiche Struktur haben wie das Chart.js Plugin Clientseitig
        /// </summary>
        /// <param name="reportId">Identifizierende ID eines Reports</param>
        /// <returns>Bestücktes DiagrammDaten Model</returns>
        public static DiagrammDaten GetDiagrammDaten(int reportId)
        {
            using (var dbContext = new marketingEntities())
            {
                // Holt den aktuellen Report
                var aktuellerReport = dbContext.vw_Marketing.FirstOrDefault(e =>
                    e.ID == reportId);

                //Holt die vorherigen Reports (6 ist die Anzahl der Monate)
                var alleReports = ReportsHelper.GetVorherigeReports(aktuellerReport, 6);

                //Zählt die Anzahl der geholten Reports um die Arraylänge zu bestimmen
                int anzahlReports = alleReports.Count;

                //Definierung der Arrays
                var bezeichnungen = new string[anzahlReports];
                var verweise = new int[anzahlReports];
                var suchmaschinen = new int[anzahlReports];
```

```
                var sonstige = new int[anzahlReports];
                var trafficVorjahr = new int[anzahlReports];
                var verweiseHintergrund = new string[anzahlReports];
                var suchmaschinenHintergrund = new string[anzahlReports];
                var sonstigeHintergrund = new string[anzahlReports];
                var trafficVorjahrHintergrund = new string[anzahlReports];

                //Fügt die Werte in die einzelnen Arrays ein
                for (var i = 0; i < anzahlReports; i++)
                {
                    var report = alleReports.ElementAt(i);

                    //Werte setzen
                    bezeichnungen[i] = ReportsHelper.GetMonatDurchZahl((report.NUM_Monat));
                    verweise[i] = report.NUM_Verweise;
                    suchmaschinen[i] = report.NUM_Suchmaschinen;
                    sonstige[i] = report.NUM_Sonstige;
                    trafficVorjahr[i] = report.BesucheVorjahr;
                    //Hintergrundfarben setzen
                    verweiseHintergrund[i] = ResourceHelper.GetFarbe("hintergrundVerweise");
                    suchmaschinenHintergrund[i] = ResourceHelper.GetFarbe("hintergrundSuchmaschinen");
                    sonstigeHintergrund[i] = ResourceHelper.GetFarbe("hintergrundSonstige");
                    trafficVorjahrHintergrund[i] = ResourceHelper.GetFarbe("hintergrundTrafficVorjahr");

                    //Wenn es der letzte Report ist, werden die Hintergrundfarben gesetzt die hervorhebend sind
                    if (i + 1 == alleReports.Count)
                    {
                        verweiseHintergrund[i] = ResourceHelper.GetFarbe("hintergrundVerweiseHervorgehoben");
                        suchmaschinenHintergrund[i] = ResourceHelper.GetFarbe("hintergrundSuchmaschinenHervorgehoben");
                        sonstigeHintergrund[i] = ResourceHelper.GetFarbe("hintergrundSonstigeHervorgehoben");
                    }
                }

                //Konvertiert die Arrays zu einem DiagrammDatenObjekt
                return GetDiagrammDatenObjekt(bezeichnungen, verweise, suchmaschinen, sonstige, trafficVorjahr, verweiseHintergrund,
suchmaschinenHintergrund, sonstigeHintergrund, trafficVorjahrHintergrund);
            }
        }

        /// <summary>
```

```
        /// Konvertiert die verschiedenen Arrays zu einem DiagrammDatenObjekt
        /// </summary>
        /// <param name="labels">String-Array von Labels die im Diagramm unter den Balken dargesellt werden</param>
        /// <param name="verweise">Int-Array der Werte des Verweistraffics aus der Datenbank</param>
        /// <param name="suchmaschinen">Int-Array der Werte des Suchmaschinentraffics aus der Datenbank</param>
        /// <param name="sonstige">Int-Array der Werte des Sonstigen Traffics aus der Datenbank</param>
        /// <param name="trafficVorjahr">Int-Array der Werte des Traffic des Vorjahres</param>
        /// <param name="verweiseHintergrund">String-Array der Farben für den Hintergrund der Verweise</param>
        /// <param name="suchmaschinenHintergrund">String-Array der Farben für den Hintergrund der Suchmaschinen</param>
        /// <param name="sonstigeHintergrund">String-Array der Farben für den Hintergrund des sonstigen Traffics</param>
        /// <param name="trafficVorjahrHintegrrund">String-Array der Farben für den Hintergrund des Traffics des Vorjahrs</param>
        /// <returns>Ein vollwertiges DiagrammDatenObjekt</returns>
        public static DiagrammDaten GetDiagrammDatenObjekt(string[] labels, int[] verweise, int[] suchmaschinen, int[] sonstige, int[]
trafficVorjahr, string[] verweiseHintergrund, string[] suchmaschinenHintergrund, string[] sonstigeHintergrund, string[]
trafficVorjahrHintegrrund)
        {
            var dataSet = new DataSet[4];
            dataSet[0] = new DataSet()
            {
                Label = "Traffic Vorjahr",
                Data = trafficVorjahr,
                BackgroundColor = trafficVorjahrHintegrrund,
                BorderColor = "#FFC513",
                Type = "line",
                Fill = false
            };
            dataSet[1] = new DataSet()
            {
                Label = "Suchmaschinen",
                Data = suchmaschinen,
                BackgroundColor = suchmaschinenHintergrund,
                BorderColor = ResourceHelper.GetFarbe("rahmenSuchmaschinen")
        };
            dataSet[2] = new DataSet()
            {
                Label = "Sonstige",
                Data = sonstige,
                BackgroundColor = sonstigeHintergrund,
                BorderColor = ResourceHelper.GetFarbe("rahmenSonstige")
            };
```

```
            dataSet[3] = new DataSet()
            {
                Label = "Verweise",
                Data = verweise,
                BackgroundColor = verweiseHintergrund,
                BorderColor = ResourceHelper.GetFarbe("rahmenVerweise")
            };

            var chartData = new DiagrammDaten()
            {
                Labels = labels,
                Datasets = dataSet
            };
            return chartData;
        }

    }
}
```

Lognhelper.cs

```
// ============================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Hilft die Loginlogik zu verwalten
// ============================================================
using System.Web;

namespace ClassLibrary
{
    public static class LoginHelper
    {
        /// <summary>
        /// Prüft ob eine Session besteht
        /// </summary>
        /// <returns>Ob der User eingeloggt ist</returns>
        public static bool IstEingeloggt()
        {
            return (HttpContext.Current.Session["KundenNr"] != null);
        }
    }
}
```

HistoryHelper.cs

```
// ============================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Hilft die History zu verwalten
// ============================================================
using Data;

namespace ClassLibrary
{
    public static class HistoryHelper
    {
        /// <summary>
        /// Hängt Daten an die History Tabelle an
        /// </summary>
        /// <param name="history">Historyobjekt befüllt mit Daten für die Historisierung</param>
        public static void Schreiben(TAB_Adresse_History_Login history)
        {
            //Die lokale IP wird abgefangen und mit "localhost" ersetzt.
            //Ist besser verständlich
            if (history.TXT_IP == "::1")
            {
                history.TXT_IP = "localhost";
            }

            using (var dbContext = new marketingEntities())
            {
                //Hängt die Historydaten an die Tabelle an
                dbContext.TAB_Adresse_History_Login.Add(history);
                dbContext.SaveChanges();
            }
        }
    }
}
```

ReportsHelper.cs

```
// ============================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Beinhaltet alle Methoden um Reports
//                      zu verwalten
// ============================================================
using System;
using System.Collections.Generic;
using System.Linq;
using Data;

namespace ClassLibrary
{
    public static class ReportsHelper
    {
        /// <summary>
        /// Datenbankabfrage aller Reports eines Kunden
        /// </summary>
        /// <param name="kundenNr">Identifizierende Nummer eines Kunden</param>
        /// <returns>Gibt alle Reports eines Kunden zurück</returns>
        public static List<vw_Marketing> GetReportsDurchKundenNummer(int kundenNr)
        {
            using (var dbContext = new marketingEntities())
            {
                var reportsListe = (from a in dbContext.vw_Marketing where a.NUM_Kunde == kundenNr orderby a.NUM_Jahr descending,
a.NUM_Monat descending select a).ToList();

                //Der aktuelle Report ist immer der aktuelle Monat - 1
                int aktuellerMonat = DateTime.Now.Month - 1;
                int aktuellesJahr = DateTime.Now.Year;
                foreach (var report in reportsListe)
                {
                    //Prüft ob der Report der Report des aktuellen Datums ist
                    report.Besuche = (report.NUM_Sonstige + report.NUM_Suchmaschinen + report.NUM_Verweise);
                    if (report.NUM_Monat == aktuellerMonat && report.NUM_Jahr == aktuellesJahr)
                    {
                        report.IstAktuellerReport = true;
                    }
```

```
            }
            return reportsListe;
        }
    }
    /// <summary>
    /// Datenbankabfrage eines Reports durch die ReportId
    /// </summary>
    /// <param name="reportId">Identifizierende ID eines Reports</param>
    /// <returns>ein einziger Report</returns>
    public static vw_Marketing GetReportDurchReportId(int reportId, int? kundenNr = 0)
    {
        using (var dbContext = new marketingEntities())
        {
            var report = new vw_Marketing();

            //prüft ob der Report zu diesem Kunde gehört
            if (kundenNr != 0)
            {
                 report = dbContext.vw_Marketing.FirstOrDefault(e =>
                    e.ID == reportId && e.NUM_Kunde == kundenNr);
            }
            else
            {
                report = dbContext.vw_Marketing.FirstOrDefault(e =>
                    e.ID == reportId);
            }

            //Wenn der Report Null ist wird ein leeres vw_Marketing Objekt zurückgegeben
            if (report == null) return new vw_Marketing();

            //Besuche werden berechnet
            report.Besuche = (report.NUM_Verweise + report.NUM_Suchmaschinen + report.NUM_Sonstige);
            return report;

        }
    }

    /// <summary>
    /// Fragt den Vormonat eines Reports ab
    /// </summary>
```

```
/// <param name="reportId">Identifizierende ID eines Report, von dem man den vorherigen Report sucht</param>
/// <returns>Report des Folgemonatss eines Reports</returns>
public static vw_Marketing GetReportVormonatDurchReportId(int reportId)
{
    var aktuellerReport = GetReportDurchReportId(reportId);
    int aktuellesJahr = aktuellerReport.NUM_Jahr;
    int aktuellerMonat = aktuellerReport.NUM_Monat;

    //Falls der Monat = 1 ist, wird er auf 12 gesetzt. (Januar wird zu Dezember)
    if (aktuellerMonat == 1)
    {
        aktuellerMonat = 12;
        aktuellesJahr--;
    }
    else
    {
        aktuellerMonat--;
    }
    return GetReportDurchMonatJahrKundenNr(aktuellesJahr, aktuellerMonat, aktuellerReport.NUM_Kunde);
}

/// <summary>
/// Fragt den Vormonat eines Reports ab
/// </summary>
/// <param name="reportId">Identifizierende ID eines Report, von dem man den nachfolgenden Report sucht</param>
/// <returns>Report des Vormonats eines Reports</returns>
public static vw_Marketing GetReportFolgeMonatDurchReportId(int reportId)
{
    var aktuellerReport = GetReportDurchReportId(reportId);
    int aktuellesJahr = aktuellerReport.NUM_Jahr;
    int aktuellerMonat = aktuellerReport.NUM_Monat;

    //Falls der Monat = 12 ist, wird er auf 1 gesetzt. (Dezember wird zu Januar)
    if (aktuellerMonat == 12)
    {
        aktuellerMonat = 1;
        aktuellesJahr++;
    }
    else
    {
```

```
            aktuellerMonat++;
        }
        return GetReportDurchMonatJahrKundenNr(aktuellesJahr, aktuellerMonat, aktuellerReport.NUM_Kunde);
    }

    /// <summary>
    /// Sucht in der Datenbank nach einem Report mit übergebenem Jahr, Monat und Kundennummer
    /// </summary>
    /// <param name="aktuellesJahr">Jahr des zu suchenden Reports</param>
    /// <param name="aktuellerMonat">Monat des zu suchenden Reports</param>
    /// <param name="aktuellerKunde">Identifizierende ID eines Kunden</param>
    /// <returns></returns>
    public static vw_Marketing GetReportDurchMonatJahrKundenNr(int aktuellesJahr, int aktuellerMonat, int? aktuellerKunde)
    {
        using (var dbContext = new marketingEntities())
        {
            //Holt den aktuellen Report
            var rueckgabeWert = dbContext.vw_Marketing.FirstOrDefault(e =>
                e.NUM_Jahr == aktuellesJahr
                && e.NUM_Monat == aktuellerMonat
                && e.NUM_Kunde == aktuellerKunde);
            if (rueckgabeWert != null)
            {
                //Berechnet die Besuche
                rueckgabeWert.Besuche = (rueckgabeWert.NUM_Verweise + rueckgabeWert.NUM_Suchmaschinen +
                                         rueckgabeWert.NUM_Sonstige);
                return rueckgabeWert;
            }
            return new vw_Marketing();
        }
    }

    /// <summary>
    /// Sucht nach vorhergehenden Reports eines übergebenen Reports
    /// </summary>
    /// <param name="aktuellerReport">Der Report, von dem aus vorhergehende Reports gesucht werden sollen</param>
    /// <param name="anzahlMonateZurueck">Anzahl vorhergehende Reports die zurückgegeben werden sollen</param>
    /// <returns>Eine Liste mit vorhergehenden Reports</returns>
    public static List<vw_Marketing> GetVorherigeReports(vw_Marketing aktuellerReport, int anzahlMonateZurueck)
    {
```

```
        var kundennummer = aktuellerReport.NUM_Kunde;
        int aktuellesJahr = aktuellerReport.NUM_Jahr;
        int aktuellerMonat = aktuellerReport.NUM_Monat;
        var reportListe = new List<vw_Marketing>();
        using (var dbContext = new marketingEntities())
        {
            for (var i = 0; i < anzahlMonateZurueck + 1; i++)
            {
                var report = dbContext.vw_Marketing.FirstOrDefault(e =>
                    e.NUM_Jahr == aktuellesJahr
                    && e.NUM_Monat == aktuellerMonat
                    && e.NUM_Kunde == kundennummer);

                if (report != null)
                {
                    //Berechnet die Besuche
                    report.BesucheVorjahr = GetBesucheVorjahr(report);
                    reportListe.Add(report);
                }

                //Prüft ob der Monat = 1 ist und falls ja wird er auf 12 gesetzt
                if (aktuellerMonat == 1)
                {
                    aktuellerMonat = 12;
                    aktuellesJahr--;
                }
                else
                {
                    aktuellerMonat--;
                }
            }
        }
        return reportListe.AsEnumerable().Reverse().ToList();
    }
    /// <summary>
    /// Konvertiert den Monat als Int zu einem String
    /// 1 zu Januar
    /// </summary>
    /// <param name="monat">Monat als Int (1-12)</param>
    /// <returns>Monat als String</returns>
```

```
        public static string GetMonatDurchZahl(int monat)
        {
            return Enum.GetName(typeof(Inhalte.Monate), monat);
        }
/// <summary>
        /// Vergleicht den übergebenen Report mit dem eigenen Report.ReportVormonat
        /// </summary>
        /// <param name="report">Report der vergleicht werden soll</param>
        /// <returns>Ein Model mit den verglichenen Daten</returns>
        public static vw_Marketing VergleicheVormonatMitReport(vw_Marketing report)
        {
            //Prüft ob es den Vormonat überhaupt gibt
            if (report.ReportVormonat != null)
            {
                var rueckgabeReport = new vw_Marketing()
                {
                    //Prozentuale Änderungen werden berechnet
                    ImpressionenProzent = 100 - (report.ReportVormonat.NUM_Impressions / (double)report.NUM_Impressions * 100),
                    KlicksProzent = 100 - (report.ReportVormonat.NUM_Clicks / (double)report.NUM_Clicks * 100),
                    NUM_ClickThroughRate = 100 - (report.ReportVormonat.NUM_ClickThroughRate / report.NUM_ClickThroughRate * 100),
                    NUM_ConversionRate = 100 - (report.ReportVormonat.NUM_ConversionRate / report.NUM_ConversionRate * 100),
                    NUM_CostPerConversion = 100 - (report.ReportVormonat.NUM_CostPerConversion / report.NUM_CostPerConversion * 100),
                    NUM_CostTotal = 100 - (report.ReportVormonat.NUM_CostTotal / report.NUM_CostTotal * 100),
                    BesucheProzent = 100 - report.ReportVormonat.Besuche / (double)report.Besuche * 100,
                    NUM_Absprungrate = 100 - (report.ReportVormonat.NUM_Absprungrate / report.NUM_Absprungrate * 100),
                    NUM_Durchschnittsverweildauer = 100 - (report.ReportVormonat.NUM_Durchschnittsverweildauer /
report.NUM_Durchschnittsverweildauer * 100),

                    //Absolute Änderungen werden berechnet
                    NUM_CostPerClick = report.NUM_CostPerClick - report.ReportVormonat.NUM_CostPerClick,
                    NUM_Conversions = report.NUM_Conversions - report.ReportVormonat.NUM_Conversions
                };
                return rueckgabeReport;
            }
            //gibt ein leeres vw_Marketing Model zurück
            return new vw_Marketing();
        }
```

```
        /// <summary>
        /// Sucht nach dem Vorjahresreport und gibt die Anzahl Besuche zurück
        /// </summary>
        /// <param name="report">Report von dem der Vorjahresreport gesucht werden soll</param>
        /// <returns>Anzahl Besuche im Vorjahr</returns>
        public static int GetBesucheVorjahr(vw_Marketing report)
        {
            //Holt den Vorjahresreport und gibt die Besuche davon zurück
            var reportVorjahr = GetReportDurchMonatJahrKundenNr(report.NUM_Jahr - 1, report.NUM_Monat, report.NUM_Kunde);
            return reportVorjahr.Besuche;
        }

        /// <summary>
        /// Sucht nach Massnahmen eines Reports
        /// </summary>
        /// <param name="reportId">Identifizierende ID eines Reports</param>
        /// <returns>Alle Massnahmen eines Reports</returns>
        public static List<TAB_Marketing_Kampagne_Massnahme> GetMassnahmenDurchReportId(int reportId)
        {
            using (var dbContext = new marketingEntities())
            {
                return (from massnahme in dbContext.TAB_Marketing_Kampagne_Massnahme
                        where massnahme.NUM_MarketingKampagneId == reportId
                        orderby massnahme.NUM_Position
                        select massnahme).ToList();
            }

        }
    }

}
```

ResourceHelper.cs

```
// ============================================================
// Erstelldatum:        06.04.2018
// Ersteller:           Chili/hth
// Zweck:               Hilft die Resourcefiles zu verwalten
// ============================================================
using Inhalte.Properties;
namespace Inhalte
{
    public class ResourceHelper
    {
        /// <summary>
        /// Sucht im Resourcefile "Texte" nach Daten
        /// </summary>
        /// <param name="key">Schlüsselwort nach dem gesucht werden soll</param>
        /// <returns>Den Inhalt dieses Schlüsselworts</returns>
        public static string GetText(string key)
        {
            return Texte.ResourceManager.GetString(key);
        }
        /// <summary>
        /// Sucht im Resourcefile "Validierungsmeldungen" nach Daten
        /// </summary>
        /// <param name="key">Schlüsselwort nach dem gesucht werden soll</param>
        /// <returns>Den Validierungsmeldung dieses Schlüsselworts</returns>
        public static string GetValidierungsmeldung(string key)
        {
            return Validierungsmeldungen.ResourceManager.GetString(key);
        }
        /// <summary>
        /// Sucht im Resourcefile "Farben" nach Daten
        /// </summary>
        /// <param name="key">Schlüsselwort nach dem gesucht werden soll</param>
        /// <returns>Die Farbe dieses Schlüsselworts</returns>
        public static string GetFarbe(string key)
        {
            return Farben.ResourceManager.GetString(key);
        }
    }}
```

Monate.cs

```
namespace Inhalte
{
    //Enthält alle Monate
    public enum Monate
    {
        Januar = 1,
        Februar = 2,
        März = 3,
        April = 4,
        Mai = 5,
        Juni = 6,
        Juli = 7,
        August = 8,
        September = 9,
        Oktober = 10,
        November = 11,
        Dezember = 12
    }
}
```

DiagrammDaten.cs

```
// ============================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Gibt die Struktur der Diagrammdaten vor
// ============================================================
namespace Data.Models
{
    public class DiagrammDaten
    {
        public string[] Labels { get; set; }
        public DataSet[] Datasets { get; set; }
    }

    public class DataSet
    {
        public string Label { get; set; }
        public int[] Data { get; set; }
        public string[] BackgroundColor { get; set; }
        public string BorderColor { get; set; }
        public string Type { get; set; }
        public bool Fill { get; set; }
    }
}
```

MetaDaten.cs

```
// ============================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Verwaltet die Validierungsmetadaten
// ============================================================
using System.ComponentModel.DataAnnotations;

namespace Reporttool
{
    public class TAB_AdresseMeta
    {
        [Required(ErrorMessageResourceType = typeof(Inhalte.Properties.Validierungsmeldungen),
            ErrorMessageResourceName = "kundennummerPflicht")]
        [Range(0, int.MaxValue, ErrorMessageResourceType = typeof(Inhalte.Properties.Validierungsmeldungen),
            ErrorMessageResourceName = "nummerFehler")]
        public int NUM_Nummer { get; set; }

        [Required(ErrorMessageResourceType = typeof(Inhalte.Properties.Validierungsmeldungen),
            ErrorMessageResourceName = "passwortPflicht")]
        public string TXT_Passwort { get; set; }
    }
}
```

Teilklassen.cs

```
// ============================================================================
// Erstelldatum:        09.04.2018
// Ersteller:           Chili/hth
// Zweck:               Wird benutzt um vorhandene Datenbankmodels zu erweitern
//                      und Metadaten mit Datenbankmodels zu verknüpfen
// ============================================================================
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using Reporttool;

namespace Data
{
    [MetadataType(typeof(TAB_AdresseMeta))]
    public partial class TAB_Adresse
    {

    }

    public partial class vw_Marketing
    {
        public List<vw_Marketing> VorherigeReports { get; set; }
        public vw_Marketing ReportVormonat { get; set; }
        public vw_Marketing ReportFolgemonat { get; set; }
        public vw_Marketing AenderungenVormonat { get; set; }
        public int Besuche { get; set; }
        public int BesucheVorjahr { get; set; }
        public double BesucheProzent { get; set; }
        public double ImpressionenProzent { get; set; }
        public double KlicksProzent { get; set; }
        public List<TAB_Marketing_Kampagne_Massnahme> MassnahmenVormonatListe { get; set; }
        public List<TAB_Marketing_Kampagne_Massnahme> MassnahmenListe { get; set; }
        public bool IstAktuellerReport { get; set; }
    }
}
```

vw_Marketing.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace Data
{
    using System;
    using System.Collections.Generic;

    public partial class vw_Marketing
    {
        public int Hosting_ID { get; set; }
        public string TXT_Passwort { get; set; }
        public Nullable<int> NUM_Kunde { get; set; }
        public int NUM_MarketingKampagneId { get; set; }
        public int NUM_Verweise { get; set; }
        public int NUM_Sonstige { get; set; }
        public double NUM_Absprungrate { get; set; }
        public int NUM_Suchmaschinen { get; set; }
        public double NUM_Durchschnittsverweildauer { get; set; }
        public string TXT_Kommentar_Traffic { get; set; }
        public int NUM_Jahr { get; set; }
        public int NUM_Monat { get; set; }
        public int NUM_Impressions { get; set; }
        public int NUM_Clicks { get; set; }
        public double NUM_ClickThroughRate { get; set; }
        public int NUM_Conversions { get; set; }
        public double NUM_CostPerConversion { get; set; }
        public double NUM_ConversionRate { get; set; }
        public double NUM_CostPerClick { get; set; }
        public string TXT_Ziele { get; set; }
        public double NUM_CostTotal { get; set; }
        public string TXT_Kommentar { get; set; }
```

```
        public int ID { get; set; }
        public string TXT_Betreff { get; set; }
        public string TXT_Adress1 { get; set; }
        public string TXT_URL { get; set; }
    }
}
```

TAB_Adresse_History_Login.cs

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace Data
{
    using System;
    using System.Collections.Generic;

    public partial class TAB_Adresse_History_Login
    {
        public int ID { get; set; }
        public int NUM_AdresseId { get; set; }
        public System.DateTime DAT_DatumZeit { get; set; }
        public string TXT_IP { get; set; }
        public string TXT_UserAgent { get; set; }
        public string TXT_Text { get; set; }

        public virtual TAB_Adresse TAB_Adresse { get; set; }
    }
}
```

TAB_Marketing_Kampagne_Massnahme

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace Data
{
    using System;
    using System.Collections.Generic;

    public partial class TAB_Marketing_Kampagne_Massnahme
    {
        public int ID { get; set; }
        public int NUM_MarketingKampagneId { get; set; }
        public int NUM_Position { get; set; }
        public string TXT_Massname { get; set; }

        public virtual TAB_Marketing_Kampagne TAB_Marketing_Kampagne { get; set; }
    }
}
```

style.css

```
/* ============================================================
// Erstelldatum:        06.04.2018
// Ersteller:           Chili/hth
// Zweck:               Enthält die CSS-Styles der Applikation
============================================================ */

/* CHILI COLORS Start */

.test {
    background-color: #89484c;
    background-color: #f94246;
    background-color: #b0c9d3;
    background-color: #141414;
    background-color: #515151;
}

/* CHILI COLORS End */
/* Globale Styles Start */



body {
    font-family: "Montserrat";
}

a:hover {
    text-decoration: none;
}

body {
    background-size: cover;
    background-image: url('../_img/background.jpg');
    background-attachment: fixed;
}

h1, h2, h3, h4, h5, h6 {
    margin: 0;
    text-transform: uppercase;
    font-weight: 800;
    color: white;
}

h1 {
    font-size: 55px;
    line-height: 50px;
}

.kopfzeile, .fusszeile {
    display: none;
}

h2 {
    font-size: 30px;
    line-height: 30px;
}

.minHeight {
    min-height: 800px;
}

.fright {
    float: right;
```

```
}

.footerweg .zertifizierungen, .footerweg .website {
    display: none;
}
/* Globale Styles End */
/* Abstandsklassen Start */
.paddingTopSmall {
    padding-top: 25px;
}

.paddingTopMedium {
    padding-top: 50px;
}

.paddingTopBig {
    padding-top: 100px;
}

.paddingBottomSmall {
    padding-bottom: 25px;
}

.paddingBottomMedium {
    padding-bottom: 50px;
}

.paddingBottomBig {
    padding-bottom: 100px;
}

/* Abstandsklassen Ende */

/* Formstyling Start */

input[type="text"], input[type="submit"], input[type="email"], input[type="password"]
{
    width: 100%;
    padding: 15px;
    font-size: 12px;
    text-align: center;
    outline: 0;
    outline: none;
    background-color: white;
    border-radius: 3px;
    border: 1px solid #ccc;
}

input.loginButton {
    background-color: #f94246;
    border: 2px solid #f94246;
    color: #fff;
    letter-spacing: 1.2px;
    cursor: pointer;
    font-weight: 700;
    text-transform: uppercase;
    margin: 0;
    transition: all ease-out .5s;
}

    input.loginButton:hover {
        background-color: white;
        color: #f94246;
        transition: all ease-in .3s;
```

```
    }

.formWrapper {
    padding: 25px;
    background-color: white;
    box-shadow: 2px 3px 22px -4px rgba(0,0,0,1);
}

.titleBox {
    font-family: 'Crimson Text', serif;
    background-color: #f94246;
    color: white;
    padding: 10px 25px;
    font-weight: bold;
    font-size: 24px;
    box-shadow: 2px 3px 22px -4px rgba(0,0,0,1);
}

/* Formstyling End */


/* Footerstyling Start */

footer {
    background-color: #141414;
    color: white;
    /*position: absolute;
    bottom: 0;*/
    width: 100%;
    padding-top: 55px;
    /*margin-bottom: -405px;*/
}

    footer h2 {
        color: #fff;
        font-weight: 400;
        letter-spacing: 1.2px;
        margin-bottom: 13px;
        line-height: 24px;
        font-size: 12px;
    }

    footer address {
        line-height: 24px;
        font-size: 12px;
        color: #636363;
    }

    footer a {
        color: #f94246;
    }

        footer a:hover {
            text-decoration: none;
            padding-bottom: 3px;
            color: #f94246;
            border-bottom: 1px solid #f94246;
        }


    footer .copyright {
        font-size: 11px;
    }
```

```
        footer .copyright ul {
            padding: 0;
        }

            footer .copyright ul li {
                float: left;
                list-style: none;
                margin-right: 17px;
            }
/* Footerstyling End*/

/* ButtonStyling Start */

.customButton {
    background-color: #f94246;
    color: #fff;
    letter-spacing: 1.2px;
    cursor: pointer;
    font-weight: 700;
    text-transform: uppercase;
    margin: 0;
    transition: all ease-out .5s;
    width: 100%;
    max-width: 150px;
    padding: 10px;
    font-size: 12px;
    text-align: center
}

    .customButton:hover {
        background-color: white;
        color: #f94246;
        transition: all ease-in .3s;
    }

/* ButtonStyling End */

/* Feldervalidation Start */

.validationContainer {
    font-size: 12px;
    color: #f94246;
    margin-bottom: 20px;
    padding-top: 5px;
}

/* Feldervalidation End */

/* Reportübersichtsseite Start */

.suchen {
    max-width: 350px;
    background-image: url(../_img/search.png);
    background-repeat: no-repeat;
    background-position: left;
    background-size: 60px;
    margin-bottom: 20px;
}

.inhalt {
    padding-top: 40px;
    padding-bottom: 40px;
    background-color: white;
    min-height: 700px;
```

```
}

.inhaltsseite {
    padding-top: 40px;
    padding-bottom: 40px;
}

.reportuebersicht .titleBox {
    font-size: 20px;
    box-shadow: none;
    margin-top: 15px;
}

.reportuebersicht .animierungsDiv {
    transition: transform ease .25s;
}

    .reportuebersicht .animierungsDiv:hover {
        transform: scale(1.15);
    }

.report {
    background-color: #b0c9d3;
    padding: 20px;
    margin-bottom: 15px;
}

.active .report {
    border: 2px solid gray;
}

.filterValue {
    margin-bottom: 20px;
    font-size: 12px;
}

.reportuebersicht .active .titleBox {
    background-color: #FF0000;
    border-bottom: none;
}

.reportuebersicht .active .customButton {
    background-color: #FF0000;
    border-bottom: none;
}

    .reportuebersicht .active .customButton:hover {
        background-color: white;
    }

.reportuebersicht .active .report {
    background-color: #c3c3c3;
    border-top: none;
}

.reportuebersicht a {
    color: white;
}

.grid-item {
    height: 280px;
}

.filterValue {
```

```
    font-size: 14px;
}

.underline {
    text-decoration: underline;
}

.bold {
    font-weight: bold;
}

.sortBtn {
    max-width: 200px;
    display: inline-block;
    margin-top: 0;
    border: 1px solid #f94246;
    opacity: 0.6;
}


    .sortBtn.aktiv {
        opacity: 1;
    }

        .sortBtn.aktiv:hover {
            background-color: white;
        }



/* Reportsübersichtsseite End */


/* ReportDetail Start */


.inhaltsseite .customButton {
    margin-bottom: 10px;
    display: block;
    max-width: 250px;
}

.middle {
    margin: 0 auto;
}

.reportdetail {
    background-color: white;
    padding-top: 50px;
    padding-bottom: 50px;
}

    .reportdetail h2, .reportdetail h3, .reportdetail h4, .reportdetail h5 {
        color: #141414;
    }

    .reportdetail h2 {
        font-size: 22px;
    }

    .reportdetail h3 {
        font-size: 18px;
        font-weight: 600;
    }
```

```
    .reportdetail h4 {
        font-size: 18px;
        font-weight: 400;
        text-decoration: underline;
    }

.bildVolleBreite {
    width: 100%;
}

.tabelleVolleBreite {
    width: 100%;
}

    .tabelleVolleBreite th a {
        color: white;
        text-decoration: underline;
    }



    .tabelleVolleBreite td, .tabelleVolleBreite th {
        border: 1px solid #ddd;
        padding: 8px;
    }

    .tabelleVolleBreite tr:nth-child(even) {
        background-color: #f2f2f2;
    }

    .tabelleVolleBreite tr:hover {
        background-color: #ddd;
    }

    .tabelleVolleBreite th {
        padding-top: 12px;
        padding-bottom: 12px;
        text-align: left;
        background-color: #f94246;
        color: white;
    }




        .tabelleVolleBreite th:nth-last-child(-n+2) {
            background-color: #FF0000;
        }

        .tabelleVolleBreite th:nth-last-child(+2), .tabelleVolleBreite td:nth-last-
child(+2) {
            border-left: 3px solid black;
        }

.fusszeiledesktop {
    margin-top: 20px;
    font-size: 12px;
}
.fusszeile .customButton {
    max-width: 190px;
    border: 1px solid #f94246;
}
/* ReportDetail End
```