IHK Industrie- und Handelskammer Schwarzwald-Baar-Heuberg

Deckblatt zur Dokumentation

Prüflingsnummer*
169 50550

Christopher | Mogler

Vorname | **Nachname**

Weiss GmbH Softwarelösungen

Ausbildungsbetrieb

Schenkenzeller Str. 163, 77761 Schiltach

Ausbildungsort

Entwickler

Zielgruppe (Auftraggeber) für die Präsentation

Persönliche Erklärung

zur Fachaufgabe und zum Report im Rahmen der Abschlussprüfung in den IT-Berufen

Ich versichere durch meine Unterschrift, dass ich die betriebliche Projektarbeit und die dazugehörige Dokumentation selbstständig in der vorgegebenen Zeit erarbeitet habe. Alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, wurden von mir als solche kenntlich gemacht.

Ebenso bestätige ich, dass ich bei der Erstellung der Dokumentation zu meiner Projektarbeit weder teilweise noch vollständig Passagen aus anderen Dokumentationen übernommen habe, die bei der prüfenden oder einer anderen Kammer eingereicht wurden.

Schiltach, 29.05.2020

Ort, Datum | Unterschrift des Prüfungsteilnehmers

Ich habe die obige Erklärung zur Kenntnis genommen und bestätige, dass die betriebliche Projektarbeit einschließlich der Dokumentation in der vorgegebenen Zeit in unserem Betrieb durch den Prüfungsteilnehmer selbstständig ausgeführt wurde.

Ausbilder/-in

IHK Industrie- und Handelskammer Schwarzwald-Baar-Heuberg

Sommerprüfung 2020

Ausbildungsberuf
Fachinformatiker/-in Anwendungsentwicklung

Prüfungsbezirk
SBH IT-FI-AE03 (AP T2V1)

Herr Christopher Mogler
Identnummer: 540083
Prüflingsnummer: 50550
E-Mail: christophermogler@outlook.de, Telefon: +49 172 6261182

Ausbildungsbetrieb: Weiss GmbH Softwarelösungen
Projektbetreuer: Herr Tobias Oehler
E-Mail: oehler@weissedv.de, Telefon: +49 172 9587700

Thema der Projektarbeit
Vereinfachte Verwaltung, Installation und Steuerung von Zusatzprogrammen auf Kundenseite, mithilfe einer intuitiven Oberflächenverwaltung.

1 Thema der Projektarbeit

Vereinfachte Verwaltung, Installation und Steuerung von Zusatzprogrammen auf Kundenseite, mithilfe einer intuitiven Oberflächenverwaltung.

2 Geplanter Bearbeitungszeitraum

Beginn: 29.03.2020
Ende: 29.05.2020

3 Projektbeschreibung

Problem

Die Firma Weiss arbeitet mit verschiedenen Schnittstellen und Protokollen, die mit Diensten oder als Hintergrundanwendung realisiert wurden. Diese Anwendungen müssen installiert, konfiguriert und aktualisiert werden.
Die Verwaltung der Dienste auf Kundenseite kostet wertvolle Ressourcen und Kapazität, die anderweitig verwendet werden könnte. Zielführend gilt es einen Dienst zu entwickeln, der die Installation, Konfiguration und Wartung vereinfacht.

Lösung

Die Umsetzung wird als Service realisiert, die es dem Kunden ermöglicht Zusatzprogramme zu installieren. Der Service bekommt über eine API mitgeteilt, welche Zusatzprogramme für die Installation/Update zur Verfügung stehen. Über eine Oberfläche ist via Interaktion möglich die gewünschte Programme zu installieren oder zu updaten. Nicht jedem Kunden stehen die gleichen Möglichkeiten zu Installation/Update der Zusatzprogramme zur Verfügung. Nur gekaufte Anwendungen sollen dem Kunden in seiner Übersicht erscheinen. Daher soll es über eine Oberfläche seitens Weiss möglich sein, die Zusatzprogramme kundenspezifisch anzupassen.

Damit der Service beim Kunden die Anwendungs- und Konfigurationsdateien installieren und bereitstellen kann, werden über eine REST-API benötigten Informationen heruntergeladen. Um eine Zuordnung zum Kunden herzustellen wird jedem Service ein SHA-512 Token zugewiesen. Dieser Token ist eindeutig und bei jedem Kunden hinterlegt. Dieser Token wird auch für die Authentifizierung für die REST-API verwendet. In der Oberfläche kann der aktuelle Status von Diensten bei jedem Kunden überblickt werden, solange dieser eine Verbindung zur REST-API aufbauen kann. Wenn der Kunden keine Möglichkeit hat, eine Verbindung zur REST-API aufzubauen, so ist es möglich eine offline Installation durchzuführen. Der Service wird als Windows-Dienst realisiert. Die Dienste und Anwendungen werden regelmäßig auf Updates überprüft, in dem die Dateiversionen mit den aktuellen Versionen abgeglichen werden. Jedoch werden aus Performance-Gründen nur einzelne Dateien ausgetauscht, nie eine komplette

Anwendung. Vor jedem Austauschen wird sichergestellt, dass der betroffene Dienst gestoppt wurde. Dies übernimmt der Service. Die Dienste und Anwendungen werden vom Service überblickt und bei Fehlern die Weiss GmbH informiert. Herausforderungen sind die unterschiedlich aufgebauten Zusatzprogramme. Sie unterscheiden sich in der eingesetzten Sprache und Konfiguration, so kann es vorkommen, dass sie manchmal in JSON, XML oder im INI Format vorliegen. Um dieses Problem zu verhindern, sollte jeder Dienst den gleichen Standard aufweisen. Dadurch sind eventuelle Formangleichungen notwendig.

4 Projektumfeld

Das Projekt wird von Christopher Mogler in der Firma Weiss GmbH Softwarelösungen entwickelt. Sie wurde 1975 von Rolf Weiss mit Hauptsitz in Schiltach gegründet. Zu den ersten Anfängen der Firma wurde für mittelständische Unternehmen Auftragsprogrammierungen auf IBM-Systemen durchgeführt. Im Jahre 1985 wurde die Produktpalette auf ERP-Systeme, für mittelständische und größere Handels bzw. Industrieunternehmen, erweitert.

Die Geschäftsleitung wurde 1998 von Martin Lauble übernommen. Ab dem Jahr 2002 wurde der Schwerpunkt der Software auf das Produkt PowerWeiss für Windows gelegt, welches als CRM-System für Handel, Industrie und Versicherungsagenturen genutzt werden kann.

Christopher Mogler arbeitet als Fachinformatiker für Anwendungsentwicklung in der Firma Weiss.

5 Projektphasen mit Zeitplanung

Die gesamte geschätzte Zeit beläuft sich auf 70h (h = Stunden):

Planung und Konzeption ca. 20h
- Selbstständige Fortbildung: ca. 10h
- Absprache mit der internen Entwicklungsabteilung für Schnittstellen und Dienste: ca.5h
- Projektstruktur bilden (Struktogramme und Programmablaufpläne): ca. 5h

Programmierung ca. 32h
- Entwicklung der REST-API ca. 10h
- Entwicklung der Oberfläche ca. 7h
- Entwicklung des Servicemanagers ca. 10h
- Gesamte Testphase ca. 5h•Erstellung der Dokumentation ca. 10h
- Anwenderdokumentation: ca. 2h
- Technische Dokumentation: ca. 8h

Abnahme ca. 8h
- QS: ca. 5h
- interne Anwender: ca. 3h

6 Dokumentation zur Projektarbeit

Es werden drei Dokumentationen bereitgestellt. Sie umfasst die technische Dokumentation, Anwenderdokumentation und Installationsdokumentation. Die technische Dokumentation umfasst die Bereiche:
- Konzept
- Realisierung
- Struktur
- Quellcode
- Test

7 Anlagen

siehe Anlage 1

8 Präsentationsmittel

- PowerPoint
- Beamer
- Laptop

9 Hinweis!

Ich bestätige, dass der Projektantrag dem Ausbildungsbetrieb vorgelegt und vom Ausbildenden genehmigt wurde. Der Projektantrag enthält keine Betriebsgeheimnisse. Soweit diese für die Antragstellung notwendig sind, wurden nach Rücksprache mit dem Ausbildenden die entsprechenden Stellen unkenntlich gemacht.

Mit dem Absenden des Projektantrages bestätige ich weiterhin, dass der Antrag eigenständig von mir angefertigt wurde. Ferner sichere ich zu, dass im Projektantrag personenbezogene Daten (d. h. Daten über die eine Person identifizierbar oder bestimmbar ist) nur verwendet werden, wenn die betroffene Person hierin eingewilligt hat.

Bei meiner ersten Anmeldung im Online-Portal wurde ich darauf hingewiesen, dass meine Arbeit bei Täuschungshandlungen bzw. Ordnungsverstößen mit „null“ Punkten bewertet werden kann. Ich bin weiter darüber aufgeklärt worden, dass dies auch dann gilt, wenn festgestellt wird, dass meine Arbeit im Ganzen oder zu Teilen mit der eines anderen Prüfungsteilnehmers übereinstimmt. Es ist mir bewusst, dass Kontrollen durchgeführt werden.

Anlage 1

1 Prüfungsanmeldung - Christopher Mogler

1.1 Thema

Vereinfachte Verwaltung , Installation und Steuerung von Zusatzprogrammen auf Kundenseite, mithilfe einer intuitiven Oberflächenverwaltung.

1.2 Projektumfeld

Das Projekt wird von Christopher Mogler in der Firma Weiss GmbH Softwarelösungen entwickelt. Sie wurde 1975 von Rolf Weiss mit Hauptsitz in Schiltach gegründet. Zu den ersten Anfängen der Firma wurde für mittelständische Unternehmen Auftragsprogrammierungen auf IBM-Systemen durchgeführt. Im Jahre 1985 wurde die Produktpalette auf ERP-Systeme, für mittelständische und größere Handels bzw. Industrieunternehmen, erweitert.

Die Geschäftsleitung wurde 1998 von Martin Lauble übernommen. Ab dem Jahr 2002 wurde der Schwerpunkt der Software auf das Produkt PowerWeiss für Windows gelegt, welches als CRM-System für Handel, Industrie und Versicherungsagenturen genutzt werden kann.

Christopher Mogler arbeitet als Fachinformatiker für Anwendungsentwicklung in der Firma Weiss.

1.3 Projektbeschreibung

1.3.1 Problem

Die Firma Weiss arbeitet mit verschiedenen Schnittstellen und Protokollen, die mit Diensten oder als Hintergrundanwendung realisiert wurden. Diese Anwendungen müssen installiert, konfiguriert und aktualisiert werden. Die Verwaltung der Dienste auf Kundenseite kostet wertvolle Ressourcen und Kapazität, die anderweitig verwendet werden könnte. Zielführend gilt es einen Dienst zu entwickeln, der die Installation, Konfiguration und Wartung vereinfacht.

1.3.2 Lösung

Die Umsetzung wird als Service realisiert, die es dem Kunden ermöglicht Zusatzprogramme zu installieren. Der Service bekommt über eine API mitgeteilt, welche Zusatzprogramme für die Installation/Update zur Verfügung stehen. Über eine Oberfläche ist via Interaktion möglich die gewünschte Programme zu installieren oder zu updaten. Nicht jedem Kunden stehen die gleichen Möglichkeiten zu Installation/Update der Zusatzprogramme zur Verfügung. Nur gekaufte Anwendungen sollen dem Kunden in seiner Übersicht erscheinen. Daher soll es über eine Oberfläche seitens Weiss möglich sein, die Zusatzprogramme kundenspezifisch anzupassen.

Damit der Service beim Kunden die Anwendungs- und Konfigurationsdateien installieren und bereitstellen kann, werden über eine REST-API benötigten Informationen heruntergeladen. Um eine Zuordnung zum Kunden herzustellen wird jedem Service ein SHA-512 Token zugewiesen. Dieser Token ist eindeutig und bei jedem Kunden hinterlegt. Dieser Token wird auch für die Authentifizierung für die REST-API verwendet. In der Oberfläche kann der aktuelle Status von Diensten bei jedem Kunden überblickt werden, solange dieser eine Verbindung zur REST-API aufbauen kann. Wenn der Kunden keine Möglichkeit hat, eine Verbindung zur REST-API aufzubauen, so ist es möglich eine offline Installation durchzuführen. Der Service wird als Windows-Dienst realisiert. Die Dienste und Anwendungen werden regelmäßig auf Updates überprüft, in dem die Dateiversionen mit den aktuellen Versionen abgeglichen werden. Jedoch werden aus Performance-Gründen nur einzelne Dateien ausgetauscht, nie eine komplette Anwendung. Vor jedem Austauschen wird sichergestellt, dass der betroffene Dienst gestoppt wurde. Dies übernimmt der Service. Die Dienste und Anwendungen werden vom Service überblickt und bei Fehlern die Weiss GmbH informiert. Herausforderungen sind die unterschiedlich aufgebauten Zusatzprogramme. Sie unterscheiden sich in der eingesetzten Sprache und Konfiguration, So kann es vorkommen, dass sie manchmal in JSON, XML oder im INI Format vorliegen. Um dieses Problem zu verhindern, sollte jeder Dienst den gleichen Standard aufweisen. Dadurch sind eventuelle Formatangleichungen notwendig.

1.4 Projektphasen

Die gesamte geschätzte Zeit beläuft sich auf 70h (h = Stunden):

- Planung und Konzeption ca. 20h
  - Selbstständige Fortbildung: ca. 10h
  - Absprache mit der internen Entwicklungsabteilung für Schnittstellen und Dienste: ca.5h
  - Projektstruktur bilden (Struktogramme und Programmablaufpläne): ca. 5h

- Programmierung ca. 32h
  - Entwicklung der REST-API ca. 10h
  - Entwicklung der Oberfläche ca. 7h
  - Entwicklung des Service-Manager ca. 10h
  - Gesamte Testphase ca. 5h
- Erstellung der Dokumentation ca. 10h
  - Anwenderdokumentation: ca. 2h
  - Technische Dokumentation: ca. 8h
- Abnahme ca. 8h
  - QS: ca. 5h
  - interne Anwender: ca. 3h

1.5 Dokumentation

Es werden drei Dokumentationen bereitgestellt. Sie umfasst die technische Dokumentation, Anwenderdokumentation und Installationsdokumentation.

Die technische Dokumentation umfasst die Bereiche:

- Konzept
- Realisierung
- Struktur
- Quellcode
- Test

Abschlussprüfung Sommer 2020

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Entwicklung eines Service-Manager

Vereinfachte Dienstverwaltung, mithilfe einer intuitiven Oberflächenverwaltung.

Abgabetermin: Schiltach, den 29.05.2020

Prüfungsbewerber:
Christopher Rudolf Karl-Heinz Mogler
Schramberger Str. 13
77761 Schiltach

Ausbildungsbetrieb:
Weiss GmbH Softwarelösungen
Schenkenzeller Str. 163
77761 Schiltach

Inhaltsverzeichnis

Abkürzungsverzeichnis

IDE	Integrated Development Environment
IIS	Microsoft Internet Information Services
IT	Informationstechnik
UI	Userinterface
API	Application Programming Interface
MVC	Model View Controller
DBMS	Datenbankmanagementsystem
HTTP	Hypertext Transfer Protocol
SHA	Secure Hash Algorithm
VPN	Virtual Private Network
DLL	Dynamic Link Library
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
C#	C-Sharp
JS	JavaScript
INI	Initialisierungsdatei
JSX	JavaScript Extensible Markup Language
ASP.NET	Active Server Pages .NET
REST	Representational State Transfer
ODBC	Open Database Connectivity
XML	Extensible Markup Language
SQL	Structured Query Language
JSON	JavaScript Object Notation
ERP	Enterprise Resource Planning
CRM	Customer Relationship Management
IBM	International Business Machines Corporation

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

1.1 Projektumfeld

Das Projekt wurde von Christopher Mogler in der Firma Weiss GmbH Softwarelösungen entwickelt. Die Firma wurde 1975 von Rolf Weiss mit Hauptsitz in Schiltach gegründet. Am Anfang hat die Firma für mittelständische Unternehmen Auftragsprogrammierungen für IBM-Systemen durchgeführt. Im Jahre 1985 wurde die Produktpalette mit ERP-Systemen für mittelständische und größere Handels bzw. Industrieunternehmen erweitert.
Die Geschäftsleitung wurde 1998 von Martin Lauble übernommen. Ab dem Jahr 2002 wurde der Schwerpunkt der Software auf das Produkt PowerWeiss für Windows gelegt, welches als CRM-System für Handel, Industrie und Versicherungsagenturen genutzt werden kann.

1.2 Projektziel

Um Dienste verwalten zu können benötigt der Weiss Mitarbeiter Zugriff auf das aktuelle Kundensystem. Um den Zugriff gewährleisten zu können, müssen bestimmte Anwendung installiert sein. Ferner sind manche Systeme nur über einen Client-Rechner eines Kunden direkt erreichbar.
Wenn der Fernzugriff gewährleistet ist, müssen die aktuellen Dateien des Dienstes auf das Kundensystem kopiert werden. Aktuell ist das Problem, dass Dienst Dateien nicht standardisiert abgelegt werden. Es müssen Dienst Dateien so oft mühsam gesucht werden.
Dienste auf Windows-System verwalten zu können wird das Tool InstallUtil verwendet, was von Microsoft bereitgestellt wird. Der Nachteil dieses Tools ist, dass es nur über eine Eingabeaufforderung verwendet werden kann, was die Fehleranfälligkeit erhöht.
Nach einer Installation müssen Dienste konfiguriert werden. Wenn der Dienst nicht konfiguriert wird, kommt es zu einem Fehlerhaften verhalten.
Die Konfigurationen werden in verschiedenen Auszeichnungssprache geschrieben (z. B. JSON, XML, INI). Da jede Sprache eine eigene Syntax aufweist, kann es schnell zu Syntaxfehler führen.

1.3 Projektschnittstellen

Für die Datenspeicherung wird eine SQL-Anywhere-Datenbank verwendet. Der Zugriff auf diese Datenbank, wird über ODBC realisiert.
Oberfläche und API werden als Web-Applikation bereitgestellt, diese wird über ASP.NET verwaltet. ASP.NET auf Windows-Systeme ausführen zu können wird der IIS-Dienst von Windows verwendet.

Um auf den Dienst-Pool von Windows zugreifen zu können, wird auf die Windows-API zugegriffen. Dies wird durch die AdvAPI32-Bibliothek ermöglicht. Diese Bibliothek wird von Windows bereitgestellt und muss daher nicht gesondert nachinstalliert werden.

Da durch das die Oberfläche auf ReactJS aufbaut ist, wird Serverseitig eine NodeJS Anwendung ausgeführt.
Das Projekt wurde durch Herrn Richter genehmigt, der Abteilungsleiter der Entwicklung – bei der Firma Weiss. Die Endbenutzer sind die Mitarbeiter der Firma Weiss, die geschult wurden durch den Autor. Für bestimmte Fragen wird eine Anwenderdokumentation bereitgestellt, die eine Übersicht aller Prozesse bietet.

1.4 Projektabgrenzung

Oberfläche und API wird auf vorhandenen Systemen installiert und ausgeführt. Es wird keine neue Hardware benötigt.
Der IT-Betrieb der Weiss GmbH übernimmt die Installation auf den Kundensysteme.

2 Projektplanung

2.1 Projektphasen

Das Projekt wurde vom 20. April bis 05. Mai realisiert. Der Arbeitstag wurde in diesem Zeitraum in sieben Stunden für das Projekt und eine Stunde für andere Themen aufgeteilt. Andere Themen sind z. B. Fehlerbehebung, Support und Telefondienst.

Tabelle 1 zeigt die grobe Zeitplanung.

Projektphase	Geplante Zeit
Planung und Konzeption	20 h
Programmierung	32 h
Abnahme	4,5 h
Dokumentation	13,5 h
Gesamt	**70 h**

Tabelle 1: Zeitplanung

Eine detaillierte Zeitplanung findet sich im Anhang A.1: Zeitplanung auf Seite i.

2.2 Abweichungen vom Projektantrag

Im Projektantrag wurde eine Offline Installation aufgeführt. Diese wurde aber durch die Leitung als nicht mehr relevant angesehen. Daher ist diese nicht etabliert worden.

Bei einer Versionsänderung werden die Dateien komplett mit den neuen überschrieben. Das hat den

Vorteil: weniger Quellcode für die Versionsprüfung und verringerte Fehler, bei einer falschen Validierung. Die Performance wird nicht beeinträchtigt, weil die Dienste nicht groß sind und selten viele einzelne Dateien vorweisen.

2.3 Ressourcenplanung

Entwickelt wurde in Schiltach im Büro des Entwicklerteam 1. Von der Firma Weiss wurde die benötigten Ressourcen Hard- und Software bereitgestellt.

Um Kosten gering wie möglich zu halten, wurde auf Software zurückgegriffen die frei zur Verfügung steht oder bereits Lizenzen im Unternehmen vorhanden waren. Hier ist eine Liste der verwendeten Anwendungen:

- Visual Studio 2019 (IDE)
- Windows 10 (Betriebssystem)
- SQL Anywhere (DBMS)
- Interactive SQL (SQL Editor mit integrierter Datenbank Schnittstelle)
- SQL Central (Administrations Tool für SQL Anywhere)
- .NET Framework / Core (Runtime für C#)
- ReactJS (Frontend-JS-Framework)
- Bootstrap (Frontend-CSS-Framework)
- NodeJS (Runtime für JS)

Als organisatorische Hilfe wurde Herr Oehler zur Verfügung gestellt, für den Autor.

2.4 Entwicklungsprozess

Da die einzelnen Programme nicht sehr aufwendig waren, wurde das erweiterte Wasserfallmodell eingesetzt.

Die Implementierungsphase wurde in iterativen Zyklen durchgeführt. Ferner ist zu beachten, dass die einzelnen Teile komplett entwickelt wurden. Jedes Teilprojekt ist nach der Fertigstellung getestet worden, um neue oder vorhandene Funktionen auf Fehler zu prüfen.

3 Analysephase

3.1 Ist-Analyse

Wenn ein Kunden-System angepasst werden muss, benötigt der Mitarbeiter der Firma Weiss Zugriff auf dieses System. Er muss einen Remotezugriff aufbauen, oft wird die Anwendung TeamViewer dafür verwendet.
Es kann vorkommen, dass die Fernwartung nicht durchgeführt werden kann. Die Ursache können sein: Das die entsprechende Software nicht ausgeführt wird, auch kann es an aktuelle Probleme eines Internetproviders liegen, bestimmte Kundenserver können nur über einen Client-Rechner vom Kunden erreicht werden, selten müssen sich Mitarbeiter den Zugriff über den Kunden freischalten lassen.
Es werden Systeme Eingriffe durchgeführt, dadurch werden administrative Rechte benötigt.

Bei einer Installation eines Dienstes wird der Zugriff auf den Kundenserver benötigt.
Hat ein Mitarbeiter Zugriff auf den Kundenserver, so müssen die Dateien für die Dienste auf den Server kopiert werden. Sind die Dateien auf das System kopiert, so kann der Mitarbeiter die Dateien aufbereiten für die Installation. Der Mitarbeiter kann frei entscheiden, wo die Dienste installiert werden. Um einen Dienst installieren zu können wird das InstallUtil-Tool, was von Windows bereitgestellt wird, verwendet. Dieses Tool kann nur über eine Eingabeaufforderung angesteuert werden. Ferner ist zu beachten, dass eine fehlerhafte Installation, aus der Log, nicht direkt ersichtlich ist. Sollte die Installation erfolgreich sein, muss die Konfigurationsdatei angepasst werden. Im Anschluss muss der Mitarbeiter den Dienst zusätzlich starten. Der Dienst sollte nach dem Starten geprüft werden, ob Fehler auftauchen: Status des Dienstes in der Diensteverwaltung von Windows, Ereignisanzeige oder Log-Datei. Nachdem der Dienst installiert und geprüft wurde, kann der Fernzugriff beendet werden.

Muss ein Dienst angepasst werden, so führt der Mitarbeiter eine Fernwartung durch, um Zugriff auf das Kundensystem zu bekommen. Bevor ein Dienst angepasst werden kann, muss dieser zuvor gestoppt werden. Das bei der Installation der Pfad frei gewählt werden kann, kann es vorkommen, dass die Installationspfade der Dienste nicht direkt gefunden werden.
Sollte nur die Konfiguration angepasst werden, wird diese über den vorhandenen Text-Editor beim Kunden bearbeitet und gespeichert.
Bei einer Versionsänderung müssen die passenden Version Dateien gesucht und auf das System hochgeladen werden. Die Dateien haben keinen einheitlichen Ablageort. Um die Versionsänderung durchzuführen, werden die Dateien mit den neuen überschrieben. Es kann vorkommen, dass die Konfigurationsdatei mit überschrieben wird.
Nach der Anpassung kann der Dienst erneut gestartet werden. Wie bei der Installation sollte nach dem Starten der Dienst noch zusätzlich geprüft werden.

Sollte ein Dienst nicht mehr im Gebrauch sein, so kann dieser aus dem Kundensystem entfernt werden. Dazu benötigt der Mitarbeiter Zugriff auf das System, wo der Dienst installiert ist. Um Dienste aus dem Dienst-Pool von Windows entfernen zu können, wird das gleiche Tool, wie bei der Installa-

tion verwendet. Auch hier ist zu beachten, dass die Log-Ausgabe nicht direkt darauf hinweist, dass ein Fehler entstanden ist. InstallUtil übernimmt das Stoppen des Dienstes, bei der Deinstallation. Die Dienst-Dateien müssen nicht entfernt werden. Sollen die Dateien entfernt werden, so muss der Mitarbeiter den Installations-Pfad des Dienstes herausfinden und die Dateien löschen.

3.2 Wirtschaftlichkeitsanalyse

Sollte ein Mitarbeiter keinen direkten Zugriff auf einen Server bekommen, so wird hier viel Zeit benötigt für die Kommunikation und Organisation mit dem Kunden. Wenn der Verwaltungsdienst bereits beim Kunden installiert ist, wird kein Fernzugriff mehr benötigt. Der Dienst kann komplett autonom die Dienste verwaltet.

Version Dateien für Dienste müssen gesucht werden, da es keinen standardisierten Ablageort gibt. Durch die Oberfläche sind alle Dienste tabellarisch aufgelistet. Die Entwickler laden über die Oberfläche aktuelle Versionen mit dazugehöriger Konfiguration hoch. Hier ergibt sich ein einheitlicher Ablageort.

Beim Installieren und Deinstallieren von Diensten, wird InstallUtil benötigt. Dieses Tool kann nur über eine Eingabeaufforderung verwendet werden. Es ist Fehleranfälliger und Fehler werden in der Log nicht direkt ersichtlich. Durch die Oberfläche können Dienste per Knopfdruck installiert oder wieder entfernt werden.
Es kann Zeit gespart und Fehler minimiert werden. Man kann Abläufe besser kontrolliert und steuern.

3.2.1 „Make or Buy“-Entscheidung

Es gibt Anwendungen die Dienste verwalten und Statusmeldung zurückliefern können. Aus wirtschaftlicher Sicht ist trotzdem die “Make”-Methode gewählt worden. Die Anwendungen haben zu viele andere Funktionen, die nicht verwendet werden. Was es dadurch nicht wirtschaftlich macht, wenn für die meisten Funktionen bezahlt wird, aber diese nicht verwendet werden.

3.2.2 Projektkosten

Die Kosten für die Durchführung des Projekts setzen sich sowohl aus Personal-, als auch aus Ressourcenkosten zusammen. Der Autor verdient als Auszubildender 950 € im Monat.

$$8\ \text{h/Tag} \cdot 220\ \text{Tage/Jahr} = 1.760\ \text{h/Jahr} \tag{1}$$

$$950\,€/\text{Monat} \cdot 12\ \text{Monate/Jahr} = 11.400\,€/\text{Jahr} \tag{2}$$

$$\frac{11.400\,€/\text{Jahr}}{1.760\ \text{h/Jahr}} \approx 6{,}477\,€/\text{h} \tag{3}$$

Es ergibt sich also ein Stundenlohn von 6,477 €. Die Durchführungszeit des Projekts beträgt 70 Stunden. Für die Nutzung von Ressourcen[1] wird ein pauschaler Stundensatz von 15 € angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundenlohn von 30 € angenommen. Eine Aufstellung der Kosten befindet sich in Tabelle 2 und sie betragen insgesamt 2.663,39 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	70 h	6,477 € + 15 € = 21,477 €	1.503,39 €
Fachgespräch	3 h	30 € + 15 € = 45 €	135 €
Abnahmetest	1 h	30 € + 15 € = 45 €	45 €
Anwenderschulung	25 h	30 € + 15 € = 45 €	1.125 €
			2.663,39 €

Tabelle 2: Kostenaufstellung

3.2.3 Amortisationsdauer

Bei einer Zeiteinsparung von 30 Minuten am Tag für jeden der 10 Anwender und 220 Arbeitstagen im Jahr ergibt sich eine gesamte Zeiteinsparung von

$$10 \cdot 220\ \text{Tage/Jahr} \cdot 30\ \text{min/Tag} = 66.000\ \text{min/Jahr} \approx 1.100\ \text{h/Jahr} \tag{4}$$

Dadurch ergibt sich eine jährliche Einsparung von

$$1.100\text{h} \cdot (30 + 15)\,€/\text{h} = 49.500\,€ \tag{5}$$

Die Amortisationszeit beträgt also $\frac{2.663{,}39\,€}{49.500\,€/\text{Jahr}} \approx 0,054$ Jahre $\approx 2,4$ Wochen.

Das Projekt hat sich also innerhalb von zwei bis drei Wochen amortisiert.

[1] Räumlichkeiten, Arbeitsplatzrechner etc.

3.3 Anwendungsfälle

3.3.1 Installation

Wenn beim Kunden ein Dienst installiert werden soll, wird über eine Oberfläche der Dienst mit der Version ausgewählt. Die Oberfläche zwingt die Anpassung der Konfiguration. Beim Speichern werden die Daten in die Datenbank gespeichert. Der Dienst beim Kunden selektiert die Daten und installiert automatisch diesen Dienst mit der gelieferten Konfigurationsdatei.

3.3.2 Versionsänderung

Sollte beim Kunden eine neue oder andere Version installiert sein, wird dies wieder über die Oberfläche gesteuert. Die Konfigurationsdatei wird mit der neuen Konfiguration ausgetauscht. Alle Änderungen werden wieder in die Datenbank der Firma Weiss gespeichert. Über die Web-Schnittstelle bekommt der Verwaltungsdienst die Änderungen mitgeteilt und aktualisiert den Dienst.
Beim ändern der Dateien von einem Dienst muss der Dienst gestoppt werden, dies wird vom Verwaltungsdienst übernommen. Und bei erfolgreicher Installation wieder gestartet.

3.3.3 Anpassung

Über die Oberfläche wird die Konfiguration angepasst, wenn diese Fehlerhafte oder veraltete Daten enthält. Beim speichern werden die änderungen in die Datenbank der Firma Weiss gespeichert. Der Dienst bekommt die Änderung mit und überschreibt die aktuelle Konfigurationsdatei.
Der Dienst wird davor gestoppt und nach dem Austausch wieder gestartet.

3.3.4 Deinstallation

Bei einer Deinstallation wird der Dienst über eine Oberfläche entfernt. Der Status in der Datenbank wird zu REMOVE geändert. Beim Kunden wird der Dienst gestoppt und aus dem Dienst-Pool von Windows entfernt.

3.3.5 Statusmeldung

Status und Zustände der aktuellen Dienste werden regelmäßig überliefert und in die Datenbank gespeichert. Über die Oberfläche kann in der Kundenansicht die Stati der Dienste gesehen werden.

4 Entwurfsphase

4.1 Zielplattform

Für DBMS wurde SQL-Anywhere verwendet. Durch die Standardisierung der Firma Weiss, ist es bereit bei jedem Kunden installiert.
Der Verwaltungsdienst und die Web-Anwendung wurden in C# programmiert. Einer der standardisierten Programmiersprachen in der Firma Weiss ist C#. Daher besitzen alle System von Kunden und der Firma Weiss die benötigte Software um die Applikationen ausführen zu können.
Die Web-Anwendungen verwenden als Background Framework ASP.NET, dadurch wird können HTTP Anforderungen einfacher bearbeitet und zurückgegeben werden.

Die Oberfläche verwendet als Frontend Framework ReactJS. Da ReactJS auf MVC-Paradigma aufbaut, wird viel rekursiver Code gespart und hat zu einem großen Zeitersparnis geführt.
Ferner ist zu beachten, dass die Oberfläche als Webanwendung realisiert wurde. Dadurch sinkt die Wartungsarbeit für einzelne Rechner und zudem steigt die Benutzerfreundlichkeit, keine Zusatzsoftware installiert werden muss. Updates sind für den Endbenutzer nicht relevant, da die Versionsänderung Serverseitig stattfindet. Man ist als Web-Anwendung nicht System gebunden, was dazu führt das man mit jedem Browser fähigen Endgerät die Dienste verwalten und steuern kann. Das Gerät sollte aber einen aktuellen Browser verwenden.
Um bestimmte Funktionen von ReactJS steuern zu können, muss Serverseitig eine NodeJS Anwendung laufen.

Die API wurde auf den REST-Standard aufbaut. Aus diesem Grund wurde die Anwendung als Webapplikation realisiert. Es gibt unterschiedliche REST-Methoden, diese können problemlos mit HTTP-Methoden überlagert werden.

Der Verwaltungsdienst wurde als Windows-Dienst realisiert. Die Anwendung läuft im Hintergrund und wird daher als Dienst programmiert. Um Dienste verwalten und steuern zu können, wird auf die Windows-API zugegriffen. Als Bibliothek wird die AdvAPI32 verwendet. Diese wird von Windows bereitgestellt.
Alle Kunden-Systeme laufen auf Windows-Betriebssystemen, daher muss nicht zwingend Systemübergreifend programmiert werden.

4.2 Architekturdesign

Die Webanwendung wurde mit ASP.NET realisiert. ASP.NET ist ein Framework für C#, um die Kommunikation über HTTP zu vereinfachen. Es ist Open-Source und kann frei genutzt werden, zudem wird aktiv daran weiterentwickelt und ist sehr Performant.

Für die Oberfläche wurde ReactJS und Bootstrap verwendet.
ReactJS ist ein Framework was oft für Single-Page-Applikationen verwendet wird. Das bedeutet, dass eine komplette Anwendung nur über eine HTML-Seite funktioniert. Um das Realisieren zu können wird die Seite in Komponenten aufgeteilt. Durch JSX wird es ermöglicht in JavaScript HTML-Code zu schreiben. Komponenten können daher aus JavaScript und HTML bestehen. Durch Komponenten wird viel rekursiver Code gespart, das hat den Grund, dass Komponenten mehrfach eingesetzt werden können.

Der Dienst wird als Windows-Dienst realisiert. Es hat den Grund um die Anwendung von Windows-Dienst-Pool verwalten zu lassen. Das hat den Vorteil, dass die Anwendung komplett im Hintergrund laufen kann.

Das MVC-Paradigma unterteilt die Anwendung in drei Bereiche. Model, repräsentiert die Daten; View, zeigt die Daten an; Controller, verbindet Model mit der View. Die Zielsetzung ist es den Quellcode zu vereinfachen und die Wiederverwendbarkeit zu steigern.

4.3 Entwurf der Benutzeroberfläche

Im Anhang sind die erstellten Konzepte Designs für die Oberfläche. Diese wurden mit dem Mockup-Programm Pencil erstellt. Die Designs konzentrierten sich auf Benutzerfreundlichkeit, weil die Oberfläche nur für interne zugänglich ist. Ferner wurde auf Corporate Design verzichtet.
Aus Zeitgründen wurde auf die Optimierung der Oberfläche für Mobile Endgeräte verzichtet.

4.4 Datenmodell

Die endgültigen Tabellen und Datenstrukturen wurde in einem Tabellenmodell konzeptioniert. Zeitgleich wurden die Relationen, mit deren Kardinalitäten, gezeichnet.
Im Anhang A.5: Tabellenmodell auf Seite viii finden sich die Konzeptionierten Modelle, mit deren Entitäten.

4.5 Geschäftslogik

Die Aktivitätsdiagramme wurden mit ARIS Express erstellt. Das Aktivitätsdiagramm wurde im Anhang A.2.1: Verwaltung von Diensten auf Seite ii beigefügt. Das Diagramm zeigt den Ablauf für die Oberfläche und Dienstprüfung.

5 Implementierungsphase

5.1 Implementierung der Datenstrukturen

Die Tabellen wurden mit hilfe von SQL Central erstellt. Der Autor musste die aufwendigen SQL-Scripts nicht selber schreiben.
Die SQL-Skripte können über SQL Central generiert werden. Zum Erstellen der Tabellen auf der Datenbank, müssen nur die generierten Skripte auf der Datenbank ausgeführt werden.

5.2 Implementierung der Benutzeroberfläche

Die Oberfläche wurde als Webapplikation und auf Basis einer Single-Page-Anwendung realisiert. Als Frontend Framework wurde ReactJS verwendet.
ReactJS Seiten werden in Komponenten aufgeteilt, um Code und vor allem rekursiven Code zu sparen. Komponenten bestehen aus JSX-Dateien. JSX ist eine von React entwickeltes Format. Es ermöglicht in JavaScript XML/HTML Code zu verwenden. Komponenten können auch mehrfach verwendet werden.
Als Frontend-CSS-Framework wurde Bootstrap verwendet.
Teile der Seite sind im Anhang A.3.2: Echtsystem auf Seite vi, als Screenshots beigefügt.

5.3 Implementierung der Geschäftslogik

Um die Dienste installieren zu können wurde die Windows-API angesprochen. Dies wurde mit der AdvAPI32 Bibliothek durchgeführt. Bevor diese verwendet werden kann, muss die bestimmten Funktion der Bibliothek importiert werden. Siehe Anhang A.6: Quellcode auf Seite ix erstes Abbild.
Um auf den Dienst-Pool zugreifen zu können, wird der ServiceControll-Manager von Windows geöffnet. Dadurch bekommt der Dienst die Berechtigung Dienste zu Verwalten.
Wenn ein Dienst deinstalliert werden soll, wird der Dienst gestoppt (wird von der Bibliothek übernommen) und vom Dienst-Pool entfernt. Sollte ein Dienst hinzugefügt werden, wird der Dienstname, der Pfad der Dienst-Anwendung und über welchen Benutzer der Dienst gestartet werden soll angegeben. Nach dem Hinzufügen wird die Rückmeldung der Funktion geprüft, ob Fehler entstanden sind.
Wenn nicht, wird der Dienst gestartet und beim erfolgreichen Start wird die Installation als abgeschlossen angesehen.

6 Einführungsphase

6.1 Anwendungen

6.1.1 Oberfläche / UI

Die Oberfläche wurde auf dem Cloud-Server der Firma Weiss hochgeladen. Auf diesen Cloud-Server läuft ein Windows-Betriebssystem mit der IIS-Applikation.
Die Webanwendungen kann nur aus dem lokalen Netz abgerufen werden, um Unautorisierte Zugriffe zu verhindern. Um Außerhalb der Firma zugreifen zu können, wird ein VPN-Tunnel benötigt.

6.1.2 API

Die API läuft wie die Oberfläche auf dem Cloud-Server der Firma Weiss. Die API ist aus dem öffentlichen Netz erreichbar, da die Daten nur über einen SHA-512 Token freigegeben werden.

6.1.3 Verwaltungsdienst

Um die Mitarbeiter zu Schulen, wurde auf einem Testgerät der Firma Weiss dieser Verwaltungsdienst installiert. Dieser ist mit einem Testkunde verknüpft.

In der Schulung wurden die Dienste auf ausgewählten Kunden-Systeme installiert und ausgeführt.

Dadurch das der Dienst sich selbst installieren kann, muss dieser nur ausgeführt werden. Davor muss die Konfiguration angepasst werden: Authentifizierung Token, Datenbank ODBC Daten, Filestore wo die Dienste installiert werden soll.

6.2 Schulung

Die Mitarbeiter wurden in einer 2,5 stündigen Schulungsveranstaltung geschult. Es wurde die Oberfläche präsentiert und erklärt. Danach ist auf die Installation des Verwaltungsdienst eingegangen worden. Die Mitarbeiter mussten selbst auf Kundensysteme den Verwaltungsdienst installieren und konfigurieren.
In den letzten 10min ist man auf Fragen und Anregungen eingegangen.

7 Dokumentation

8 Fazit

8.1 Soll-/Ist-Vergleich

Das Projektziel wurde erfüllt. Es ist möglich Dienste direkt über die Webapplikation zu steuern. Der Status der Dienste wird rückgemeldet, ob diese Laufen, gestoppt sind, etc.

Projektphase	geplante Zeit	tatsächliche Zeit	Differenz
Planung und Konzeption	20h	23h	+3h
Programmierung	32h	37h	+5h
Abnahme	8h	10h	+2h
Puffer	10h	0h	-10h
	70	**70**	

Tabelle 3: Zeitplanung Soll-/Ist-Vergleich

8.2 Lessons Learned

Der Autor konnte lernen wie eine richtige REST-Anwendung aufgebaut wird. Mit den HTTP-Statusen und Authenfizierungstokens. Auch wie man über C# auf die Windows-API zugreifen kann und darüber Windows-Dienste installieren kann. Wie man DLLs in C# einbindet und die jeweiligen Funktionen importiert.
Wie man von Grund auf eine Webapplikation entwickelt und diese den Endbenutzern erklärt und schult.
Aus einer Idee ein komplett strukturiertes Projekt mit Dokumentation, Zeitmanagement und Ressourcenplanung.

8.3 Ausblick

Der Autor wird nach der Ausbildung die Firma Weiss verlassen, das Produkt wird an das EntwicklerTeam-1 übergeben. Geplante Erweiterungen sind Fernsteuerung der Dienste: z. B. Stoppen, Starten, Neu Starten.

A Anhang

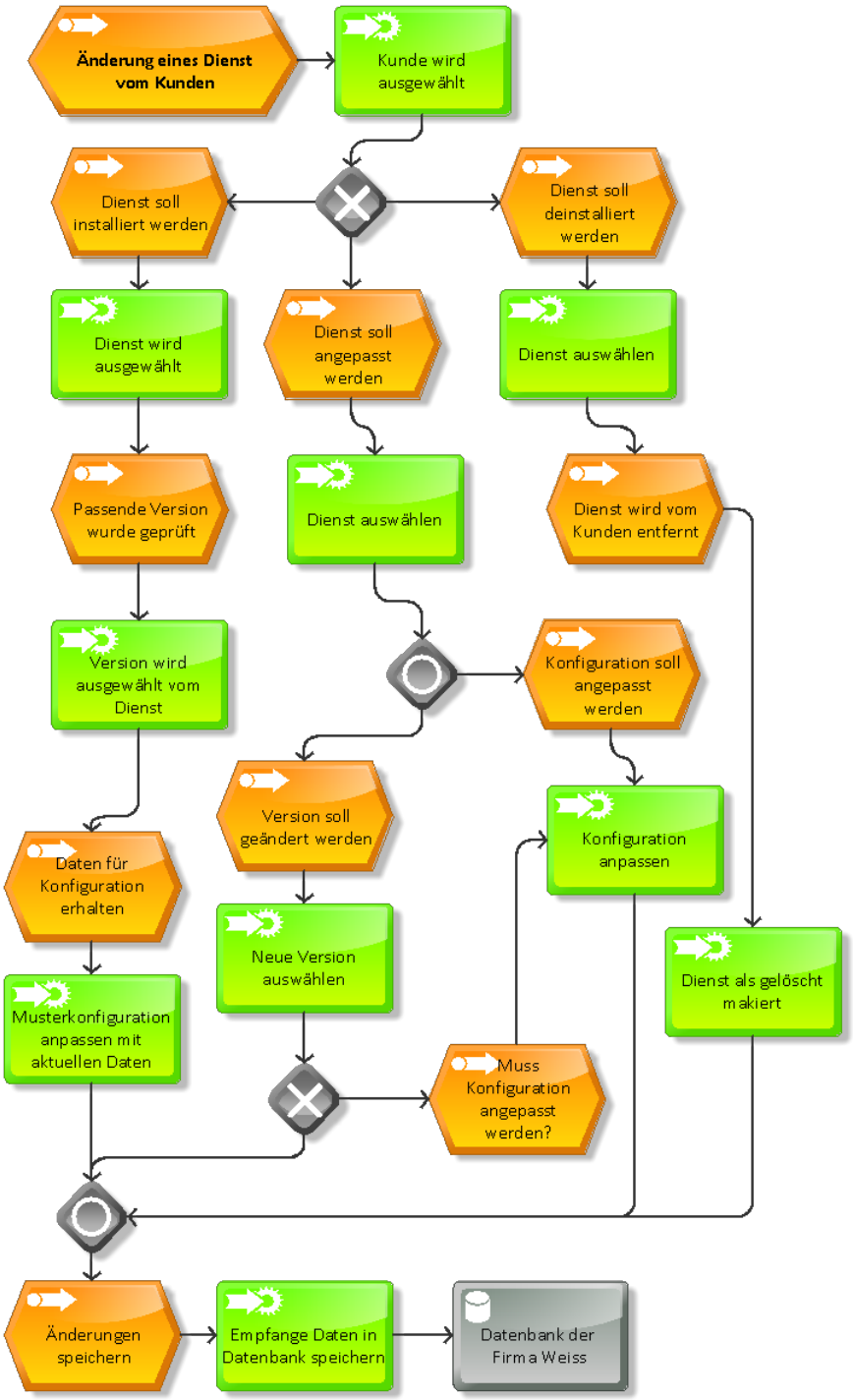
A.1 Zeitplanung

Planung und Konzeption	**20**
Konzeption API und Datenbank	8
Konzeption Oberfläche	2
Konzeption Service	5
Projektstruktur	2
Ressourcenplanung	3
Programmierung	**32**
Datenbank Tabellen	1
API	9
Oberfläche Frontend	10
Oberfläche Backend	2
Dienst	10
Abnahme	**4,5**
Test	2
Schulung	2,5
Dokumentation	**13,5**
Projekt Dokumentation	7,5
Technische Dokumentation	3
Anwender Dokumentation	2
Installations Dokumentation	1
Gesamt	**70 h**

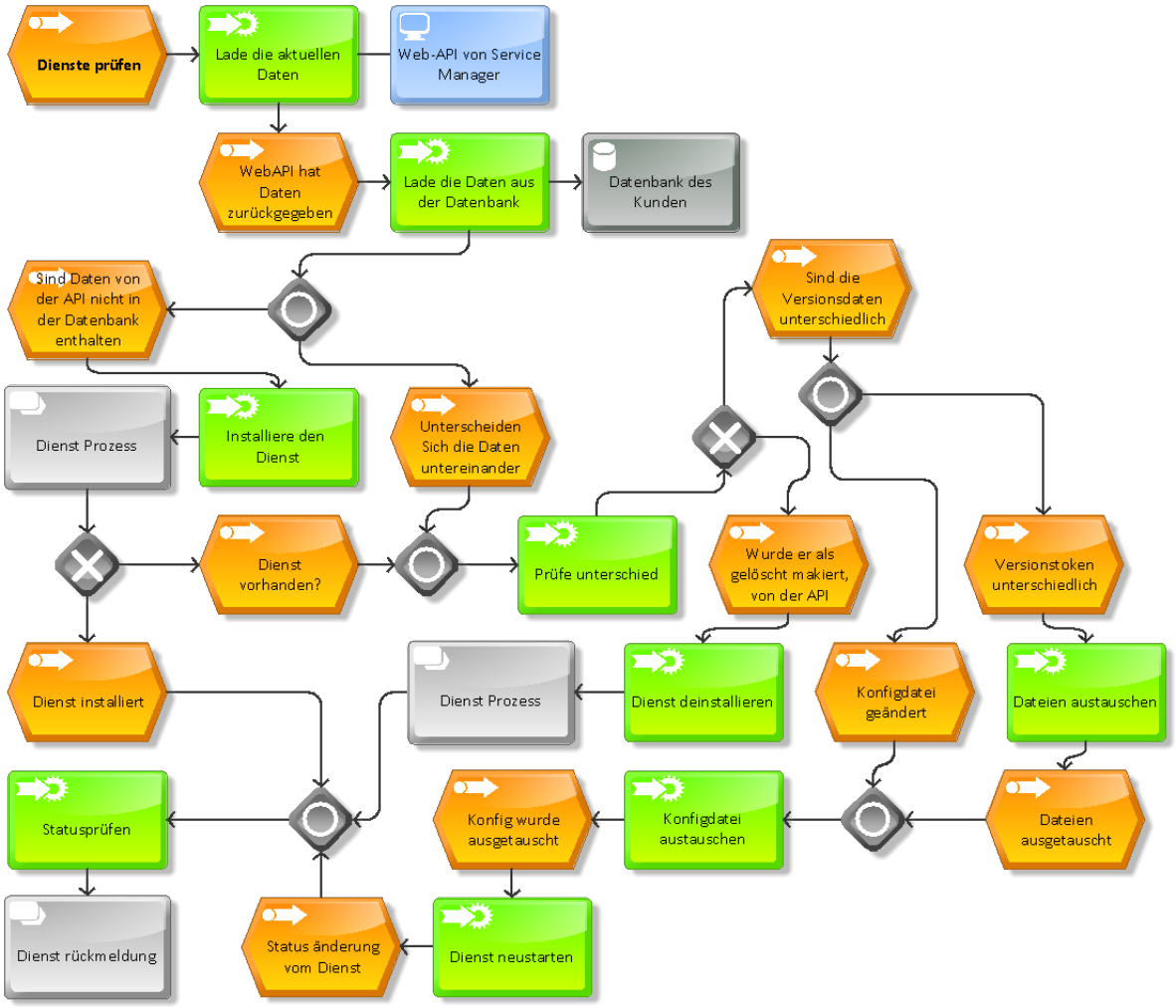
Tabelle 4: Detaillierte Zeitplanung

A.2 Anwendung

A.2.1 Verwaltung von Diensten



A.2.2 Dienstprüfung



A.3 Oberfläche

A.3.1 Konzept

Kunden Nr.:	genau / ungefähr	10123213	Name	Max Mustermann	Suchen ...

#	Kunden Nr.	Name	Intialisiert ?
[EDIT]	10123213	Max Mustermann	Ja
[CREATE]	232131	Josef Adam	Nein

<	Erste Seite	Letzte Seite	>	Seite 1

Abbildung 1: Kundenliste

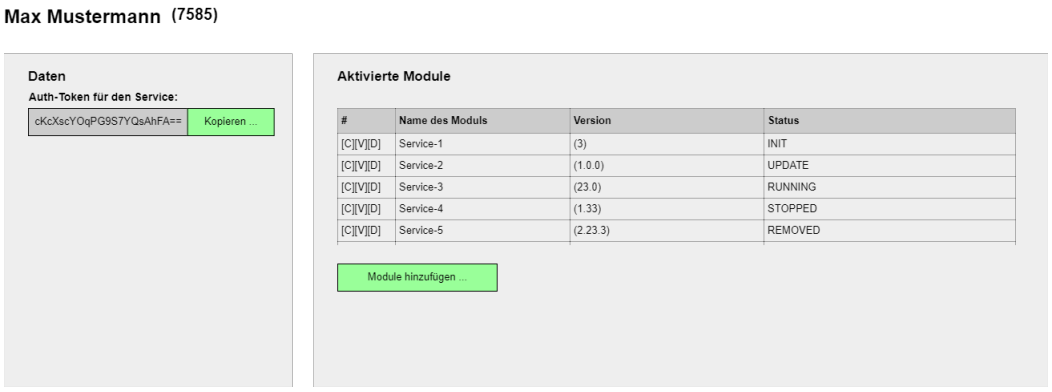


Abbildung 2: Kundenansicht

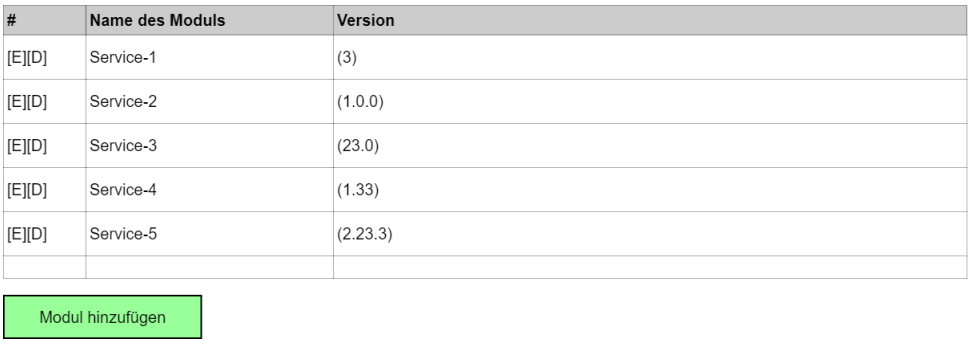


Abbildung 3: Dienstliste

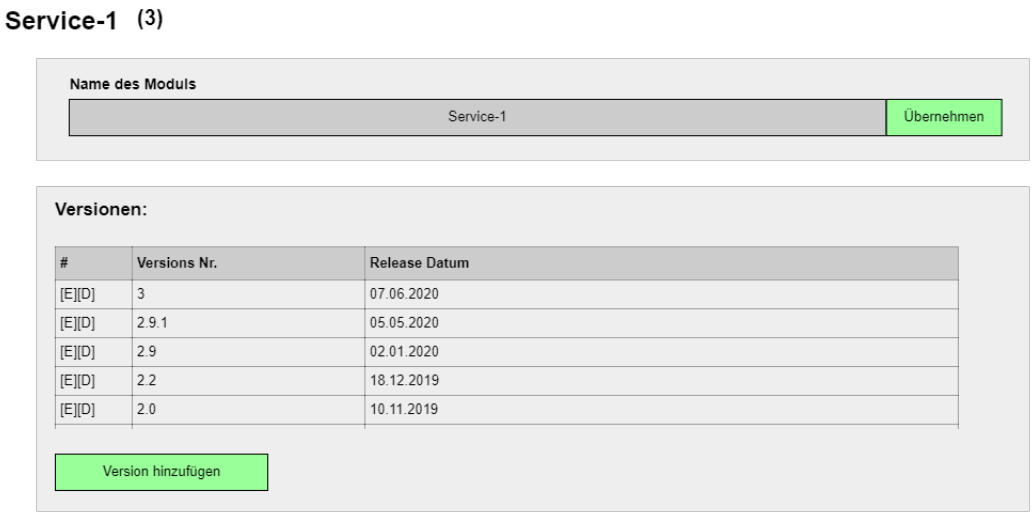


Abbildung 4: Dienstansicht

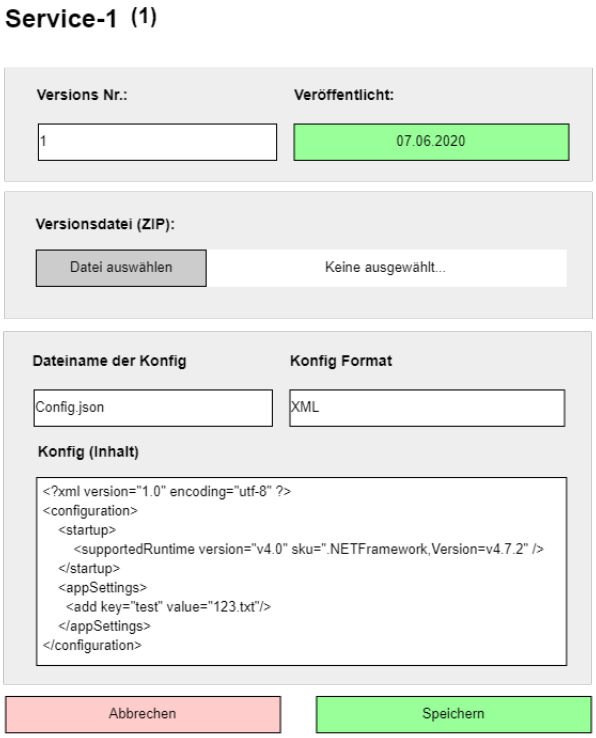


Abbildung 5: Versionseditor

A.3.2 Echtsystem

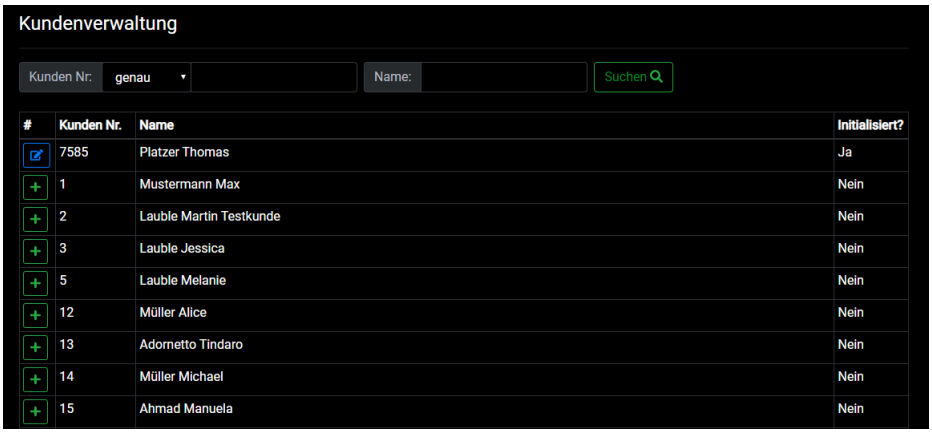


Abbildung 6: Kundenliste

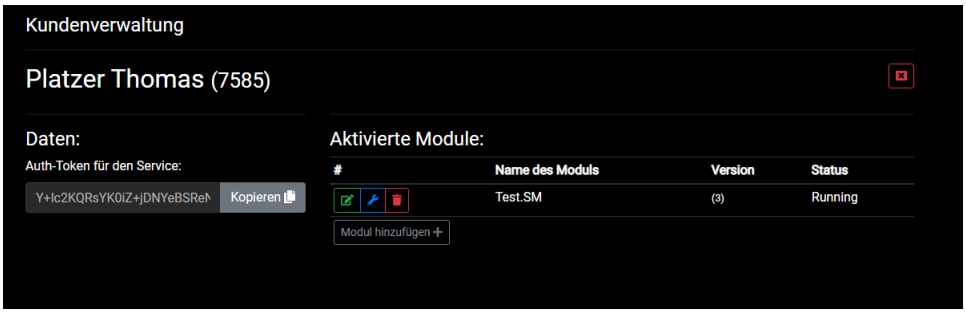


Abbildung 7: Kundenansicht

Abbildung 8: Dienstliste

Abbildung 9: Dienstansicht

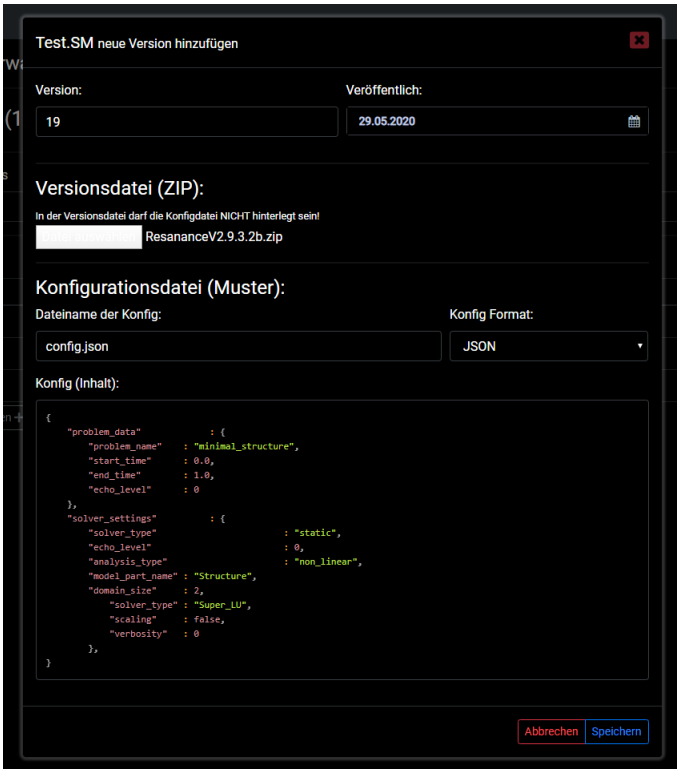


Abbildung 10: Versionseditor

A.4 Klassendiagramme

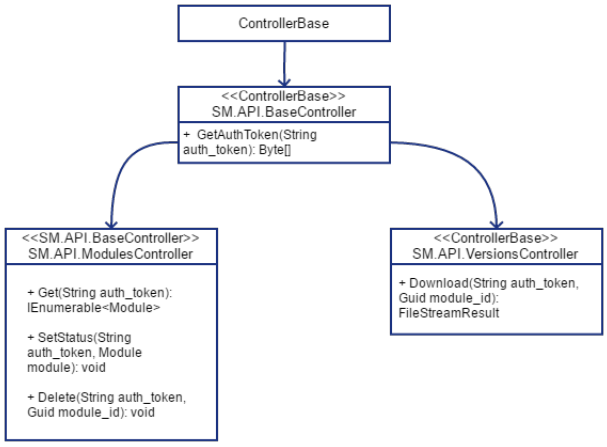


Abbildung 11: Controller-Klassen der API Anwendung

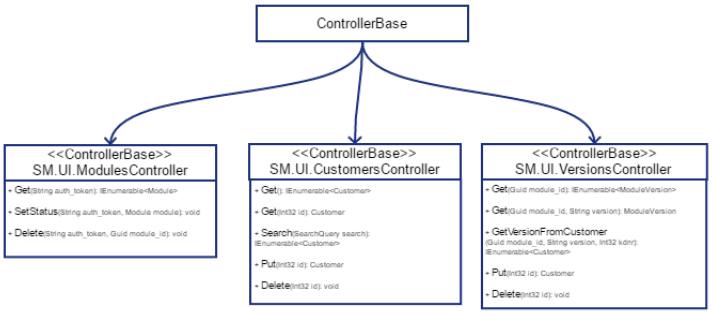


Abbildung 12: Controller-Klassen der Oberflächen/UI Anwendung

A.5 Tabellenmodell

	SM_Modules_Installed	
PK	ModuleId	GUID
	ServiceName	varchar(255
	Version	varchar(16)
	ValidationToken	binary
	Created	timestamp
	Modified	timestamp
	Deleted	timestamp
	IsActive	bit

Abbildung 13: Tabelle die in der Kunden-Datenbank ist

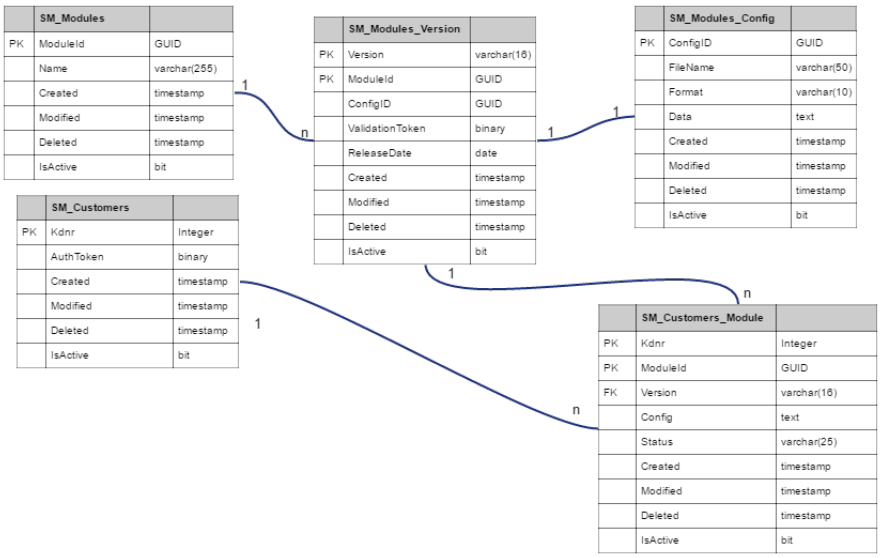


Abbildung 14: Tabellen in der Datenbank der Firma Weiss

A.6 Quellcode

Listing 1: Quellcode zum importieren der Funktionen, um Dienste verwalten zu können

```
// Importieren der DLL AdvAPI32
// ------------------------------
[DllImport("advapi32.dll")]
// Funktion um SC-Manager zu oeffnen, damit der Zugriff auf Dienstpool gewaehrt wird
public static extern IntPtr OpenSCManager(string lpMachineName, string lpSCDB, int scParameter);
[DllImport("advapi32.dll")]
// Funktion um Dienste hinzuzufuegen im Dienstpool
public static extern IntPtr CreateService(IntPtr SC_HANDLE, string lpSvcName, string lpDisplayName,
int dwDesiredAccess, int dwServiceType, int dwStartType, int dwErrorControl, string lpPathName,
string lpLoadOrderGroup, int lpdwTagId, string lpDependencies, string lpServiceStartName, string lpPassword);
[DllImport("advapi32.dll")]
// Schliesse den SC-Manager
public static extern void CloseServiceHandle(IntPtr SCHANDLE);
[DllImport("advapi32.dll")]
// Starte den Diensts
public static extern int StartService(IntPtr SVHANDLE, int dwNumServiceArgs, string lpServiceArgVectors);
[DllImport("advapi32.dll")]
// Oeffne den Dienst vom Dienstpool, um Dienst zu bearbeiten
public static extern IntPtr OpenService(IntPtr SCHANDLE, string lpSvcName, int dwNumServiceArgs);
[DllImport("advapi32.dll")]
// Entferne den Dienst vom Dienstpool
public static extern int DeleteService(IntPtr SVHANDLE);
```

Listing 2: Quellcode um Dienste installieren zu können (DLLs müssen importiert sein!)

```
// Installations Methode
// ----------------------
public static Boolean Install(string servicePath, string serviceName, string serviceDisplayName)
{
    // .... Default werden erstellen und zuweisen .....

    // ServiceController-Manager wird geoeffnet
    IntPtr sCtrlHandler = OpenSCManager(null, null, SC_MANAGER_CREATE_SERVICE);
    // Konnte der SC-Manager geoeffnet werden?
    if (sCtrlHandler != IntPtr.Zero)
    { // Ja
        IntPtr sv_handle = CreateService(sCtrlHandler, servicePath, serviceName, SERVICE_ALL_ACCESS,
            SERVICE_WIN32_OWN_PROCESS, SERVICE_AUTO_START, SERVICE_ERROR_NORMAL,
            serviceDisplayName, null, 0, null, null, null);

        // Konnte der Dienst zum Dienstpool hinzugefuegt werden?
        if (sv_handle == IntPtr.Zero)
        {// Konnte nicht hinzugefuegt werden
            // Schliesse den SC-Manager
            CloseServiceHandle(sCtrlHandler);
            return false; // Dienst konnte nicht zum Dienstpool hinzugefuegt werden!
        }
        else
        { // Wurde hinzugefuegt
            // Hier wird der Dienst gestartet
            if (StartService(sCtrlHandler, 0, null) == 0)
            {// Wenn 0 zurueck gegeben wird, konnte der Dienst nicht gestartet werden!
                return false; // Dienst konnte nicht gestartet werden!
            }

            // Schliesse den SC-Manager
            CloseServiceHandle(sCtrlHandler);
            return true; // Dienst konnte erfolgreich hinzugefuegt und gestartet werden!
        }
    }
    else
        return false; // SC-Manager konnte nicht geoeffnet werden!
}
```

Listing 3: Quellcode zum entfernen der Dienste (DLLs müssen importiert sein!)

```

// Deinstallations Methode
// ------------------------
public static Boolean Uninstall(string serviceName)
{
    // .... oeffne SC-Manager(=sCtrlHandler); Siehe Install Methode! ....

    int DELETE = 0x10000; // Delete
    // Oeffne den ServiceHandler
```

```
10      IntPtr serviceHandler = OpenService(sCtrlHandler, serviceName, DELETE);
11      if (serviceHandler != IntPtr.Zero)
12      { // ServiceHandler konnte geoeffnet werden
13          if (DeleteService(serviceHandler) != 0){ //HINWEIS! Der ServiceHandler stoppt automatisch den Dienst,
                beim deinstallieren!
14              // Schliesse den SC-Manager
15              CloseServiceHandle(sCtrlHandler);
16              return true; // Der Dienst konnte erfolgreich entfernt werden
17          }
18          else
19          {
20              // Schliesse den SC-Manager
21              CloseServiceHandle(sCtrlHandler);
22              return false; // Der Dienst konnte NICHT entfernt werden!
23          }
24      }
25      else
26          return false;
27  }
```