Outlook/Exchange Adapter für CGM G3 Clinical Information System MRP

Krainer Philipp

BACHELORARBEIT

Nr. 1310458007-B

eingereicht am
Fachhochschul-Bachelorstudiengang

Medizin- und Bioinformatik

in Hagenberg

im Dezember 2016

Diese Arbeit entstand im Rahmen des Gegenstands

IGS

im

Sommersemester 2016

Betreuer:

Oliver Krauss MSc

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 1. Dezember 2016

Krainer Philipp

Kurzfassung

Diese Arbeit setzt sich mit der Implementierung eines Moduls für MRP mit der EWS API auseinander. Das MRP-System generiert mit Hilfe von Ressourcen und Regeln Termine, welche in das Microsoft Outlook übertragen werden. Ziel ist es die Verbindung von MRP und Microsoft Outlook herzustellen. Das Modul ist in JAVA programmiert. Mit Hilfe der EWS API wird die Verbindung zum Server geschaffen.
Bei der Implementierung wird geachtet, dass die Richtlinien von der Zielsetzung eingehalten werden. Es wird ein Konzept zur Ansteuerung mittels MRP vorgestellt. In der Implementierung werden die Funktionalitäten sowie Methoden beschrieben, die verwendet worden sind. Weiters werden in der Implementierung die EWS Funktionen angewendet und an das MRP angepasst. Die einzelnen Funktionen werden mittels der Methode des benutzerinteraktiven Testens überprüft. Die verschiedenen Funktionalitäten werden getestet und die Ergebnisse diskutiert.

Abstract

This thesis deals with the implementation of a module for the MRP System using the EWS API. The MRP System generates Appointments using resources and rules. These Appointments get exported to Microsoft Outlook. The connection of MRP and Microsoft Outlook presents the goal of this thesis. The programming language JAVA is used to implement the functions and methods of this software. The EWS API handles the requests from the MRP System and communicates with the server. The implementation follows the rules of the given topic. There is a concepts for using MRP and EWS. The implementation shows used methods and functions for the implementation with EWS. The implementation contains the EWS related functions as well as the methods, which are adapted for the MRP System. The functions as described in the implementation are tested with the method called userinteractive testing. These functionalities are tested and the results are discussed.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Arbeit ist der praktische Teil der Bachelorarbeit. Diese wurde im Rahmen des Semesterpraktikums vom 01.03 bis 30.06.2016 verfasst. Das Praktikum im Umfang von 16 Wochen wurde bei der Firma CGM Clinical Austria am Standort Linz absolviert. CompuGroup Medical (CGM) ist ein österreichischer Softwarehersteller, der umfassende IT-Lösungen zur Optimierung des Gesundheitswesens produziert. Software-Lösungen, welche die Prozesse von niedergelassenen Ärzten und deren Personal sowie von medizinischen, pflegerischen und administrativen Krankenhauspersonal unterstützen. Die CGM beschäftigt in Österreich ca. 250 Mitarbeiter (https://www.cgm.com/at/index.de).

1.1 Zielsetzung

Das Modul Multidimensionale Ressourcen Planning (MRP) ermittelt Termine für Operationen in Krankenhäusern unter der Berücksichtigung der beteiligten Ressourcen (Personen, Geräte, Material) und vor und nachgelagerten Untersuchungen inklusive Betten und Zimmern. Ziel dieses Projektes ist es für die ressourcenbezogenen Daten (z.B. Dienste von Ärzten , Operationstermine etc.) eine Möglichkeit zu schaffen, Termine in den persönlichen Kalendern der beteiligten Personen (Teams) anzuzeigen. Damit sollen die Anwender in ihrer gewohnten Umgebung die Ergebnisse des komplexen Planungsprozesses des MRP-Systems vermittelt bekommen. Es wird davon ausgegangen, dass Termine in Microsoft Outlook eingetragen werden. Dies soll automatisch nach der Generierung von vorgegebenen Terminvorschlägen geschehen. Diese Terminvorschläge werden als Teil des MRP-Systems generiert. Die Aufgabe war es mit Hilfe der Java Exchange Web Services (EWS) Application Programming Interface (API) die Daten an das Microsoft Outlook zu senden bzw. an den Server zu übermitteln. Die EWS API erleichtert das Arbeiten mit Microsoft Outlook.

Bei flexibler API, kann sowohl Java, C# als auch Extensible Markup Language (XML) verwendet werden. Die Termine werden dabei in Microsoft Outlook-Kalendern dargestellt. Änderungen von Terminen im Microsoft Outlook haben keine Auswirkung auf das MRP System. Weiters können sogenannte „Appointments“ mit Hilfe von vordefinierten Filtern gesucht und exportiert werden. Dies ermöglicht eine einfachere Handhabung von Abfragen. Die Appointments werden dann in Outlook-Termine umgewandelt und können so exportiert werden.

Kapitel 2

State of the Art

Im folgenden Kapitel werden die Features vom MRP-Planungssystem sowie der EWS Schnittstelle dargestellt.

2.1 EWS API

Die Java Exchange Web Services (EWS) Application Programming Interface (API) ist eine Schnittstelle, die von Microsoft bereitgestellt wird, welche die Kommunikation mit einen Microsoft Exchange-Server zulässt. Die API kann mit Java, C# angesteuert werden und beinhaltet einen umfangreichen Funkionskatalog um die verschiedenen Funktionen in Microsoft Outlook anzusteuern. Diese Funktionen dienen zur Verwaltung von Emails, Kontakten, Aufgaben, Terminen, Kalendern sowie Rechten. Die API wurde entwickelt um programmgesteuerte Abfragen und Aufgaben mit Hilfe eines Microsoft Outlookservers zu realisieren. Sie unterstützt synchrone sowie asynchrone Abfragen und Anfragen an den Server, die zur Abfrage von Verfügbarkeiten und Benutzerrechten sowie von Microsoft Outlookelementen und Ordnern dienen. Weiters kann mit Hilfe der API gefiltert und gesucht werden, weiters lässt die EWS API einen Mehrbenutzerbetrieb zu.[7]

HyperText Transfer Protocol und HyperText Transfer Protocol Secure

Das Internetprotokoll HyperText Transfer Protocol (HTTP) beschreibt die zustandslose Übertragung von Internetdaten. Die Übertragung läuft über die Webmethoden GET, PUT und POST. HTTP verwendet die Standardports, welche in der Transmission Control Protocol (TCP)-Richtlinie definiert worden sind.
HyperText Transfer Protocol Secure (HTTPS) stellt die sichere Verbindung von HTTP dar, wobei die Verschlüsselung über Secure Socket Layer (SSL) erfolgt. Webseiten werden mittels HyperText Markup Language (HTML),

JavaScript und Personal Home Page (PHP) angesteuert, wobei die Daten mit Extensible Markup Language (XML) und JavaScript Object Notation (JSON) übertragen werden.[7, 6]

2.1.1 Funktionen

Exchange Service

Der Exchange Service ist der zentrale Service, welcher Aufgaben verarbeitet und in Verbindung mit dem HTTP(S) Protokoll den Exchange-Server ansteuert. Dieser Service stellt die Verbindung mit dem Server mittels Benutzername, Passwort, Serveradresse her. Außerdem sind zusätzliche Informationen zur Serverversion und Übertragungsart gespeichert. Um die Kommunikation mit dem Server zu ermöglichen, muss die EWS Funktion freigeschaltet sein.[7]

Items und Folder

Die Funktionen der EWS API funktionieren nach dem Prinzip der Eindeutigkeit. Dies bedeutet, dass jeder Ordner, jedes Mail und jeder Termin seine eigene ID besitzt, welche vom Server festgelegt wird. Dabei gibt es keine Einschränkung, da jedes Element über die gleiche Priorität verfügt. Weiters kann mit Hilfe dieser eindeutigen ID jedes gesonderte Element abgerufen und beliebig verändert werden. Die Ordner verfügen über eine eindeutige ID mit der auf den ganzen Ordner zugegriffen werden kann. Auch besitzen spezielle Ordner wie z.B. der Posteingang sogenannte *WellKnownFolderNames*, welche wie Aliase bzw. Pseudonyme für IDs fungieren.[7]

Email

Emails können mit Hilfe der API gesendet und empfangen werden. Dabei besitzt jedes Email einen zugeordneten Ordner. Die Emails werden automatisch versendet sobald sie am Server eingetroffen sind. Auch können Emails von beliebigen Konten abgefragt und verändert werden. Vordefinierte Kategorien ermöglichen eine organisierte Verwaltung. Es können Dateien aus dem lokalen Dateisystem an Emails angehängt werden. Wichtige Emails können per Filter gesucht und gefunden werden sowie Posteingangsregeln erstellt und abgerufen werden. Emails können bei Bedarf in einen Unterordner verschoben werden. [8]

Termine / Appointments

Termine können erstellt, abgerufen und gelöscht werden. Sie repräsentieren eine Hauptfunktionalität im Outlook. Jeder Termin beinhaltet Datum, Zeit, Ort sowie Teilnehmer. Die Termine werden in den persönlichen Kalendern

abgelegt. Weiters können den Terminen optionale Teilnehmer hinzugefügt werden. Periodisch wiederholende Termine werden als Serientermine hinterlegt. Neben dem Hauptkalender können zusätzliche Kalender erstellt werden. Eine weitere Funktionalität stellt die Abfrage von Verfügbarkeiten dar. Diese liefern in weiterer Folge Terminvorschläge. Dabei wird auch die terminliche sowie räumliche Verfügbarkeit mitgeliefert. [8]

Tasks

Ein weitere Funktionalität der API stellen Tasks dar, welche erstellt und abgerufen werden können. Tasks enthalten eine Liste von Aufgaben, welche Personen zugeordnet werden und von diesen abgearbeitet werden. Einzelne Aufgaben können priorisiert und kategorisiert werden.[7]

Kontakte

Kontakte stellen neben den Terminen eine wichtige Rolle in der Microsoft Outlookumgebung dar. Die Kontakte werden im Adressbuch verwaltet und können angelegt, abgerufen sowie gelöscht werden. Kontakte können aus vordefinierten Dateien importiert und exportiert werden. Das Adressbuch enthält Kontakte sowie Kontaktgruppen. In den Kontakten sind Namen, Telefonnummern, Emailadressen und persönliche Daten hinterlegt. In dem Adressbuch kann gesucht und gefiltert werden.[8]

2.2 Multidimensionales Ressourcen Planning (MRP)

Das MRP-System ist zur Planung von Krankenhausterminen entwickelt worden. Die Hauptaufgabe von MRP ist es aus gegebenen Ressourcen (Personen, Räume, Geräte, Betten, Materialien) in Verbindung mit Regeln Planungskalender für einzelne Räume zu erstellen. Diese Kalender repräsentieren Operationsplanungen, die eine zentrale Einheit in einem Krankenhausinformationssystem (KIS) darstellen.
Ein KIS besteht in erster Linie aus den Zentralkomponenten Organisation, Bettenmanagement, Aufnahme, Entlassung sowie Operationsplanungen. MRP deckt nicht die zentralen Komponenten ab, da sich dieses System nur mit der Planung von Räumen und Terminen mithilfe von Ressourcen beschäftigt. Es übernimmt nicht die Verwaltung des Krankenhauses! [4, 5]

2.2.1 Funktionsweise

Das MRP System generiert aus vorgegebenen Daten geeignete Termine und Pläne für Räume des Krankenhauses. Es werden die benötigten Ressourcen

gesammelt und in einer zentralen Datenbank verwaltet. Aus den Datensätzen werden mittels Ressourcenmanager die Ressourcen zusammengetragen und Appointments generiert. Diese werden dann mit Hilfe des Kapazitätsmanagementtools den verfügbaren Kapazitäten der einzelne Ressourcen zugeordnet und in einer Kalenderansicht dargestellt. [4]

2.2.2 Masterdata

Jedes Feature im MRP-System folgt der selben Struktur. Jedes Feature besteht aus den Teilen API, Component, Datenbank und Service. Dabei wird die Drei-Schichten-Architektur realisiert. Dies bedeutet, dass es drei getrennte Implementierungen hinsichtlich jeder Schicht gibt, welche als eigenständig fungieren. Die unterste Schicht wird als Datenbank-Schicht bezeichnet und ist zuständig für die Verwaltung von persistenten Daten. Die Mittlere Schicht wird als Business-Logik bzw. Verarbeitungsschicht bezeichnet. Diese ruft die Daten von der untersten Schicht ab und leitet sie gegebenenfalls an die oberste Schicht weiter. Es besteht die Möglichkeit für Berechungen und Änderungen der Daten. Die oberste Schicht stellt die Präsentationsschicht dar und ist zuständig für die visuelle Darstellung der Daten. Die 3-Schichten-Architektur hat den Vorteil, dass die Aufgaben der Schichten klar von einander getrennt sind. Weiters gibt es die Component-Service-Implementation, welche aus den Komponenten, Datenbank, Service, und API besteht.[2, 4]

Struktur

Jedes Feature folgt der Component-Service-Implementation, welche aus folgenden Teilen besteht(siehe Abb. 2.1)[1–4]:

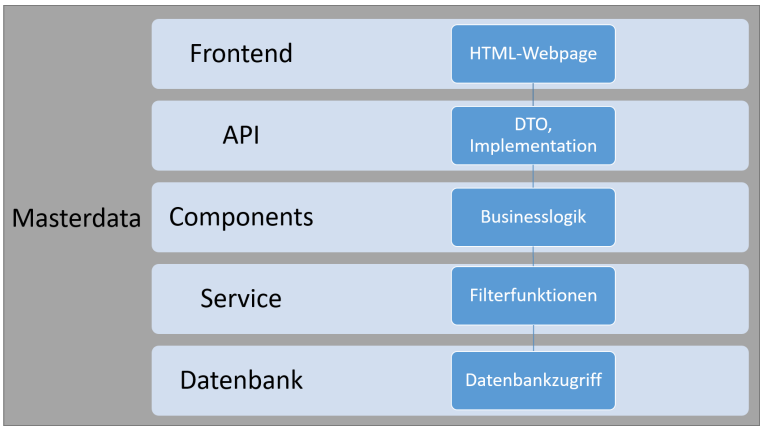

Abbildung 2.1: Architekturdiagramm

- *Datenbank:* Das Datenbankfeature verwaltet alle Daten in einer Datenbanktechnologie, welche variabel auswählbar ist. Das Speichern und Abrufen von Daten wird über den jeweiligen Datenbanktreiber betrieben, welcher mit einer standardisierten Datenbankansteuerung kommuniziert. Dieser befindet sich innerhalb eines Data Access Object (DAO). Dieses beinhaltet die relevanten Datenbankzugriffsfunktionen und liefert ein Data Transfer Object (DTO).

- *Service:* Der Service erhält Daten von der Datenbank und schickt sie an die Components weiter. Die Services dienen den Components für eine erleichterte Abfragemöglichkeit von Datenobjekten. Weiters benötigen die Services vordefinierte Filterfunktionen. Die Funktionalität der Services gehört der Business-Logik-Schicht an.

- *Components:* Die Components stellen im Feature die Hauptbestandteile dar und sind ein Teil der Businesslogik. Die Components repräsentieren die eigentliche Logik, da in diesen Funktionen die Verarbeitung der Daten erfolgt. Die Components erhalten die Daten aus der Datenbank und wandeln diese in *Data Transfer Objects* um. Die Objekte werden an die API weitergeleitet.

- *API:* Die API ist ein Teil der Businessschicht und stellt die Verbindung zum Frontend da. Die API bekommt die Daten von den Components und wandelt diese in webkonforme Übertragungsobjekte um, welche an das Frontend weitergeleitet werden. Außerdem ist die API zuständig für das Empfangen von Daten aus dem Frontend.

- *Frontend:* Das Frontend repräsentiert die Präsentationssschicht, welche für den jeweiligen Benutzer sichtbar ist. Im Frontend werden die Benutzerinteraktionen verwaltet, Daten zu Transferobjekte umgewandelt und an das Backend gesendet.

2.2.3 Hospital Information System

Das Hospital Information System (HIS) von CGM wird unter der Produktbezeichnung G3 HIS geführt und besteht aus mehreren Modulen. Die Zentralkomponenten eines KIS sind in diesem System integriert und bilden eine einheitliche Weboberfläche. Diese ist betriebssystemunabhängig und wird in einem vom G3 HIS unterstützten Browser verwendet. Das G3 HIS Projekt umfasst mehrere Projektgruppen, die in die unterschiedlichen Aufgaben (Organisation, Bettenmanagement, Aufnahme, Entlassung, Operationsplanungen) eines Kranken-

hauses unterteilt sind. Sie bieten eine umfangreiche Konfiguration und Verwaltung der einzelnen Funktionsgruppen bzw. Module an. Die Software stellt die 3. Generation dar und wird ständig weiterentwickelt.

Kapitel 3

Konzept und Design

In diesem Kapitel wird das Konzept zur Übertragung von Daten aus dem MRP ins Microsoft Outlook beschrieben.

3.1 Ansteuerung mittels MRP

MRP ist eine von CGM entwickelte Software, die Termine für Operationen in Krankenhäusern unter der Berücksichtigung aller beteiligter Ressourcen optimiert und berechnet. Das MRP-Modul beinhaltet folgende Funktionalitäten[1, 2, 4]:

1. Zu Beginn werden die verfügbaren Ressourcen (Personen, Räume, Geräte, Betten, Materialien) angelegt und zugeordnet.
2. Zuordnung der ressourcenbezogenen Daten: Jeder Ressource werden Informationen zugeteilt.
3. Mittels der Ressourcen werden mithilfe von Kapazitäten(terminliche Verfügbarkeiten) Organisationseinheiten (Zeitbereiche mit verknüpften Ressourcen) gebildet.
4. Aus diesen Einheiten werden Vorschläge gebildet. Diese Vorschläge beinhalten lediglich eine Zeitspanne. Um einen Termin festlegen zu können, müssen Ressourcen definiert werden. Dies entsteht aus den bereits hinterlegten Personendaten sowie aus den Verfügbarkeitsabfragen der EWS Schnittstelle. Mit diesen erweiterten Funktionen können konkrete Termine generiert u. nach deren Bestätigung ins Outlook exportiert werden. Die exportierten Termine können im betreffenden Kalender eingesehen werden. Das Anlegen von Ressourcen u. das Selektieren von exportierten Terminen erfolgt manuell.

Die Erstellung eines MRP Moduls erfolgt nach den Programmierrichtlinien von CGM. Dieses Modul dient zum Export von generierten Terminen ins

Outlook. Weiters werden Filter- und Suchfunktionen bereitgestellt. Diese können MRP-Appointments und Outlook-Appointments filtern und suchen.

Kapitel 4

Implementierung

In Abstimmung mit CGM wurde das Konzept aus Kapitel 3 für die Umsetzung gemäß Zielsetzung herangezogen. Dabei wurden die Funktionen der EWS-API verwendet und in den Methoden der nachfolgenden Module aufgerufen:

4.1 Microsoft Outlook

Die nachfolgenden aufgelisteten Packages beziehen sich auf die Klassen: *Packagename + Service* , *I + Packagename + Service* und *Packagename + Dto* (siehe Klassendiagramm Abb. 4.1)
Die jeweilige Klasse des Packages implementiert eine vordefinierte Schnittstelle (Interface), welche einen vordefinierten Benennungsschema folgt. Die Funktionalitäten sind unabhängig vom MRP-System, können aber von diesem verwendet werden. Die für die Arbeit entwickelten Datentypen sind im Package *Model* definiert. Diese Klassen können gleichnamige Klassen im MRP haben, haben aber keine Beziehung zu diesen Klassen.

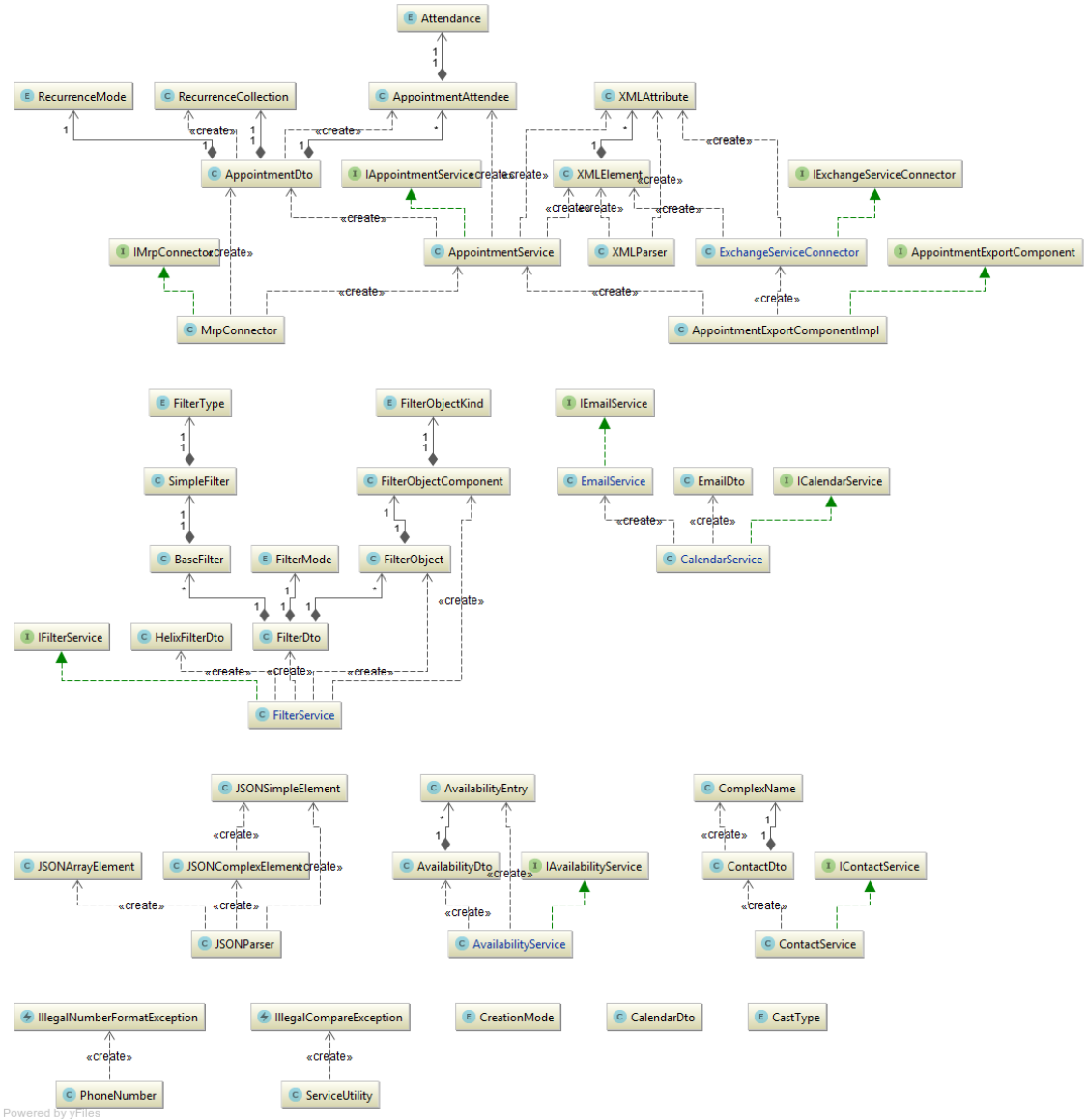

Abbildung 4.1: Klassendiagramm

4.1.1 Exchangeservice

Der Exchangeservice dient als zentrale Verbindung zum Microsoft Outlook. Die Features bzw. Methoden, welche mit dem Exchange-Server kommunizieren, verwenden ExchangeService als Parameter. Die Klasse wird *ExchangeServiceConnector* genannt und ist zuständig für die Verbindung zum Server sowie für die Anfragen der Features. Dieser Service bezieht sich ausschließlich auf das eigene Outlookkonto bzw. auf das Konto, welches mit Passwort

und Benutzername angegeben wird. Weiters besteht die Möglichkeit die Exchangeserverversion festzulegen. Diese Klasse implementiert das zugehörige Interface *IExchangeServiceConnector*.
Die Serververbindung wird mit Hilfe der Methode *createServerConnection* initialisiert, wobei der Benutzername und das Passwort als Parameter übergeben werden. Die gleichnamige Methode verwendet zusätzlich den Parameter Server-URL. Die Methode *createServerConnection* baut mit Hilfe der Daten eines XML-Dokuments eine Serververbindung auf. Die genanten Methoden retournieren eine Instanz des Exchangeservice. Dieser Service existiert nur einmal zur Laufzeit. Damit wird garantiert, dass immer der gleiche Server angesteuert wird. Mit der Methode *saveSettingsToXML* können die Servereinstellungen in einem XML-Dokument gespeichert werden.

4.1.2 Model

Die verwendeten Objekte bzw. Datentypen in den einzelnen Features bzw. Packages werden im Model repräsentiert. Die einzelnen Klassen repräsentieren die jeweiligen Datentypen, welche einer vordefinierten Struktur folgen. Diese folgt der Bean-Definition. Diese bedeutet, dass alle Membervariablen *private* sind und daher wird auf die Variablen mit Getter und Setter zugegriffen. Die einzelnen definierten Datentypen besitzen einen öffentlichen Default-Konstruktor ohne Parameter.

4.1.3 Kalender

Mittels Kalenderservice können Kalender angelegt und abgerufen werden. Der *CalendarService* implementiert das zugehörige Interface *ICalendarService* und verwendet den Datentyp *CalendarDto*.
Die Mehode *getCalendars* ruft die verfügbaren Kalender einer Emailadresse ab. Die Emailadresse repräsentiert Personen oder Räume. Zusätzliche Kalender werden durch die Methode *createNewCalendar* angelegt. Auch gibt es die Methode *sendWarningToUser*, mit dieser kann ein Email an den betreffenden Benutzer versendet werden. Die Emailnachricht enthält die betreffende Warnung.

4.1.4 Appointments

Mit Hilfe des Appointmentservices können Microsoft Outlook-konforme Appointments angelegt, verwaltet und abgerufen werden. Weiters ist möglich Appointments in einem XML-Dokument zu speichern. Das XML Dokument beinhaltet die Datenfelder aus dem *Model* in einer validen XML-Struktur. Diese XML-Dokumente können zur Erstellung von Appointments verwendet werden. Es besteht die Möglichkeit Konflikte zu betreffenden Appointments vom Server abzufragen. Der *AppointmentService* implementiert das zugehörige Interface *IAppointmentService* und verwendet den Datentyp *Appoint-*

mentDto.
Mit der Methode *createAppointment* kann ein Appointment angelegt werden. Der *ExchangeService* wird als Parameter übergeben. Die Methode *buildAppointment* erstellt den Datentyp *Appointment*. Die Methode *createAppointments* ermöglicht das gleichzeitige Speichern und Anlegen von Appointments. Eine weitere Funktionalität ist durch die Methode *hasConflictedAppoinments* gegeben. Diese Funktion überprüft mit Hilfe des Exchangeservers Konflikte bei betreffenden Appointments. Die Funktionen *appointmentListToXML*, *xmlToDto* und *xmlToAppointmentList* ermöglichen die Verwaltung von outlookkonformen Appointments mithilfe von XML.

4.1.5 Availability

Durch den Availabilityservice können Verfügbarkeiten von Personen und Räumen abgefragt werden. Der *AvailabilityService* implementiert das zugehörige Interface *IAvailabilityService* und verwendet den Datentyp *AvailabilityDto*.
Die Hauptfunktion *checkAvailability* ruft mit Hilfe des *ExchangeService* Verfügbarkeiten über die EWS-Schnittstelle ab. Die Methode *checkAvailbilityForPerson* überprüft die Verfügbarkeit von Personen anhand deren Email-Adressen. Mit Hilfe der Methode *checkAvailbilityForRooms* können im Gegensatz zur obengenannten Methode anstelle von Personen Räume abgefragt werden. Die beiden Funktionen *getRoom* und *getAllRooms* liefern Räume zurück. Personen und Räume werden durch die jeweiligen Emails repräsentiert.

4.1.6 Kontakt

Der Kontaktservice ermöglicht das Anlegen, Abrufen und Verwalten von Outlookkontakten in den jeweiligen persönlichen Adressbüchern. Der Kontaktservice arbeitet mit dem Benutzer des Exchangeservices. Der *ContactService* implementiert das zugehörige Interface *IContactService* und verwendet den Datentyp *ContactDto*.
Die Methode *getContacts* ruft persönliche Kontakte des angemeldeten Benutzers aus dem persönlichen globalen Adressbuch ab. Die Kontakte sind unsortiert und können mittels Filter zusätzlich gefiltert werden. Neue Kontakte werden über die Funktion *buildContact* erstellt. Die Funktion *getContactDtos* liefert mithilfe der Funktion *buildContact* eine Liste vom vordefinierten Datentyp *ContactDto*.

4.1.7 Email

Der Emailservice ermöglicht das Senden von bestehenden Emails. Diese beinhalten die relevanten Felder der Emails. Weiters können die Emails aus dem jeweiligen Postfach des Benutzers aufgerufen werden. Der *EmailService* implementiert das zugehörige Interface *IEmailService* und verwendet den

Datentyp *EmailDto*.
Durch die Mehode *sendEmail* können Emails versendet werden. Diese werden in der Mehode *buildEmail* erstellt und enthalten alle relevanten Informationen zu Sender und Empfänger sowie Betreff und Emailtext. Die Methode *getEmails* ruft eine vordefinierte Anzahl Emails aus dem Posteingang ab.

4.1.8 Filter

Die Filter bestehen aus Filterkomponenten, welche vordefiniert sind. Diese Filterkomponenten folgen dem Prinzip der überpersistenten Speicherung. Dies bedeutet, dass bereits gefilterte Daten trotzdem erhalten bleiben. Grundsätzlich ist jede Filterkomponente aus zwei Teilen aufgebaut: Der erste Teil beinhaltet die Informationen zu den Filtern. Der zweite Teil stellt eine Datensammlung dar. Diese Daten bleiben solange erhalten solange die Filterkomponenten aktiv sind. Die Filterkomponenten werden gleichzeitig bzw. hintereinander angewendet, um die logischen Funktionen AND und OR zu repräsentieren Eine weitere Funktionalität ist der Vergleich von Methoden mit den Variablen.
Die Mehode *apply* beschreibt die Hauptfunktionalität des Filters und beinhaltet die Funktionalität, die zur Anwendung eines bestimmten Filters erforderlich ist. Das *FilterDto* beinhaltet sowohl die Filterkomponenten als auch die zu filternden Daten. Mit der Methode *applySimpleFilter* wird ein Filter für eine Sammlung von zu filternden Objekten angewandt. Die Funktion *applyFilter* wird verwendet, wenn die zu filternden Objekte den Typ *AppointmentComponent* darstellen. Die zwei Funktionen *applyHelixFilter* werden verwendet, wenn die Filter vom MRP-System stammen. Die Funktionen *getFiltered* und *getUnfiltered* dienen zum Abrufen von gefilterten und ungefilterten Daten eines *FilterDto*s

4.2 Masterdata-Feature

Als Masterdata werden Features bezeichnet, welche zentrale Daten im MRP verwalten und welche in einer Datenbank gespeichert werden (siehe Abb. 4.2).

4.2.1 Filter

In dem MRP-Feature Masterdata wurde ein Feature hinzugefügt, welches zur Verwaltung von Filtern dient. Die Filter werden in der zentralen Datenbank von MRP gespeichert und über die Weboberfläche von MRP verwaltet, da die Filter zu den Masterdata-Features zählen. Die Filter dienen zur Filterung von MRP-Appointments. Jeder Filter beinhaltet die benötigten Filterkomponenten, welche für das Filtern der Daten verantwortlich sind.

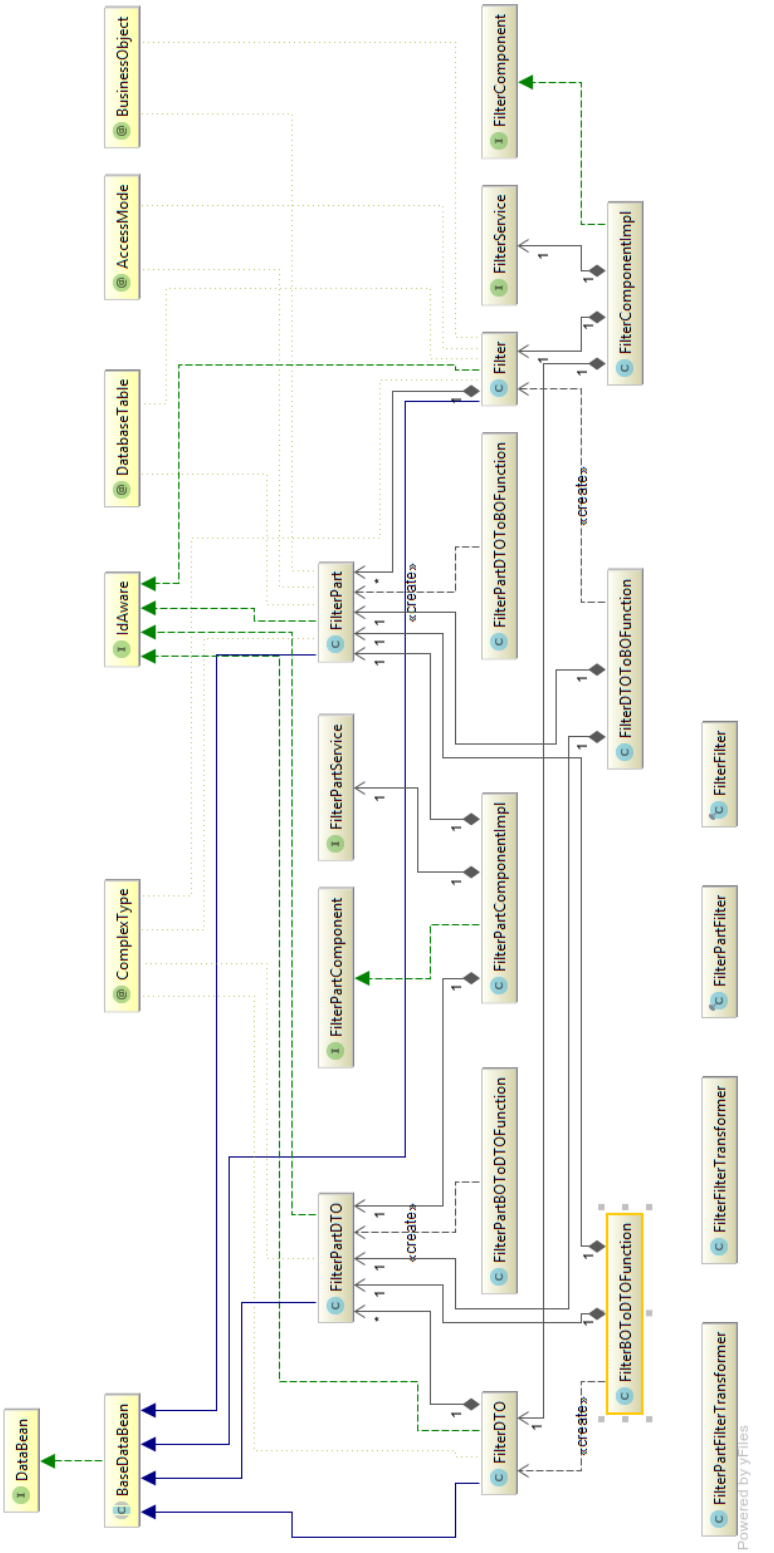

Abbildung 4.2: Klassendiagramm

4.2.2 Filterkomponenten

Jeder Filter hat Filterkomponenten untergeordnet, welche die Informationen zu Filterart, Filtertyp und Filterwert enthalten. Diese Komponenten werden in einer Relation zu den Filtern in der Datenbank abgelegt. Die Filterkomponenten sind immer für einen bestimmten Filter zuständig und können nicht einzeln verwendet werden. Diese Filterkomponenten dienen zum Filtern von Appointments.

4.2.3 Weboberfläche

In der Weboberfläche von dem MRP-System werden die Filter und Filterkomponenten verwaltet. Die Oberfläche wird von G3 HIS bereitgestellt und bietet die Funktonalität von MRP. Mit Hilfe der Filter werden Appointments gesucht und angezeigt. Zukünftig sollen selektierte Appointments ins Microsoft Outlook exportiert werden können.

Kapitel 5

Evaluierung

5.1 Einleitung

Das Outlookmodul ist im MRP-System teilintegriert. Die Testumgebung wird mit Hilfe eines Jenkins-Maven-Testserver realisiert, welcher selbständiges automatisiertes Testen zulässt. Bei einen Jenkins-Maven-Testserver handelt es sich um einen Linux-Server, welcher das gesamte Projekt in regelmäßigen Abständen compiliert und die vorgesehenen Tests ausführt. Die Ergebnisse sind in einer Weboberfläche einsehbar. In dem Repository gibt es die Branches *master* und *dev*. Diese werden in die automatisierte Testumgebung integriert und in fixierten Zeitabständen getestet. Da das zu bearbeitende Modul kein Teil der zwei Branches ist, kann es nicht getestet werden. Durch den Umstand des Testprozesses muss auf andere Testmittel umgestiegen werden.

5.2 Benutzerinteraktives Testen

Auf Grund des Testprozesses wird auf die Methodik benutzerinteraktives Testen zurückgegriffen. Die gewählte Methodik ist besser bekannt als *Whitebox-Test.*[9] Diese Methodik beruht auf der visuellen Kontrolle des Programmierers. Dabei überprüft der Programmierer die Ergebnisse in der grafischen Oberfläche bzw. Konsole. Die Testmethodik dient rein zur visuellen Kontrolle ob die geforderten Datenfelder richtig in der Konsole angezeigt werden. Trotzdem kann mit Hilfe dieser Methodik die Funktion des Programms getestet werden. Dies ist auch möglich wenn kein geeignetes Testframework vorhanden ist. Da das Modul mit einem Server kommuniziert, bei dem die Verarbeitung der Anfragen einen Engpass darstellt, stellt sich automatisiertes Testen mittels Frameworks als aufwendig und komplex dar.

Im Rahmen der Testung werden folgende Funktionen überprüft:

Getestete Funktionen	Tests erfolgreich
Appointments anlegen	16
Appointments abrufen	24
Availablilities abfragen	20
Kalender anlegen	21
Kalender abfragen	21
Kontakte abrufen	14
Adressbuch abfragen	30
Emails senden	40
Emails abrufen	45
Serververbindung initialisieren	5
Serververbindung speichern	5
Serververbindung abfragen	10
Filter verwalten	38
Filterkomponenten verwalten	64
Filterkomponenten zuordnen	23
MRP-Filter verwalten	12
MRP-Filterkomponenten verwalten	17
MRP-Appointments exportieren	0 (keine Tests vorhanden)

Tabelle 5.1: Tests

Kapitel 6

Zusammenfassung

In diesem Kapitel werden die Ergebnisse zusammengefasst und diskutiert.

6.1 Resultate

Bei dieser Arbeit wurde ein Programm implementiert, welches als Modul für das MRP-System der Firma CGM fungiert. Die Implementierung wurde mittels JAVA durchgeführt. Es wurde die Java EWS API verwendet. Die EWS API dient zur Ansteuerung eines Microsoft Outlookservers mithilfe von Programmmethoden. Das Modul verwendete diese API um bereitgestellte Termine aus dem MRP-System an das Microsoft Outlook zu exportieren. Die Daten wurden von weiteren Modulen des MRP-Systems generiert und bereitgestellt. Allerdings fehlte die Integration des Moduls in das gesamtheitliche MRP-System, da dieses zum damaligen Zeitpunkt noch nicht fertiggestellt war. Das Testen erfolgte durch benutzerinteraktives Testen. Dieses beruhte auf einer rein visuellen Betrachtung des Benutzers. Die diesbezüglichen Tests verliefen positiv.

6.2 Diskussion

Das in dieser Arbeit erstellte Programmmodul ist universell einsetzbar, wurde aber geringfügig an das MRP-System angepasst. Die EWS API mit ihren komplexen Datentypen wurden durch das vereinfachte DTO ersetzt. Die universelle Einsetzbarkeit wurde damit ermöglicht. Die Implementierung der verwendeten Filter wurde generisch gehalten, um eine universelle Einsetzbarkeit zu ermöglichen.

Quellenverzeichnis

Literatur

[1] CGM Clinical. „CGM HTML5 Controls Browser“. Documentation to the UI Elements.

[2] CGM Clinical. „G3 HIS Reference Documentation“. Documentation to CGM G3 HIS Sofware.

[3] CGM Clinical. „G3 Reverse Proxy“. Technical Documentation for the G3 HIS Project.

[4] CGM Clinical. „MRP Technical Documentation“. Technical Documentation for the MRP Project.

[5] Peter Haas. *Medizinische Informationssysteme und elektronische Krankenakten*. Springer Verlag, 2004.

[6] *HTTP - Hypertext Transfer Protocol*. zuletzt besucht am 31.10.2016. https://www.w3.org/Protocols/, 2014.

[7] *Java EWS API Getting Started Guide*. zuletzt besucht am 20.09.2016. https://github.com/OfficeDev/ews-java-api/wiki/Getting-Started-Guide, 2016.

[8] T. Joos. *Microsoft Outlook 2013 - Das Handbuch*. O'Rielly Verlag Gmbh, 2013.

[9] Andreas Spillner und Tilo Linz. *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester – Foundation Level nach ISTQB-Standard*. dpunkt, 2005.