Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Einführung

Diese Arbeit ist der Praktische Teil der Bachelorarbeit und wurde im Rahmen des Semesterpraktikums verfasst. Das Praktikum im Umfang von 16 Wochen wurde bei der Firma CGM Clinical Austria am Standort Linz absolviert. CompuGroup Medical (CGM) ist der einzige österreichische Softwarehersteller, der umfassende IT-Lösungen zur Optimierung des Gesundheitswesens produziert. Software-Lösungen, welche die Prozesse von niedergelassenen Ärzten und dessen Personal sowie von medizinischem, pflegerischem und administrativem Krankenhauspersonal unterstützen. Die CGM beschäftigt in Österreich 250 Mitarbeiter.

1.2 Zielsetzung

Das Modul MRP errechnet ideale Termine für Operationen in einem Krankenhaus unter der Berücksichtigung aller beteiligter Ressourcen (Personen ,Geräte, Material) und aller vor und nachgelagerten Untersuchungen sowie Bett und Zimmer. Ziel dieses Projektes ist es für alle ressourcenbezogenen Daten (z.B. Dienste von Ärzten , OP Termin mit spezifischen Ärzten) eine Möglichkeit zu schaffen, Termine in den persönlichen Kalendern der Personen (Teams) anzuzeigen, damit die Anwender in ihrer gewohnten Umgebung die Ergebnisse des komplexen Planungsprozesses vermittelt bekommen. Darüber hinaus wäre eine Darstellung auf mobilen Geräten möglich. Programmiert wird in Java EE und AngularJs.

1.3 Motivation für die Anwendung der EWS-API

Grundsätzlich wird davon ausgegangen, dass Termine im Outlook eingetragen werden. Dies sollte 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 EWS API die Daten an das Outlook zu senden bzw. an den Server zu übermitteln. Die EWS API erleichtert das Arbeiten mit Outlook enorm.

Bei relativ flexibler API, kann sowohl Java, C# als auch XML verwendet werden. Die Termine werden dabei in Outlook-Kalendern dargestellt. Änderungen von Terminen 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 die Handhabung der vorhandenen Funktionen bzw. Features beschrieben.

2.1 EWS API

2.1.1 Beschreibung

Die (Java) EWS API ist die einzige Schnittstelle, welche die Kommunikation mit einen Exchange-Server zulässt. Die API kann mit Java, C# und XML angesteuert werden und beinhaltet einen umfangreichen Funkionskatalog um die verschiedenen Funktionen im Outlook anzusteuern. Die EWS API wurde direkt von Microsoft entwickelt und daher ist kein *Konkurrenzprodukt* verfügbar.

Diese API wurde entwickelt um programmgesteuerte Abfragen und Aufgaben mit Hilfe eines Outlookservers zu realisieren. Diese Schnittstelle ermöglicht das Erstellen von Terminen, das Senden von Emails, das Abfragen von Verfügbarkeiten und das Senden von Benachrichtigungen.

2.1.2 Funktionen

HTTP(S)

Die EWS API benützt das HTTP Protokoll um Daten zu senden und zu empfangen. Diese Webschnittstelle ermöglicht eine einfache Handhabung von Daten und verwendet dabei JSON und XML zur Übermittlung. HTTPS wird verwendet um die Nachrichten und Anfragen sicher zu verarbeiten. HTTP hat den Vorteil, dass dieses Protokoll zustandslos ist und über einfache Befehle angesteuert werden kann.

Exchange Service

Der Exchange-Service ist der zentrale Service, welcher alle Aufgaben verarbeitet und in Verbindung mit dem HTTP(S) Protokoll den Exchange-Server ansteuert. Dieser Service wird jeder verwendeten Komponente mitgegeben und beinhaltet Zugangsdaten und die URL. Die URL wird verwendet um die Daten an den Server zu schicken. Am Server selber muss die EWS-Funktionalität freigegeben sein, da sonst die EWS API keinen Zugriff auf den Server hat.

Items und Folder

Alle Funktionen in der EWS API funktionieren nach dem Prinzip der Eindeutigkeit. Das bedeutet, dass jeder Ordner, jedes Mail und jeder Termin seine eigene ID besitzt, welche vom Server festgelegt wird. Über diese eindeutige ID wird auf jedes Element zugriffen und dieses wird dann angezeigt. Dabei gibt es keine Einschränkung da jedes Element gleichberechtigt ist. Weiters kann mit Hilfe dieser eindeutigen ID auf jedes gesonderte Element zugegriffen und dieses beliebig verändert werden. Alle Ordner haben eine eindeutige ID mit der auf den ganzen Ordner zugegriffen werden kann. Auch besitzen spezielle Ordner wie z.B. der Posteingang sogenannte *WellKnownFolderName*, welche wie Aliase für IDs fungieren.

Email

Emails können mit Hilfe der API gesendet und empfangen werden. Dabei besitzt jedes Email einen zugeordneten Ordner, welcher standardmäßig dem Posteingang darstellt. Die Emails werden automatisch versendet sobald sie am Server eingetroffen sind. Auch können Emails von beliebigen Konten abgefragt und verändert werden. Wie jedes Element besitzen alle Emails eine eindeutige ID.

Termine(*Appointments*)

Termine können relativ einfach erstellt werden und belegen die wichtigsten Funktionen bei der API, da diese auch der zentralen Punkt im Outlook sind. Es gibt zwei Arten von Terminen:
Die ersten sind die einmaligen Termine. Diese beinhalten nur ein Datum und werden nur einmal im Kalender angezeigt.
Die zweite Art beschreibt die wiederkommenden Termine. Diese haben neben dem Beginn und Ende auch eine Angabe an welchen Tag diese Termine sich wiederholen. Außerdem beinhaltet jeder Termin weitere relevante Daten, wie Raum, Ort , Teilnehmer und Beschreibung.

Eine weitere Funktionalität stellt die Abfrage von Verfügbarkeiten dar. Diese können relativ einfach abgefragt werden und liefern in weiterer Folge Vorschläge. Diese beinhalten neue Termine mit Zeit , Datum sowie den Ort. Außerdem wird auch die terminliche Anwesenheit sowie die räumliche Verfügbarkeit mitgeliefert. Ebenso wird die Qualität des Vorschlages mitgeliefert, welche aussagt wie gut der jeweilige Termin für die Teilnehmer geeignet ist.

Aufgaben(*Tasks*)

Ein weitere Funktionalität sind Tasks bzw. Aufgaben, welche erstellt und abgerufen werden. Tasks enthalten Aufgaben, welche getrennt vom Kalender aufscheinen und abgearbeitet werden. Außerdem sind Tasks keiner Person zugeordnet, sondern stellen lediglich eine Aufgabe dar.

Kontakte

Neben dem Terminen sind Kontakte im Outlook sehr bedeutend, da diese eine Hauptressource für Termine darstellen. Im Outlook existiert ein globales Adressbuch, welches die eigenen Firmenkontakte enthält. Diese Kontakte können einer Such-u.Auslesefunktion unterworfen werden, um dann Termine zugehörigen Personen zuzuordnen. Die Kontakte enthalten alle persönlichen Informationen sowie deren E-Mail-Adressen. Diese sind in Outlook von Bedeutung und dienen zur Identifikation von Personen, Ressourcen und Räumen

2.1.3 Vorteile

Grundsätzlich hat die EWS API den Vorteil dass die gleiche Funktionalität des Outlook-Programms auch programmiertechnisch in EWS verfügbar ist. Das erleichtert die Handhabung vom Outlook beträchtlich, da der Benutzer nicht mehr alles händisch eintragen muss. Die meisten Funktionen sind gut umgesetzt und dokumentiert.

2.1.4 Nachteile

Die EWS API hat noch immer einige Schwachstellen, welche noch nicht behoben sind bzw. es fehlen etliche Funktionen. Weiters sind manche Funktionen nicht korrekt dokumentiert, da die Dokumentation nicht upgedatet wird. Außerdem können einige Funktionen nicht genutzt werden, da diese auf eine eindeutige ID, die am Server generiert wird, aufbaut. Die programmatische Umsetzung ist extrem eingeschränkt, da die Funktionen nur bedingt eine brauchbare Schnittstelle bieten. Diese stellt sich zu kompliziert für eine mögliche Anwendung dar.

2.2 MRP - Multidimensionales Ressourcen Planning

2.2.1 Beschreibung

Das MRP System bzw. Tool dient zur Planung von Krankenhaus-Ressourcen. Diese werden *Appointments* und *PlanUnits* genannt. Diese zwei Ressourcentypen sind Endressourcen mit dem der Endbenutzer konfrontiert wird.

Die Hauptaufgabe von MRP ist es aus gegebenen Ressourcen und Regeln Planungskalender für einzelne Räume zu erstellen. Diese Kalender repräsentieren meistens OP-Planungen, da diese eine zentrale Einheit in einem Krankenhausinformationssystem (*KIS*) darstellen. Ein KIS besteht in erster Linie aus den Zentralkomponenten Organisation, Bettenmanagement, Aufnahme und Entlassung sowie OP-Planung.

MRP deckt nicht die zentralen Komponenten ab, da sich dieses System nur mit der Planung von Räumen und Terminen beschäftigt. Es übernimmt nicht die Verwaltung des gesamten Krankenhauses. Dies übernimmt meistens ein geeignetes KIS.

2.2.2 Funktionsweise

Das MRP System generiert aus vorgegebenen Daten geeignete Termine und Pläne für diverse Räume. Zu Beginn werden alle benötigten Ressourcen gesammelt und in einer zentralen Datenbank gespeichert und verwaltet, welche *Masterdata* enthält. Diese Masterdata können unabhängig abgerufen werden.

Aus den Masterdata-Datensätzen werden mit dem Ressourcenmanager die einzelnen Ressourcen zusammengesucht und sogenannte Appointments und PlanUnits generiert. Diese werden dann mit Hilfe des Kapazitätsmanagementtool den verfügbaren Kapazitäten der einzelne Ressourcen zugeordnet und in einer Kalenderansicht dargestellt.

Dieser Kalender beinhaltet alle wichtigen Informationen zu den einzelnen Ressourcen und zeigt auch deren zeitlichen Verfügbarkeit an. In dieser Ansicht können spezifische Informationen zu den einzelnen Ressourcen angezeigt und gegebenfalls geändert werden. Es können auch Appointments mit deren Elementen exportiert werden.

Am Ende finden sich alle Daten in einer Art Übersichtsmaske wieder. Alle PlanUnits mit den zugehörigen Terminen und Ressourcen werden aufgelistet und können auch detailliert angezeigt werden.

2.2.3 G3 HIS

G3 HIS ist eine zusammengesetzte Struktur von vielen Modulen, welche ein vollständiges KIS repräsentieren. Alle Grundfunktionalitäten sind in diesem System integriert und bilden eine einheitliche Weboberfläche. Diese ist clientunabhängig und kann auf jeden Gerät ausführt werden. Sie beinhaltet die gesamte Funktionalität des Systems und ist in beinahe jeden Browser lauffähig.

Das G3 HIS Projekt umfasst mehrere Projektgruppen, die in unterschiedlichen Aufgaben eines Krankenhaus unterteilt sind. Sie bieten eine umfangreiche Konfiguration und Verwaltung der einzelnen Funktionsgruppen bzw. Module.
Die derzeitige Software stellt die 3. Generation dar und ist als Programm noch nicht fertiggestellt. Jedoch sind die meisten Funktionalitäten voll funktionstüchtig. Das Programm aber ist derzeit noch in der *Demophase*. Es kann aber in näherer Zukunft ausgeliefert werden.

Kapitel 3

Design

In diesen Kapitel werden die möglichen Konzepte zur Übertragung von Daten ins Outlook beschrieben. Im Prinzip gibt es 3 verschiedene Konzepte zum gegenständlichen Thema:

3.1 Konzept 1: Manuelle Interaktion mit der Outlookoberfläche

Der User verwendet das Outlook und hat dabei die Möglichkeit folgende Funktionen zu verwenden:

3.1.1 Exchange Server

Der Microsoft Exchange Server ist eine Server-Software des Unternehmens Microsoft. Er dient der zentralen Ablage und Verwaltung von E-Mails, Terminen, Kontakten, Aufgaben. Der Server setzt eine Microsoft-Windows-Server-Software voraus und findet deshalb vor allem in von Microsoft-Produkten geprägten Infrastrukturen Verwendung. Um als Anwender die Funktionen von Exchange Server nutzen zu können, ist eine zusätzliche Client-Software nötig. Microsoft liefert dafür das separate Programm Microsoft Outlook.

3.1.2 Emails

Emails können gesendet, empfangen u. in diversen Ordnern hinterlegt werden. Neue Emails können Termine enthalten, welche als Termineinladungen gelten u. nach der Bestätigung in den Kalender übertragen werden. Weiters ist eine schnelle Abfrage der im persönlichen Ordner gespeicherten Emails durch eine ausgeklügelte Suchfunktion möglich.

3.1.3 Termine und Kalender

Jeder Benutzer verfügt mindestens über einen Kalender. Dieser Kalender besitzt alle persönlichen Termine sowie die firmenmäßige Umwelt. Weiters werden zusätzliche Termine wie Feiertage, Geburtstage etc. angezeigt. Eine Option ist die Übermittlung von Texten bei den Ein-/Ausladungen. Darüber hinaus können ganze Gruppen eingeladen werden. Eine Erinnerungsfunktion wird als eine weitere Servicefunktion angeboten.

3.1.4 Aufgaben

Aufgaben sind da, um Notizen zu machen, welche abgearbeitet werden müssen. Diese sind keiner Person zugeordnet und finden sich in einem separaten Punkt im Outlook wieder.

3.1.5 Kontakte / Adressen

Kontakte können erstellt werden, wenn eine Emailadresse bzw. eine Telefonnummer vorhanden ist. Kontakte sind in Gruppen unterteilt und können als Gruppe einen Termin zugewiesen werden. Eine Gruppe kann Kontakte enthalten sowie noch weitere Informationen der jeweiligen Person.

3.1.6 Globales Adressbuch

Alle Kontakte sind nach Gruppen geordnet u. eingeteilt. Diesbezüglich sind Such- bzw. Filterfunktion verfügbar.

3.1.7 Outlook Web App

Mithilfe des Outlook Web Access kann im Browser und den Zugangsdaten für das eigene Outlookkonto auf Emails, Kontakte und Aufgaben zugegriffen werden. Die EWS-API hat die gleiche Funktionalität wie die Webschnittstelle, da diese nach dem selben Prinzip funktioniert. Es gibt eine mobile Outlookversion, die mithilfe der Webschnittschnittstelle auf den Outlookserver bzw. Outlook zugreift.

3.2 Konzept 2: Ansteuerung von Outlook mit Hilfe von EWS

Der Benutzer verfügt über Programmierkenntnisse u. kann die Schnittstelle ansteuern und verwenden. Die EWS API wird von Microsoft zur Verfügung gestellt und dient zur Ansteuerung von Outlook über eine geeignete Programmiersprache. Diese API beinhaltet dieselbe Funktionalität wie

die Webschnittstelle u. arbeitet nach denselben Prinzip der Webschnittstellen. EWS bietet zusätzliche Funktionen zu der Webschnittstelle wie Abfragen von Verfügbarkeiten und Benachrichtigungen. Die EWS API ermöglicht den Programmierer eine einfachere Ansteuerung von Outlookservern. Die Schnittstelle bietet eine ausreichende Funktionalität, sodass im Outlookclient keine manuellen Aktionen durchgeführt werden müssen.

3.3 Konzept 3: Ansteuerung mittels MRP

MRP ist ein von CGM entwickelte Software, die optimale Termine für Operationen in einem Krankenhaus unter der Berücksichtigung aller beteiligter Ressourcen berechnet. Das MRP-Modul beinhaltet folgende Funktionalitäten:

1. Zu Beginn müssen die verfügbaren Ressourcen (Person, Raum, Material, Gerät, Bett) angelegt und zugeordnet werden.
2. Zuordnung der ressourcenbezogenen Daten: Jeder Ressource werden Informationen beigegeben
3. Aus den Ressourcen werden mithilfe von Kapazitäten Einheiten gebildet.
4. Aus diesen Einheiten werden Vorschläge gebildet. Diese Vorschläge beinhalten lediglich eine zeitliche Spanne. Um einen Termin festlegen zu können müssen Ressourcen zugeordnet 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.

Kapitel 4

Implementierung

In diesen Kapitel wird beschrieben wie mit Hilfe des MRP-Systems, die in der Zielsetzung aufgelisteten Punkte im Outlook implementiert wurden. Alle Unterpunkte beschreiben jeweils eine eigenständige Funktionalität, wobei es vorkommt, dass Klassen andere Klassen von Modulen verwenden können. Jede Unterkategorie besteht aus einer Beschreibung der Funktionalität und einer kurzen Beschreibung der Methoden innerhalb des Moduls. Dabei wurden die Funktionen der EWS-API verwendet und in den Methoden de nachfolgenden Modulen aufgerufen:

4.1 Outlook

Jedes der einzeln unten beschriebenen Packages bezieht sich auf die jeweilige Klasse:

- *Packagename + Service*
- *I + Packagename + Service*
- *Packagename + Dto*

Die jeweilige Klasse des Packages implementiert eine vordefinierte Schnittstelle (Interface), welche einen vordefinierten Benennungssystem folgt. Die unten genannten Funktionalitäten sind unabhängig vom MRP -System, können aber in diesem benutzt werden. Die verwendeten Datentypen bzw. Datenklassen sind im Package *Model* definiert. Diese Klassen können gleichnamige Klassen im MRP haben, aber haben keine Beziehung zu diesen Klassen.

4.1.1 Appointments

Erklärung

Mit Hilfe des Appointmentservices können Outlook-konforme Appointments angelegt, verwaltet , und abgerufen werden. Der *AppointmentService* implementiert das zugehörige Interface *IAppointmentService* und verwendet den

Datentyp *AppointmentDto*.

Methoden

Die wichtigste Funktion ist die Methode *createAppointment*, mit der Appointments angelegt werden. Der *ExchangeService* wird als Parameter übergeben. Die Methode *buildAppointment* baut den Datentyp *Appointment* zusammen. Die Methode *createAppointments* ermöglicht das gleichzeitige Speichern und Anlegen von mehreren Appointments.

Eine weitere Funktionalität ist durch die Methode *hasConflictedAppoinments* gegeben. Diese Funktion überprüft mit Hilfe des Exchangeservers etwaige Konflikte bei betreffenden Appointments.

Die Funktionen *appointmentListToXML*, *xmlToDto* und *xmlToAppointmentList* ermöglichen die Verwaltung von outlookkonformen Appointments mithilfe von XML.

4.1.2 Availability

Erklärung

Durch den Availabilityservice können Verfügbarkeiten von Personen und Räumen abgefragt werden. Der *AvailabilityService* implementiert das zugehörige Interface *IAvailabilityService* und verwendet den Datentyp *AvailabilityDto*.

Methoden

Die Hauptfunktion *checkAvailability* ruft mit Hilfe des *ExchangeService* Verfügbarkeiten über die EWS-Schnittstelle ab. Diese Funktion wird in fast jeder anderen Methode aufgerufen. Die Methode *checkAvailbilityForPerson* überprüft die Verfügbarkeit von einer oder mehreren Personen anhand deren Email-Adressen. Mit Hilfe der Methode *checkAvailbilityForRooms* können im Gegensatz zur vorherigen Methode statt Personen Räume abgefragt werden. Die beiden Funktonen *getAllRooms* und *getRoom* liefern einen oder mehrere Räume zurück. Jede Person sowie jeder Raum wird durch die jeweilige Email repräsentiert.

4.1.3 Kalender

Erklärung

Der Kalenderservice kann neue Kalender im eigenen Postfach anlegen und diese wieder abrufen. Der *CalendarService* implementiert das zugehörige Interface *ICalendarService* und verwendet den Datentyp *CalendarDto*.

Methoden

Die Mehode *getCalendars* ruft alle verfügbaren Kalender einer Emailadresse ab. Die Emailadresse repräsentiert eine Person oder einen Raum. Zusätzliche Kalender werden durch die Methode *createNewCalendar* realisiert. Diese Funktion liefert ein Ergebnis zurück, ob der Kalender angelegt werden konnte.

4.1.4 Kontakt

Erklärung

Der Kontaktservice ermöglicht das Anlegen, Abrufen und Verwalten von Outlookkontakten in den jeweiligen persönlichen Adressbüchern. Der Kontaktservice arbeitet standardmäßig mit dem Benutzer des Exchangeservices. Der *ContactService* implementiert das zugehörige Interface *IContactService* und verwendet den Datentyp *ContactDto*.

Methoden

Die Methode *getContacts* ruft alle persönlichen Kontakte des derzeit angemeldeten Benutzers aus seinen Globalen Adressbuch ab und zeigt diese in einer *pageSize*-großen Liste an. Diese Größe kann durch den jeweiligen Programmierer beliebig gewählt werden. Alle Kontakte sind standardmäßig unsortiert und können durch Filter zusätzlich aussortiert werden. Neue Kontakte werden über die Funktion *buildContact* zusammengebaut. Die Funktion *getContactDtos* liefert mithilfe der Funktion *buildContact* eine Liste vom vordefinierten Datentyp *ContactDto*.

4.1.5 Email

Erklärung

Der Emailservice ermöglicht das Senden von vordefinierten Emails. Diese beinhalten alle relevanten Felder, welche im Outlook zu finden sind. Weiters können alle Emails aus dem jeweiligen Postfach des Benutzers aufgerufen werden. Der *EmailService* implementiert das zugehörige Interface *IEmailService* und verwendet den Datentyp *EmailDto*.

Methoden

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 eigenen Posteingang ab.

4.1.6 Exchangeservice

Erklärung

Der Exchangeservice dient als zentrale Verbindung zum Outlook. Diese Vordefinierte Klasse wird bei fast jeder Methode der anderen Features als Parameter mitgegeben. Die Klasse wird *ExchangeServiceConnector* genannt und ist zuständig für die Verbindung zum Server und für die Anfragen von den anderen Features. Dieser Service bezieht sich nur auf das eigene Outlookkonto bzw. auf das Konto, welches mit Passwort und Benutzername angegeben wird. Weiters kann definiert werden, welche Exchangeserverversion verwendet werden muss. Diese Klasse implementiert das zugehörige Interface *IExchangeServiceConnector*.

Methoden

Die Serververbindung wird mit Hilfe der Methode *createServerConnection* aufgebaut, wobei der Benutzername und das Passwort mitgegeben werden. Innerhalb der Methode wird definiert, welche Art von Verbindung verwendet werden soll. Die gleichnamige Methode verwendet zusätzlich zu den beiden Parametern auch noch einen weiteren Parameter, welcher die Server-URL ist. Die XML konforme Methode *createServerConnection* kann mit Hilfe eines XML-Dokument eine Serververbindung aufbauen. Alle drei genanten Methoden retournieren den Exchangeservice, welcher dann für alle anderen Features verwendet wird. Mit der Methode *saveSettingsToXML* können die derzeitigen Servereinstellungen in einem XML-Dokument geschrieben werden.

4.1.7 Filter

Erklärung

Die Filter dienen dazu, dass beliebige Datentypen verglichen und somit gefiltert werden können. Diese Filter folgen dem Prinzip der überpersistenten Speicherung. Dies bedeutet, dass bereits gefilterte Daten trotzdem erhalten bleiben. Grundsätzlich ist jeder Filter aus zwei Teilen aufgebaut: Der erste Teil beinhaltet die relevanten Daten zu den jeweiligen Filter selbst. Der zweite Teil beinhaltet die zu filternden Daten, welche bis zum Ende gespeichert bleiben. Weiters beinhaltet jeder Filter sogenannte Filterkomponenten, welche selbst Filter sind, da der eigentliche Filter nur ein Container für solche Filter ist. Eine weitere Funktionalität ist der Vergleich von Methoden mit den Variablen. Dies bedeutet, dass der eigene Variablenname nicht bekannt sein muss.

Methoden

Die Mehode *apply* beschreibt die Hauptfunktionalität des Filters und beinhaltet alle Funktionalitäten, die zur Anwendung eines bestimmten Filters notwendig sind. Das *FilterDto* beinhaltet sowohl die Filter als auch die zu filternden Daten. Mit der Methode *applySimpleFilter* wird ein Filter auf eine Sammlung von zu filternden Objekten angewendet. Die Funktion *applyFilter* wird dann verwendet, wenn alle zu filternden Objekte den Typ *AppointmentComponent* besitzen. Die zwei Funktionen *applyHelixFilter* werden dann verwendet, wenn die Filter vom MRP-System stammen. Die Funktionen *getFiltered* und *getUnfiltered* dienen zum Abrufen von gefilterten und ungefilterten Daten eines *FilterDto*s

4.1.8 Model

Die verwendeten Objekte in 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. Dieses bedeutet, dass alle Membervariablen *private* sind und daher muss auf diese Variablen mit Getter und Setter zugegriffen werden. Die einzelnen definierten Datentypen besitzen einen öffentlichen Konstruktor ohne Parameter.

4.2 Masterdata

4.2.1 Beschreibung

Jedes Feature im MRP-System folgt immer der selben Struktur. Jedes Feature besteht aus den Teilen API, Component, Datenbank und Service. Dabei wird die Drei-Schichten-Architektur realisiert. Dieses bedeutet, dass es drei unterschiedliche Implementierungsweisen gibt, welche als getrennt angesehen werden können. 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 auf und leitet sie gegebenenfalls an die oberste Schicht weiter oder es werden spezifische Änderungen durchgeführt. Die oberste Schicht wird als Präsentationsschicht bezeichnet 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 bestehen.

4.2.2 Struktur

Jedes Feature folgt der Component-Service-Implementation, welche aus folgenden Teilen besteht:

Datenbank

Das Datenbankfeature verwaltet alle Daten in einer vordefinierten Datenbanktechnologie, die variabel auswählbar ist. Die Datenbank kann entweder SQL oder file-basiert sein. 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*. Dieses Objekt beinhaltet alle relevanten Datenbankzugriffsfunktionen und liefert ein DTO.

Service

Der Service bekommt 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 Bussinnes-Logik-Schicht an.

Components

Die Components stellen im Feature die Hauptbestandteile dar. Diese repräsentieren die eigentliche Logik, da in diesen Funktionen die gesamte Verarbeitung der Daten erfolgt. Die Components erhalten die Daten aus der Datenbank und wandeln diese Daten in *Data Transfer Objects* um. Diese 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 dann an das Frontend weitergeleitet werden. Außerdem ist die API zuständig für das Empfangen von Daten auf dem Frontend.

Frontend

Das Frontend repräsentiert die Präsentationsschicht, welche für den jeweiligen Benutzer sichtbar ist. Im Frontend werden alle Benutzerinteraktionen verwaltet und alle eigenen Daten zu Transferobjekte umge wandelt und an das Backend gesendet.

Kapitel 5

Evaluierung

5.1 Einleitung

Das Outlookconnectormodul ist im MRP-System teilintegriert. Das Modul befand sich im gleichen Repository, wurde aber nicht vollständig integriert, da dieses Modul in einem eigenen Zweig des Repositories verwaltet wurde. Dieser Umstand führte dazu, dass das Modul kein Teil der vordefinierten Testumgebung darstellte.

5.2 Testumgebung

Die Testumgebung wurde mit Hilfe eines *Jenkins-Maven-Testserver* realisiert, welcher selbständiges automatisiertes Testen zulässt. In dem Repositories gab es zwei Zweige, welche *master* und *dev* genannt werden. Nur diese zwei wurden in die automatisierte Testumgebung integriert und daher in fixierten Zeitabständen getestet. Da das zu bearbeitende Modul kein Teil der zwei Zweige war, konnte es nicht getestet werden. Durch diese Aufteilung wurden bereits fertiggestellte Module verspätet getestet. Das zu bearbeitende Modul wurde erst nach Beendigung des Praktikums in das Testframework integriert. Durch den Umstand des langsamen Testprozesses musste auf andere Testmittel umgestiegen werden.

5.3 Benutzerinteraktives Testen

Die gewählte Methodik wird als Benutzer interaktives Testen bezeichnet, besser bekannt als *Whitebox-Test*. Diese Methodik beruht auf der visuellen Kontrolle des Benutzers. Dabei überprüft der Programmierer die Ergebnisse in der grafischen Oberfläche bzw. Konsole. Diese Methodik kann nicht als Test im Sinne eines Testframeworks angesehen werden, da nicht überprüft werden kann, ob die Testergebnisse den erwarteten Ergebnissen entsprechen.

Dies kann erst mit der späteren Integration des Moduls überprüft werden. Diese Testmethodik dient rein zur visuellen Kontrolle ob die geforderten Datenfelder richtig in der Konsole angezeigt werden. Da explizite Tests fehlen, ist relativ schwer zu kontrollieren, ob diese durchgehen würden. Trotzdem können mit Hilfe dieser Methodik alle Funktionen eines Programms getestet werden, auch wenn kein geeignetes Testframework vorhanden ist. Da das Modul mit einen Server kommunizieren muss, ist automatisiertes Testen mit Hilfe eines Frameworks relativ kompliziert und damit unpraktikabel.

Folgende Funktionen wurden im Rahmen der Arbeit überprüft:

- E-Mails senden und empfangen
- Kontakte anlegen und löschen
- Termine anlegen, anzeigen und löschen
- Verfügbarkeiten abrufen
- Filter anlegen, bearbeiten und löschen

Alle obengenannten Funktionen wurden ausführlich getestet und ergaben schlussendlich keinen Fehler.

Kapitel 6

Zusammenfassung

In diesem Kapitel werden die Ergebnisse zusammengefasst und diskutiert und mögliche Verbesserungsvorschläge angezeigt.

6.1 Resultate

Bei dieser Arbeit wurde ein Programm implementiert , welches als Modul für das MRP-System (Version 3) der Firma CGM vorgesehen war. Dieses Modul beschäftigte sich mit der Ansteuerung eines Outlookservers mit Hilfe der EWS-API. Die Implementierung wurde mittels JAVA durchgeführt und daher wurde Java EWS API verwendet. Die EWS-API dient zur Ansteuerung eines Outlookservers mithilfe von Programmmethoden. Das Modul verwendete diese API um bereitgestellte Termine, welche Ressourcendaten aus dem MRP-System beinhalten können an das Outlook exportieren zu können. Diese resourcenbezogenen Daten wurden von anderen Modulen des MRP-System generiert und bereitgestellt. Allerdings fehlte die Integration des Moduls in das gesamtheitliche System des MRP, da dieses System zum derzeitigen Zeitpunkt noch nicht vollständig fertig gestellt war. Dabei wurde keine vollständige Testumgebung verwendet, da das Testen nur teilweise möglich war, da das Modul nicht in dem Testumfang integriert war. Das Testen erfolgte durch benutzerorientiertes Testen. Dieses beruhte auf eine rein visuelle Anschauung des jeweiligen Programmierers. Alle diesbezüglichen Tests lieferten am Ende der Arbeit ein positives Ergebnis.

6.2 Diskussion

Das in dieser Arbeit erstellte Programmmodul ist universell einsetzbar, wurde aber geringfügig an das MRP-System angepasst. Die komplexen Datentypen die die EWS verwendet, wurden durch das vereinfachte DTO ersetzt, Die universelle Einsetzbarkeit wurde damit noch zusätzlich verbessert. Weiters wurde die XML-Konformität implementiert. Dies bedeutet, dass XML-

Dokumente gelesen und geschrieben werden können. Die Implementierung der verwendeten Filter wurde generisch gehalten, um eine universelle Einsetzbarkeit zu ermöglichen.

6.3 Ausblick

Da das MRP-Modul noch nicht vollständig implementiert wurde , wird erwartet, dass dieses in den nächsten Jahren von der Firma CGM fertiggestellt wird und somit eine Integration des Outlook erfolgen kann. Das MRP-Modul ist ein Teil der *G3 HIS* Software, bei der bereits ältere Versionen im Einsatz sind. Dies macht die Software für Krankenhäuser interessant, welche ältere Versionen dieser Software verwenden oder ein anderes KIS besitzen.

Kapitel 7

Anhang

7.1 Literatur

JAVA EWS Reference Documument in GIT

CGM G3 HIS Documentation

G3 Reverse Proxy

CGM HTML5 Controls Browser