Outlook/Exchange Adapter für CGM G3 Clinical Information System MRP (Multidimensionales Ressourceplanning)

Krainer Philipp

BACHELORARBEIT

Nr. 1310458007-B

eingereicht am
Fachhochschul-Bachelorstudiengang

Medizin- und Bioinformatik

in Hagenberg

im Juli 2016

Diese Arbeit entstand im Rahmen des Gegenstands

Softwareentwicklung mit klassischen Sprachen

im

Sommersemester 2016

Betreuer:

Oliver Krauss

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. Juli 2016

Krainer Philipp

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Einführung

Diese Arbeit ist der Praktische Teil der Bachelorarbeit und wurde im Rahmen des Semesterpraktikums verfasst. Das Praktikum wurde bei der Firma CGM Clinical Austria am Standort Linz absolviert. Die Dauer des Praktikums betrug 16 Wochen. Im Rahmen des Projektes wurde eine Outlook konforme Schnittstelle entwickelt um die geforderte Zielsetzung zu erfüllen.

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

Grundsätzlich wird davon ausgegangen, dass Termine im Outlook eingetragen werden. Dies sollte automatisch nach der Generierung von vorgegeben Terminvorschlägen geschehen. Diese Terminvorschläge werden als Teil des MRP-Systems generiert. Meine 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

Zu Beginn ist anzumerken, dass sich dieses Kapitel ausschließlich mit vorhandenen Funktionen bzw. Features beschäftigt.

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 E-Mails 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

Projektbericht

3.1 Projekt

Die Aufgabe dieses Projektes war es die Termine des MRP Systems zu filtern und ausgewählte Termine in Outlook-Kalendern zu exportieren. Dies wird mit Hilfe der Java EWS API realisiert. Da das MRP System aus Frontend und Backend besteht war es notwendig beide *Seiten* zu programmieren.

Um die Anbindung an einen Outlookserver zu ermöglichen wurden im Backend Schnittstellen entwickelt, welche Termine an den Server senden und auch wieder abrufen können.

Im Backend sind zu den bereits vorhandenen Masterdata-Funktionen zwei weitere dazugekommen.

Die Erste ist die Filter-Funktion, welche es ermöglicht zusammengesetzte Filter zu erstellen und diese in einer Datenbank abzulegen. Diese Filter dienen zum Filtern von vordefinierten Appointments, welche in einer Datenbank gespeichert sind. Die gefilterten Appointments werden an das Frontend weitergeleitet und dort angezeigt.

Die zweite Funktion ist die Abfrage der gefilterten Appointments vom Frontend. Das Frontend sendet eine Liste von Appointments an das Backend. Diese Termine werden dort geprüft , in outlookkonforme Termine konvertiert und an den Server übermittelt. Diese können dann in den jeweiligen Kalendern eingesehen werden.

Das Backend kommuniziert mit den Frontend über Services und Komponenten und diese sind daher frei austauschbar. Im Backend gibt es weitere Funktionen, welche explizit getrennt von der Hauptfunktion implementiert worden sind. Diese Funktionen dienen hauptsächlich zum Lesen und Schrei-

ben von XML und JSON Dokumenten sowie für serverspezifische Abfragen. Auch im Frontend sind zu den bereits bestehenden Funktionen zwei weitere dazugekommen. Die Erste regelt das Anlegen und Abrufen von Filtern aus dem Backend anhand vorgegebener Filter, die von den gefilterten Daten abgerufen werden können. Es kann somit keine falschen Angaben geben. Die erstellten Filter werden im Backend in der Datenbank gespeichert, gesammelt, abgerufen und angezeigt.

Die zweite Funktion ruft vom Backend alle Filter ab und zeigt diese in einer Liste an. Mit diesen Filtern können die zugehörigen Appointments abgefragt und in einer weiteren Liste angezeigt werden. Die Liste enthält alle wichtigen Informationen von den gefilterten Appointments. Die gelisteten Appointments können selektiert und dann exportiert werden. Dabei wird der gesamte Datensatz an das Backend geschickt und dort weiter verarbeitet. Das Backend meldet dem Frontend etwaige Fehlermeldungen und schickt die Daten an den Outlookserver.

3.2 Arbeitsbericht

3.2.1 Beschreibung

Die Praktikumsarbeit besteht aus einer Backend-Komponente und aus einer Frontend-Komponente, welche zum Abfragen und zum Erstellen von Terminen geeignet ist. Das Backend ist in Java programmiert und das Frontend in Angular JS. Die Client-Server-Kommunikation baut auf dem Prinzip der Komponentenservices auf. Die Kommunikation wird über separat definierte Schnittstellen abgewickelt. Die Übertragung der Daten erfolgt mit Hilfe von JSON-Dateien. Die Abfrage der Appointments wird durch anpassbare wiederverwendbare Filter realisiert, welche in einer Datenbank hinterlegt werden können. Die Appointments werden direkt aus der Datenbank mithilfe der vordefinierten Filter abgerufen und die Ergebnisse werden angezeigt. Ausgewählte Appointments können exportiert werden und scheinen dann in den jeweiligen Outlook-Kalendern auf. Diese Appointments repräsentieren meistens die Öffnungszeiten von diversen Stationen, können aber auch spezifische Termine von Personen und/oder Verfügbarkeiten enthalten. Außerdem kann geprüft werden ob bei dem Export etwaige Konflikte auftreten. Diese können durch Vorschläge gelöst werden.

3.2.2 Bericht

Zu Beginn des Praktikums habe ich meinen Arbeitslaptop erhalten und eingerichtet. Weiters startete ich auch mit der Recherche über die Java EWS API. Als erstes habe ich mich mit der Einrichtung der Outlookkonten beschäftigt und auch ein kleines Testprogramm fertiggestellt.

Als nächstes habe ich einen XML und JSON Reader programmiert, der es mir ermöglichte beliebige Dateien einzulesen. Dann folgte der dazugehöriger Writer, welcher diese Dateien schreiben kann. In weiterer Folge implementierte ich einen DTO Konverter, welche die Daten in DTOs umwandelte.

Als nächstes folgte der Kontaktservice, welcher Daten zu Kontakten abfragen kann. Auch wurde das Projekt in das MRP GIT Repo integriert um ein gemeinsames Projekt zu schaffen. Nach der Integration programmierte ich einen Connector-Service, welcher eine Verbindung zum MRP System herstellte. Als nächstes habe ich ein generisches Filtersystem entwickelt, welches den Vergleich von verschieden Datentypen zulässt. Dieses System ermöglicht es die Appointments und Termine effizient zu filtern und abzurufen.

Ich habe dann begonnen zwei Masterdata-Features zu implementieren, welche eine Erstellung und Speicherung von Filtern zulässt. Weiters implementierte ich das Masterdata-Feature, welches Appointments mit Hilfe von den vordefinierten Filtern abzurufen und exportieren kann. Die jeweiligen Features beruhen auf den Client-Server-Prinzip, welches aus Backend und Frontend besteht.