HTL - Perg
Höhere Abteilung für Informatik

Diplomarbeit

Remserhof Guesthouse Management System

Projektteam: Georg Aschauer
Dominik Mühlbachler
David Sedlak

Projektbetreuer: Prof. Dipl.-Ing. Michael Romani

In Zusammenarbeit mit der Pension Remserhof

Betreuer: Herr Karl Mayr

Bearbeitungszeitraum: 01.10.2016 – 5.04.2017

Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Perg, _____ Unterschrift _____

(Name)

Perg, _____ Unterschrift _____

(Name)

Perg, _____ Unterschrift _____

(Name)

1 Inhaltsverzeichnis

1 Inhaltsverzeichnis 3

2 Danksagung 7

3 Abstract 8

4 Kurzbeschreibung 9

5 Einführung 10

5.1 Auftraggeber 10

5.2 Ausgangssituation 11

5.3 Problemstellung 12

5.4 Verbesserung durch RGHMS 13

6 Funktionen von RGHMS 14

6.1 Desktop-Applikation 14

6.2 Mobile Applikation 15

6.3 Die Website 15

7 Funktionalität aus Sicht des Nutzers 16

7.1 Desktop-Applikation 16

7.1.1 Übersicht 16
7.1.2 Menüleiste 16
7.1.3 Kassier 17
7.1.4 Blacklist 17
7.1.5 Frühstücke 18
7.1.6 Räume 18
7.1.7 Export 19

7.2 Mobile Applikation 20

7.2.1 Installation 20
7.2.2 Startseite 20
7.2.3 Navigation in der App 21
7.2.4 Neue Reservierungen 22
7.2.5 Übersicht: Frühstück 23
7.2.6 Übersicht: Reservationen 24
7.2.7 Übersicht: Gäste 25
7.2.8 Übersicht: Blacklist 26
7.2.9 Übersicht: Zimmer 27

7.3 Website 28

7.3.1 Navigation 28
7.3.2 Orientierung 28
7.3.3 Reservierung 29
7.3.4 Freie Plätze 29
7.3.5 Annullierung irrtümlicher Reservierungen 30

8 Technische Umsetzung 31

8.1 Desktop-Applikation 31

8.2 Mobile Applikation 31

8.3 Website 31

9 Grundlagen 32

9.1 Multi-Tier-Architektur 32

9.1.1 Kommunikation zwischen den Schichten 34

9.2 Backend 35

9.2.1 Microsoft Azure 35
9.2.2 Virtual Machine 35
9.2.3 VMware Workstation 35
9.2.4 Datenbank 35
9.2.5 Microsoft SQL Server 36
9.2.6 Datenmodell 36
9.2.7 Datenmodel von RGHMS 37
9.2.8 C# 38
9.2.9 ASP.NET 39
9.2.10 WCF 39
9.2.11 Webserver 39
9.2.12 IIS 39
9.2.13 Entity Framework 40
9.2.14 API 40
9.2.15 SOAP Service 41
9.2.16 SQL Server 2014 Management Studio 41

9.3 Desktop-Applikation 41

9.3.1 LINQ 41
9.3.2 WPF 41
9.3.3 XML 42
9.3.4 Visual Studio 42
9.3.5 .NET Framework 42

9.4 Mobile Applikation 43

9.4.1 Android 43
9.4.2 RESTlet 43
9.4.3 GraphView 43
9.4.4 Android Studio 43

9.5 Website 44

9.5.1 HTML 44
9.5.2 HTML5 44
9.5.3 JavaScript 44
9.5.4 JQuery 44
9.5.5 Ajax 45
9.5.6 Bootstrap 45
9.5.7 CMS 45
9.5.8 Notepad++ 45
9.5.9 Mobirise 45

9.5.10 HTTP 46

10 Vorgehensweise 47

10.1 Auswahl der Ressourcen/Technologien 47

10.1.1 Backend 47

10.1.2 Frontend 48

10.2 Implementierung des Systems 50

10.2.1 Backend 50

10.2.2 Mobile App 54

10.2.3 Desktop Applikation 55

10.2.4 Website 56

11 Organisation 57

11.1 Projektorganisation 57

11.1.1 Fortschrittanalyse 58

11.2 Backend 59

11.2.1 Meilensteinliste 59

11.2.2 Fortschrittanalyse 60

11.3 Die Desktop-Applikation 61

11.3.1 Meilensteinliste 61

11.3.2 Fortschrittanalyse 62

11.4 Mobile Applikation 63

11.4.1 Meilensteinliste 63

11.4.2 Fortschrittanalyse 64

11.5 Website 65

11.5.1 Meilensteinliste 65

11.5.2 Fortschrittanalyse 66

12 Probleme 67

12.1 Backend 67

12.1.1 Begrenzte Server Ressourcen 67

12.1.2 Datenbank funktioniert nach geändertem Password nicht mehr 67

12.2 Mobile App 68

12.2.1 Kommunikation mit dem Backend 68

12.2.2 Netzwerkkommunikation in Android 68

12.3 Website 68

12.3.1 Kaum Kommunikationsmöglichkeiten zu WCF 68

12.3.2 Cross Site Aufrufe 68

13 Resümee 69

14 Verfassungsnachweis 70

14.1 David Sedlak 70

14.2 Georg Aschauer 72

14.3 Dominik Mühlbachler 74

15 Textquellen _____ *76*

16 Abbildungsverzeichnis _____ *78*

17 Tabellenverzeichnis _____ *78*

2 Danksagung

Danken möchten wir allen Personen, die uns im Laufe der Diplomarbeit stets unterstützt und uns bei sämtlichen auftretenden Fragen und Problemen weitergeholfen haben. Besonderer Dank gilt unserem Betreuungslehrer Herrn Dipl.-Ing. Michael Romani. Darüber hinaus möchten wir uns bei unserem Auftraggeber Herrn Karl Mayr für seine Unterstützung bedanken.

Herzlichen Dank!

3 Abstract

Managing a guesthouse with pen and paper can be very difficult. Simple processes like searching a specific guest use up a lot of time, even if the system is well managed. The Remserhof-Guesthouse-Management-System or RGHMS for short displaces the pen and the paper with a new digital system. This system makes it much easier to keep an overview and to maintain the system. RGHMS provides a WPF-application and an Android-app for the staff to use this system. Furthermore, it includes a website, which simplifies the reservation process for the guests and the employees of the guesthouse.

4 Kurzbeschreibung

Eine Pension mittels Stift und Papier zu verwalten und zu dokumentieren ist nicht sonderlich einfach. Einfache Prozesse wie das Suchen nach einem bestimmten Kunden kann sich, trotz guter Organisation als sehr schwierig erweisen. Das Rermserhof-Guesthouse-Management-System oder kurz RGHMS, löst für die Pension Remserhof den Stift und das Papier durch ein digitales Verwaltungssystem ab. Das Verwaltungssystem ermöglicht zu jederzeit einen Überblick über die Zimmerverteilung in der Pension und erleichtert im Allgemeinen den täglichen Ablauf. RGHMS bietet dem Personal der Pension eine WPF-Anwendung und eine Android-App um dieses System zu nutzen. Darüber hinaus beinhaltet das System eine Website die den Reservierungsvorgang für den Gast beschleunigt und für Mitarbeiter des Remserhofs erleichtert.

5 Einführung

5.1 Auftraggeber

Auftraggeber der Diplomarbeit ist Karl Mayr, Leiter der Pension Remserhof in St. Valentin. Die Pension ist 1979 eröffnet worden und stellt insgesamt 30 Zimmer mit 55 Betten zu Verfügung. Die Zimmergröße variiert zwischen Einzel-, Doppel-, Dreibett- und Vierbettzimmer. Von den insgesamt 30 verfügbaren Zimmern sind 8 barrierefrei. Neben der Übernachtung ist es im Remserhof möglich, ein Frühstück zu bekommen.

Die Mehrheit der Kunden besteht aus Durchreisenden, die im Remserhof nur eine Tagesnächtigung möchten und ihre Zimmer vor der Ankunft reservieren.

5.2 Ausgangssituation

Die Verwaltung der Pension wurde mittels Zimmer- und Gästelisten in Form von Stift und Papier durchgeführt.

Zimmer konnten entweder per E-Mail oder mit einem Anruf reserviert werden. Anrufe werden von dem Personal entgegengenommen, das den Telefondienst zugeteilt bekommen hat. Die E-Mails wurden regelmäßig vom Unternehmensleiter aufgerufen und beantwortet.

Die Website der Pension gab den Besucher Information über die Pension, wie Anfahrt und Preise, und ermöglichte dem Besucher über ein Formular eine Reservierungsanfrage zu erstellen, welche als E-Mail versendet wurde.

5.3 Problemstellung

Die Menge an Dokumenten wie Gästelisten, Zimmerliste oder auch schriftlichen Reservierungen, war sehr hoch und die Suche von bestimmten Dokumenten konnte trotz gut strukturierter Ordnung, sehr viel Zeit beanspruchen.

Analysen wie die Zimmerauslastung über größere Zeiträume erwiesen sich ebenfalls als umständlich.

Wenn telefonische Reservierungen aufgenommen werden, muss das Personal, welches Telefondienst hat, sich nicht in der Pension befinden. Somit kann der Diensthabende die Daten der Reservierung nicht immer in die entsprechende Liste direkt eintragen. Wenn dieser Fall eintrat, mussten die Daten provisorisch notiert werden, bis sie in das eigentliche System eintragen werden konnten.

Reservierungen per E-Mail mussten extra in die entsprechende Liste übertragen werden.

Die Website war sowohl vom Design als auch vom technischen Aspekt veraltet.

5.4 Verbesserung durch RGHMS

RGHMS ist ein digitales Verwaltungssystem mit einer zentralen Speicherung. Sämtliche Daten können über das Internet abgerufen werden und somit wird die Nutzung von den 3 Komponenten, welche zur Verwendung des Systems angeboten werden, überall ermöglicht.

Eine Komponente ist die mobile Applikation. Diese gibt dem Nutzer jederzeit einen Überblick über die Zimmerbelegung und ermöglicht Reservierungen zu verwalten. Somit können auch telefonische Reservierungen direkt und standortunabhängig in das System eingegeben werden

Eine weitere Komponente ist die Desktop-Applikation. Mit dieser können alle Daten direkt und benutzerfreundlich verwaltet werden. Die Erstellung diverser Unterlagen in Druckform (Zimmerlisten, Reservierungslisten, ...) ist ebenfalls möglich.

Die letzte Komponente ist die Website. Diese hat ein modernes Design, das den benutzerfreundlichen Besuch auf jedem internetfähigen Gerät ermöglicht. Weiters bietet die Website eine überarbeitete Reservierungsfunktion, da Reservierungen jetzt direkt in das System eingetragen werden und nicht manuell übertragen werden müssen.

6 Funktionen von RGHMS

Das System speichert Daten zentral und bietet eine öffentliche Schnittstelle über die verschiedene Komponenten Daten konsumieren und ändern können.

6.1 Desktop-Applikation

Die Desktop-App bietet…

- eine Zimmerliste, die alle Zimmer anzeigt und die momentane Belegung falls vorhanden
  - mit CRUD-Operationen (Create-Read-Update-Delete)
  - mit Datums-Filterfunktionen
- eine Gästeliste, die alle Gäste anzeigt, die im Remserhof übernachtet haben
  - mit CRUD-Operationen
  - mit der Möglichkeit einen Gast zu blacklisten
- eine Blacklist, die alle Gäste anzeigt, die für weitere Übernachtungen im Remserhof gesperrt sind und die Möglichkeit
  - Personen zur Blacklist hinzuzufügen
  - Personen von der Blacklist wieder zu entfernen
- eine Frühstücksliste, die für einen bestimmten Tag alle Frühstücke anzeigt
  - mit CRUD Operationen
  - mit der Möglichkeit das Datum der Anzeige zu ändern
- eine Belegungsliste, die alle Belegungen und Reservierungen anzeigen kann
  - mit CRUD-Operationen
  - mit Datums-Filterfunktionen
- eine Exportfunktion, die die Zimmerliste für einen bestimmten Zeitraum als PDF exportiert

6.2 Mobile Applikation

Die Android-App bietet

- eine Zimmerliste, die belegte und freie Zimmer für einen bestimmten Tag anzeigt
  - mit der Möglichkeit das Datum der Anzeige zu ändern
- eine Gästeliste, die alle Gäste anzeigt, die im Remserhof übernachtet haben
  - mit einer Detailansicht mit allen Nächtigungen und Reservierungen des Gasts
- eine Blacklist, die alle Gäste anzeigt, die für weitere Übernachtungen im Remserhof gesperrt sind
  - mit einer Detailansicht mit allen Nächtigungen des Gasts
- eine Belegungsliste, die alle Belegungen und Reservierungen anzeigen kann
- eine Frühstücksliste, die für einen bestimmten Tag alle Frühstücke anzeigt
  - mit der Möglichkeit das Datum der Anzeige zu ändern
- eine Statistik, welche die Auslastung für den aktuellen Monat widerspiegelt
- eine Möglichkeit, ein neues Zimmer zu reservieren

6.3 Die Website

Die Website bietet…

- Informationen über die Pension, wie Standort oder Preise der Zimmer
- die Möglichkeit, Zimmer zu reservieren
- die Möglichkeit, sich den Weg zur Pension über Google Maps anzeigen zu lassen
- Information, welche Zimmer (Einzelzimmer, Doppelzimmer, etc.…) noch frei sind
- die Option, irrtümliche Reservierungen wieder zu entfernen

7 Funktionalität aus Sicht des Nutzers

7.1 Desktop-Applikation

7.1.1 Übersicht

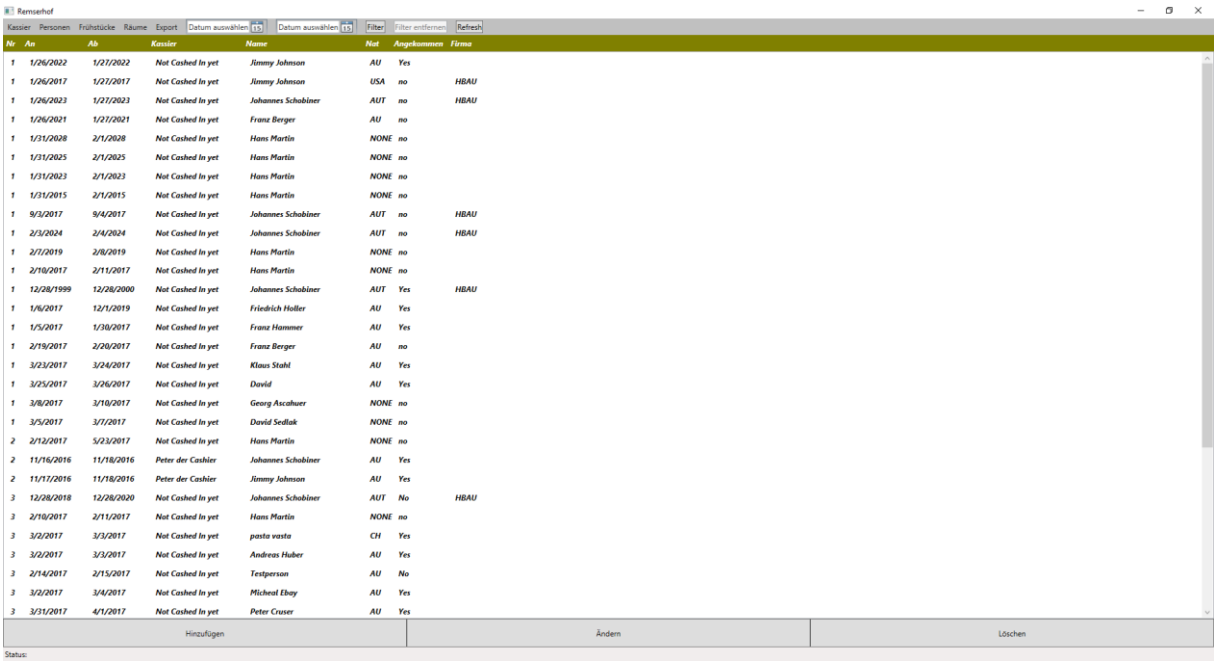

Abbildung 1 : Desktop-Applikation Übersicht

Die Übersicht zeigt alle Belegungen nach Zimmer sortiert. Mittels zwei Datepicker ist es möglich, die Einträge von einem gewissen Zeitraum zu filtern. Um einen Eintrag zu bearbeiten oder zu löschen muss man ihn nur anklicken und dann den jeweiligen Button drücken.

7.1.2 Menüleiste

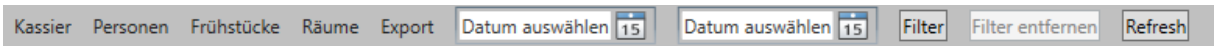

Abbildung 2 : Desktop-Applikation Menüleiste

Die Menüleiste ermöglicht die Navigation zu anderen Ansichten, einen Filter, eine Exportmöglichkeit sowie die Möglichkeit, die Ansicht zu aktualisieren.

7.1.3 Kassier

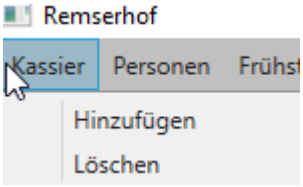

Abbildung 3 : Desktop-Applikation Kassier

Unter dem Menüpunkt „Kassier“ verbirgt sich die Möglichkeit, einen neuen Kassier anzulegen und bereits existierende wieder zu löschen. Die angelegten Kassiere können dann beim Anlegen oder Bearbeiten von Einträgen ausgewählt werden.

7.1.4 Blacklist

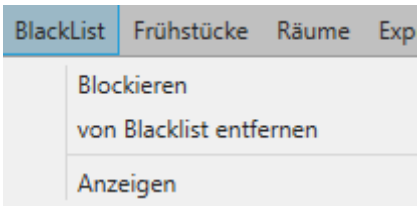

Abbildung 4 : Desktop-Applikation Blacklist

Unter dem Menüpunkt „Blacklist“ kann man die Blacklist verwalten. Mit einen Klick auf „Blocken“ kann man eine Person zu dieser hinzufügen und durch einen Klick auf „von-Blacklist-entfernen“ kann ihn auch wieder entfernen. Außerdem kann mit einem Klick auf Anzeigen eine Ansicht aufrufen die alle blockierten Personen anzeigt.

7.1.5 Frühstücke

Der Menüpunkt Frühstücke leitet auf eine eigene Ansicht für die Frühstücke weiter.

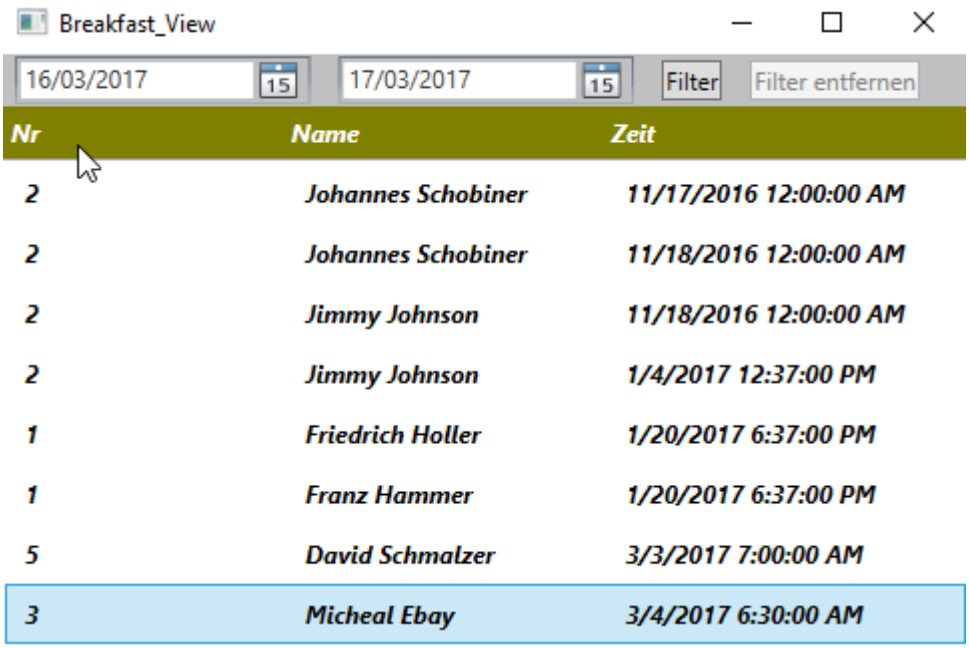

Abbildung 5 : Desktop-Applikation Früstücke

In dieser Ansicht sieht man alle Frühstücke nach Uhrzeit sortiert. Es ist wiederum möglich einen Filter mittels zwei Datepicker einzustellen.

7.1.6 Räume

Dieser Menüpunkt leitet auf eine Ansicht weiter in der angezeigt wird, ob in einer Zeit welche durch die zwei Datepicker angeben, die Zimmer frei sind oder wenn nicht, wer darin übernachtet.

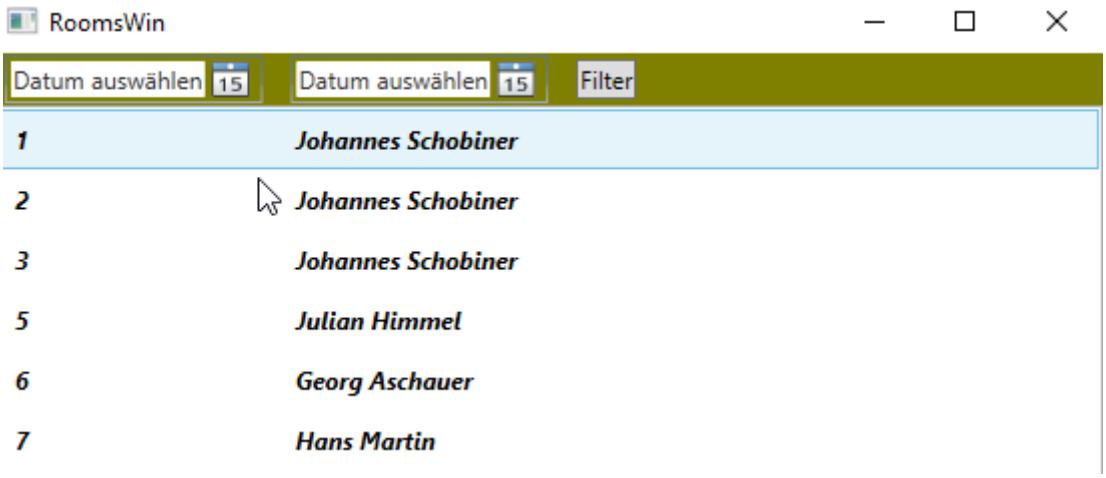

Abbildung 6 : Desktop-Applikation Räume

7.1.7 Export

Mittels des Exportpunktes im Menü ist es möglich die Einträge samt Filter in eine csv-Datei zu exportieren. Die csv-Datei kann später problemlos mit Excel bearbeitet oder ausgedruckt werden.

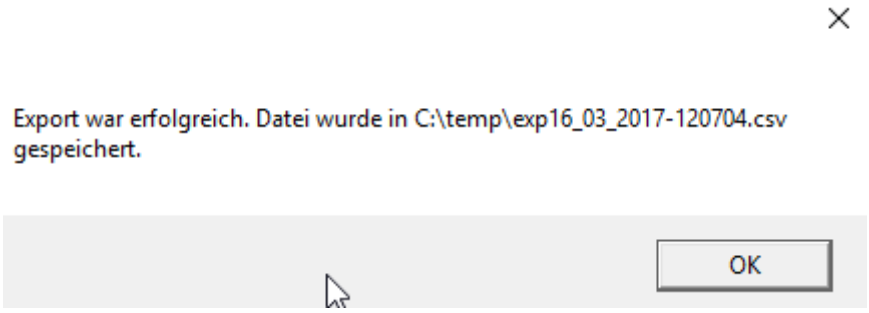

Abbildung 7 : Desktop-Applikation Exportmeldung

Nachdem man auf Export geklickt hat, bekommt man eine Meldung, die einem den Pfad der Datei und dessen Namen mitteilt.

	A	B	C	D	E	F
1	1	Jimmy Johnson	26/01/2022	27/01/2022	Not Cashed In yet	
2	1	Jimmy Johnson	26/01/2017	27/01/2017	Not Cashed In yet	
3	1	Johannes Schobiner	26/01/2023	27/01/2023	Not Cashed In yet	
4	1	Franz Berger	26/01/2021	27/01/2021	Not Cashed In yet	
5	1	Hans Martin	31/01/2028	01/02/2028	Not Cashed In yet	
6	1	Hans Martin	31/01/2025	01/02/2025	Not Cashed In yet	
7	1	Hans Martin	31/01/2023	01/02/2023	Not Cashed In yet	
8	1	Hans Martin	31/01/2015	01/02/2015	Not Cashed In yet	
9	1	Johannes Schobiner	03/09/2017	04/09/2017	Not Cashed In yet	
10	1	Johannes Schobiner	03/02/2024	04/02/2024	Not Cashed In yet	
11	1	Hans Martin	07/02/2019	08/02/2019	Not Cashed In yet	
12	1	Hans Martin	10/02/2017	11/02/2017	Not Cashed In yet	
13	1	Johannes Schobiner	28/12/1999	28/12/2000	Not Cashed In yet	
14	1	Friedrich Holler	06/01/2017	01/12/2019	Not Cashed In yet	
15	1	Franz Hammer	05/01/2017	30/01/2017	Not Cashed In yet	
16	1	Franz Berger	19/02/2017	20/02/2017	Not Cashed In yet	
17	1	Klaus Stahl	23/03/2017	24/03/2017	Not Cashed In yet	
18	1	David	25/03/2017	26/03/2017	Not Cashed In yet	
19	1	Georg Ascahuer	08/03/2017	10/03/2017	Not Cashed In yet	
20	1	David Sedlak	05/03/2017	07/03/2017	Not Cashed In yet	
21	2	Hans Martin	12/02/2017	23/05/2017	Not Cashed In yet	
22	2	Johannes Schobiner	16/11/2016	18/11/2016	Peter der Cashier	
23	2	Jimmy Johnson	17/11/2016	18/11/2016	Peter der Cashier	
24	3	Johannes Schobiner	28/12/2018	28/12/2020	Not Cashed In yet	
25	3	Hans Martin	10/02/2017	11/02/2017	Not Cashed In yet	
26	3	pasta vasta	02/03/2017	03/03/2017	Not Cashed In yet	
27	3	Andreas Huber	02/03/2017	03/03/2017	Not Cashed In yet	
28	3	Testperson	14/02/2017	15/02/2017	Not Cashed In yet	
29	3	Micheal Ebay	02/03/2017	04/03/2017	Not Cashed In yet	
30	3	Peter Cruser	31/03/2017	01/04/2017	Not Cashed In yet	
31	5	David Schmalzer	03/03/2017	03/03/2017	Not Cashed In yet	
32	5	Julian Himmel	06/01/2017	19/01/2017	Not Cashed In yet	
33	6	Georg Aschauer	05/03/2017	27/03/2017	Not Cashed In yet	

Abbildung 8 : Desktop-Applikation Export

7.2 Mobile Applikation

7.2.1 Installation

Die Applikation wird nicht über den Play-Store erhältlich sein, da nur ausgewählte Nutzer im Besitz der App sein sollen. Die App muss über den apk-Manager installiert werden und ist anschließend, wie jede App, im App-Manager zu finden.

7.2.2 Startseite

Im Startmenü wird die Verbindung zum Backend aufgebaut, was durch einen gelben Ladebalken visualisiert wird. Wenn die Verbindung hergestellt ist, wird auch eine Statistik geladen, die die Auslastung der Zimmer des aktuellen Monats repräsentiert. Ein „Neu-Verbinden“-Button stellt eine neue Verbindung zum Backend her und lädt alle Daten nochmals.

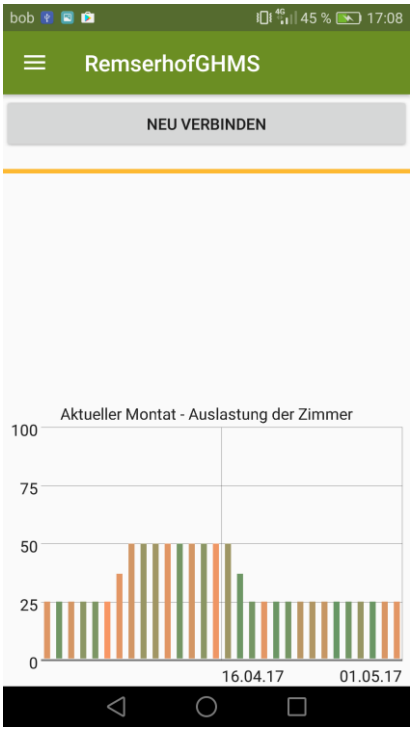

Abbildung 9 : mobile Applikation Startseite

7.2.3 Navigation in der App

Über die Menüleiste kann eine Navigationsleiste geöffnet werden, die dem Nutzer alle Übersichten öffnen lässt. Weiters wird es dem Nutzer über die Navigationsleiste ermöglicht, eine Ansicht zu öffnen, welche dem Nutzer eine neue Reservierung anlegen lässt. Diese Navigationsleiste ist nur über das Hauptmenü abrufbar. Sowohl durch eine Schaltfläche die mit „Zurück“ gekennzeichnet wird, als auch die Zurück-Taste die jedes Android-Gerät standartmäßig besitzt, ist es möglich, pro Klick eine Ebene zurückzugehen.

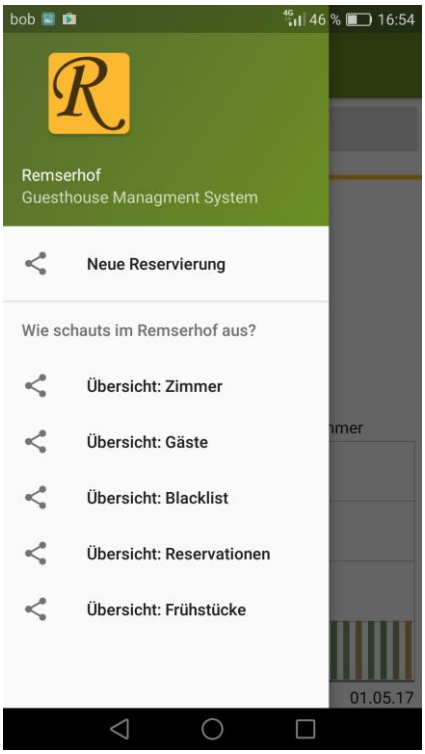

Abbildung 10 : mobile Applikation Navigation

7.2.4 Neue Reservierungen

Diese Ansicht bietet dem Nutzer die Möglichkeit eine neue Reservierung anzulegen. Um dies zu tun, muss zuerst ein Name eingegeben werden. Dieser bestimmt die Person, für die die Reservierung angelegt wird. Wenn der Name einem Namen entspricht, der bereits in der Datenbank vorhanden ist, so wird die Reservierung der Person mit dem entsprechenden Namen zugeordnet. Ist der Name nicht vorhanden so wird eine neue Person angelegt. Weiters ist es für die Reservierung nötig, ein Zimmer, das Anreisedatum und die Länge des Aufenthalts in Tagen anzugeben. Durch die Schaltfläche „Bestätigen“ wird die Reservierung bestätigt und falls alle Daten regelkonform eingegeben wurden, wird eine neue Reservierung angelegt.

Abbildung 11 : mobile Applikation neue Reservierung

7.2.5 Übersicht: Frühstück

Die Frühstücksübersicht gibt für den aktuellen Tag alle Frühstücke in einer nach Uhrzeit gruppierten Liste wieder. Dies bedeutet Frühstücke die für die gleiche Uhrzeit bestellt sind, werden zu einem Listenelement zusammengefasst. Dieses Listenelement gibt Auskunft über Anzahl und Uhrzeit der zusammengefassten Frühstücke. Durch einen *Long-Click* auf das Listenelement, eine Berührung die länger als eine Sekunde dauert, wird eine Detailansicht geöffnet die alle, zuvor gruppierten, Frühstücke des angeklickten Elements wiedergibt. In der Detailansicht werden Uhrzeit, Name und Zimmer der Frühstücksreservierung widergegeben. Über die Menüleiste ist es möglich das Datum nach dem gefiltert wird zu ändern.

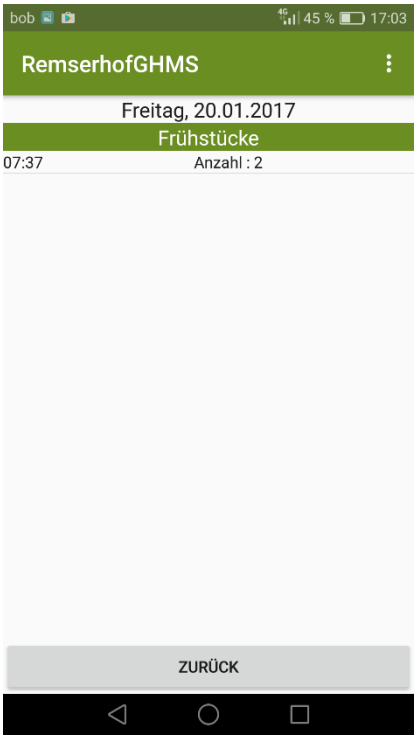

Abbildung 12 : mobile Applikation Frühstücke gruppiert

Abbildung 13 : mobile Applikation Frühstücke im Detail

7.2.6 Übersicht: Reservationen

Die Reservationenübersicht spiegelt alle Reservationen mit Details in Form von Name, Zimmer, Ankunft und Abreise der Reservation wider. Wie in der Frühstücksübersicht, besteht über die Menüleiste die Möglichkeit, das Datum nach dem die Reservierungen gefiltert werden, zu ändern.

Abbildung 14 : mobile Applikation Reservationen

7.2.7 Übersicht: Gäste

Die Gästeliste bietet eine Übersicht über alle Gäste, die im Remserhof übernachtet haben. Die Liste gibt Auskunft über Namen, Nationalität, falls vorhanden das Unternehmen das für die Übernachtung aufkommt, und eine Kontaktmöglichkeit in Form einer E-Mail-Adresse. Ein *Long-Click* auf einen Gast öffnet eine Detailansicht. Als erstes wird eine Überschrift angezeigt, die dem angeklickten Listenelement entspricht. Darunter werden alle Nächtigungen und Reservierungen des Gasts in einer Liste angezeigt. Diese beschreiben die Abreise, die Ankunft und das belegte Zimmer. Durch einen erneuten *Long-Click* auf das Element, das den Kunden beschreibt, öffnet sich die Reservierungsansicht. Die Person, für die reserviert wird, ist automatisch eingetragen und entspricht der zuvor angeklickten Person.

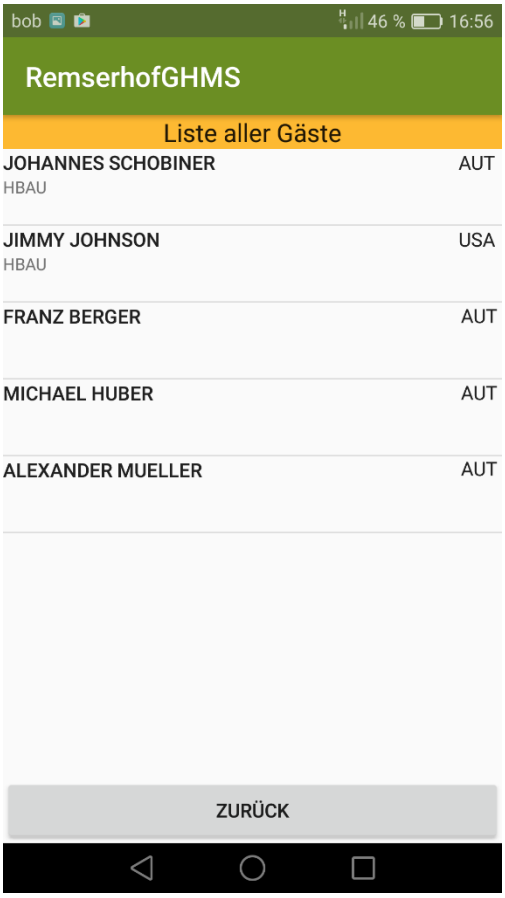

Abbildung 15 : mobile Applikation Gäste

7.2.8 Übersicht: Blacklist

Die Blacklist bietet einen Überblick aller Gäste des Unternehmens die für weitere Reservierungen bzw. Nächtigungen gesperrt sind. Graphisch entspricht diese Ansicht „Übersicht: Gäste“. Auch die Möglichkeit eine Detailansicht zu einem Gast aufzurufen besteht. Eine Reservierung über die Detailansicht aufzurufen ist nicht möglich.

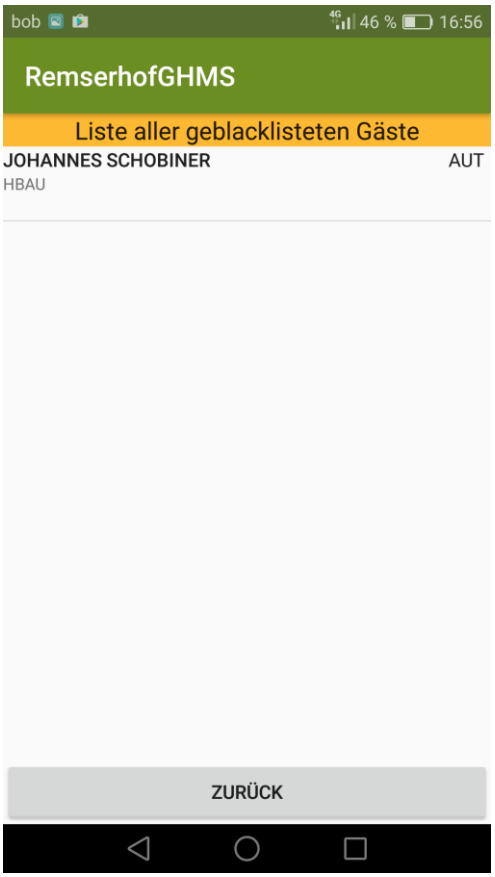

Abbildung 16 : mobile Applikation Blacklist

7.2.9 Übersicht: Zimmer

Die Zimmerübersicht lässt einen zwei Listen begutachten. Die untere Liste, die Liste der besetzten Zimmer, welche Zimmer anzeigt, die während dem eingestellte Datum, welches als Überschrift zu erkennen ist, nicht frei sind. Neben den Zimmernummern der besetzten Zimmer wird auch das jeweilige Abreisedatum angezeigt. In der oberen Liste sind alle Zimmer aufgelistet, die am gegebenen Datum zur Verfügung stehen. Ähnlich wie bei den Gästen in der Detailansicht ist es möglich über einen *Long-Click* eine Reservierungsansicht zu öffnen, die bereits das zuvor angeklickte Zimmer ausgewählt hat. Über die Menüleiste besteht die Möglichkeit das Datum, nach dem die Zimmer gefiltert werden, zu ändern.

Abbildung 17 : mobile Applikation Zimmer

7.3 Website

7.3.1 Navigation

Die Navigation auf der Website kann mithilfe von Buttons erfolgen oder auch nur durch das Scrollen, da die neue Website nur aus einer Seite besteht. Des Weiteren sind am unteren Ende eines Abschnittes meist kleine Schaltflächen, die den Benutzer einen Abschnitt weiterbefördern.

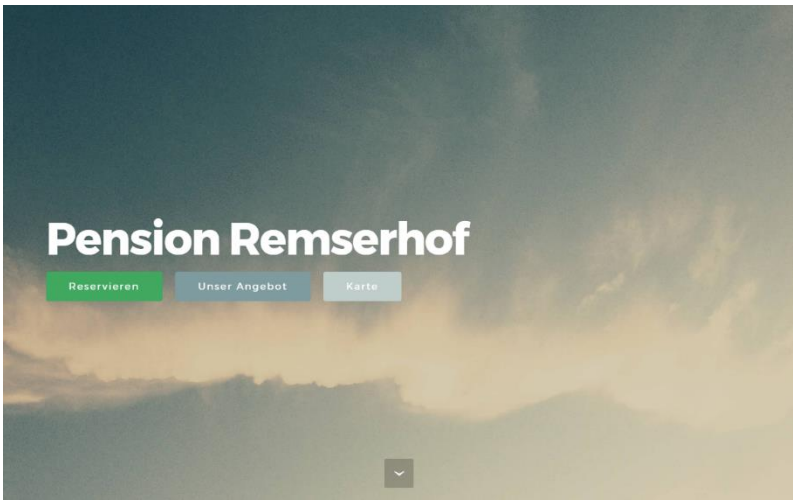

Abbildung 18 : Website Navigation

7.3.2 Orientierung

Der erste Abschnitt nach dem Titelbild der Website ist eine Karte von Google Maps welche die kürzeste Route vom aktuellen Standpunkt des Users zum Remserhof berechnet und anzeigt.

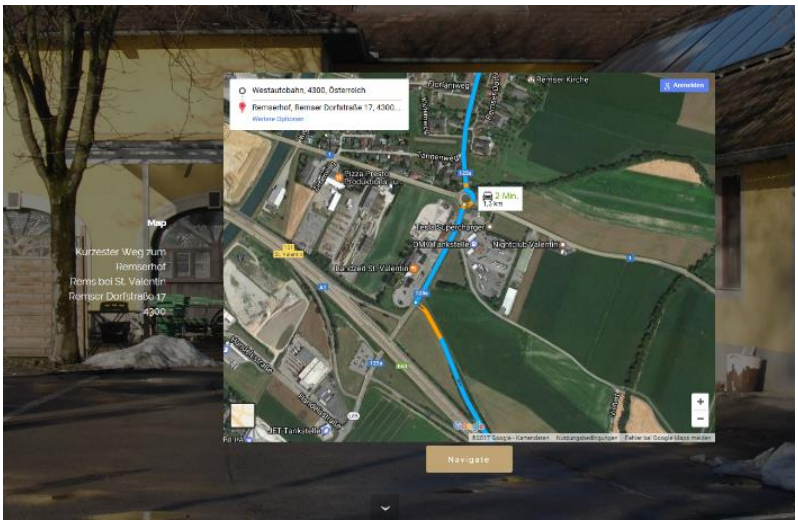

Abbildung 19 : Website Google Maps

7.3.3 Reservierung

Über ein Formular kann eine Reservierung angelegt werden. Für diese Reservierung müssen Daten wie Name, Ankunft- und Abreisedatum sowie Zimmergröße angegeben werden. Über die Schaltfläche „Reservieren“ kann diese Reservierung abgeschickt werden.

7.3.4 Freie Plätze

Neben dem Reservierungsformular gibt es eine Grafik, die angibt, ob ein Raum mit jeweiliger Bettenanzahl verfügbar ist. Beim ersten Aufrufen der Seite wird überprüft, ob in der Nacht von heute auf morgen Zimmer frei sind, und beim Eingeben des gewünschten Ab- und Anfahrtsdatum wird die Grafik auf der rechten Seite bzw. unter dem Formular aktualisiert, um die Verfügbarkeit der Räume im gewünschten Zeitraum widerspiegeln zu können.

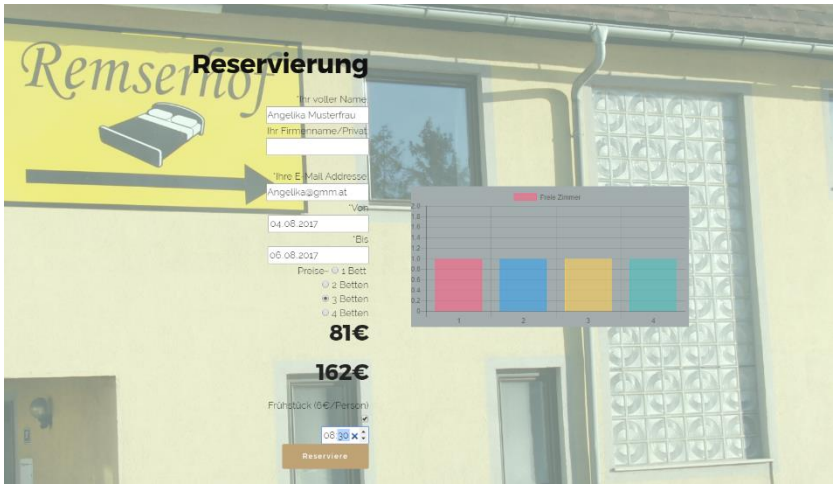

Abbildung 20 : Website Reservierung

7.3.5 Annullierung irrtümlicher Reservierungen

Neben etwaigen Statusmeldungen bei erfolgreicher Reservierung werden auch Felder generiert, welche die Reservierungen bei irrtümlicher Reservierung wieder mithilfe von den bereitgestellten Schaltflächen löschen können.

Abbildung 21 : Website Reservierungsbestätigung

8 Technische Umsetzung

RGHMS ist nach dem Konzept der Multi-Tier-Architektur konzipiert.

Das System realisiert eine Datenerhaltungsschicht mittels einer MS SQL Datenbank.

Die Anwendungsschicht wird von einer ASP.Net-Website, die über einen IIS ins Internet gehostet wird, gebildet. Die ASP.Net Website beinhaltet einen WCF-Dataservice, der als SOAP-Service, eine API bildet. Mit dieser API kommuniziert die Präsentationsschicht mit der Anwendungsschicht und konsumiert so die benötigten Daten.

Die Datenerhaltungsschicht und die Anwendungsschicht sind auf einem Windows 2012 Server implementiert. Dieser Server wird von MS Azure gemietet.

Die Präsentationsschicht besteht aus einer Android-Mobile-App, einer Desktop-App in Form eine WPF und einer Website.

8.1 Desktop-Applikation

Die Desktop Applikation wurden mittel der C# Technologie WPF realisiert. WPFs haben eine hohe Kompatibilität mit Microsoft Webservices, deswegen ist es ohne zusätzliche Bibliotheken möglich, eine Verbindung zu dem Backend zu schaffen.

8.2 Mobile Applikation

Die mobile Applikation wurde als Android-Applikation realisiert. Standardmäßig wird keine Funktionalität zu Verfügung gestellt um sich mit SOAP-APIs verbinden zu können. Die RESTLet-Bibliothek schafft hier Abhilfe. Darüber hinaus wurde RESTLet dazu verwendet um aus der API Objekte zu generieren, die in der Applikation verwendbar sind. Um eine Statistik in Android zu visualisieren, wurde die GraphView-Bibliothek eingebunden. Alle weiteren Funktionen wurden mit dem Android SDK umgesetzt.

8.3 Website

Die Website ist in HTML5, Javascript unter der Verwendung weniger Bibliotheken und, um die Kommunikation zum Backend zu ermöglichen, mit öffentlichen API-Schnittstellen im WCF Backend entwickelt worden. Ein Großteil der Funktionalität wird im Backend abgehandelt während die Website nahezu ausschließlich als Darstellungsebene fungiert.

9 Grundlagen

9.1 Multi-Tier-Architektur

Die Multi-Tier-Architektur ist eine Server-Client Architektur, die eine Anwendung in 3 Schichten zerlegt.

Die Datenerhaltungsschicht ist für die Verwaltung der Daten verantwortlich. Diese Schicht besteht meist aus einer Datenbank.

Die Anwendungsschicht liest die Daten direkt aus der Datenerhaltungsschicht und verarbeitet diese in der Anwendungslogik. Bei einer Web-Anwendung bildet ein Applikationsserver diese Schicht. Der Applikationsserver stellt eine API bereit, welche der Präsentationsschicht ermöglicht mit der Anwendungsschicht zu kommunizieren. Wird diese Anwendung im Internet gehostet, so wird auch ein Web-Server benötigt, welcher dieser Schicht zugeordnet wird.

Die Datenerhaltungsschicht bildet mit der Anwendungsschicht den Server.

Die Präsentationschicht bildet das User-Interface bzw. den Client der Anwendung. Die benötigten Daten werden aus der Anwendungsschicht konsumiert.

Die Datenerhaltungsschicht kann mit der Anwendungsschicht zum Backend zusammengefasst werden und die Präsentationsschicht bildet dabei das Frontend.

Durch die klare Trennung der Schichten entsteht eine hohe Flexibilität und Skalierbarkeit der Anwendung. So können verschiedene Applikationen entwickelt werden die mit den gleichen Daten arbeiten, solange sie ihre Daten vom selben Applikationsserver beziehen.

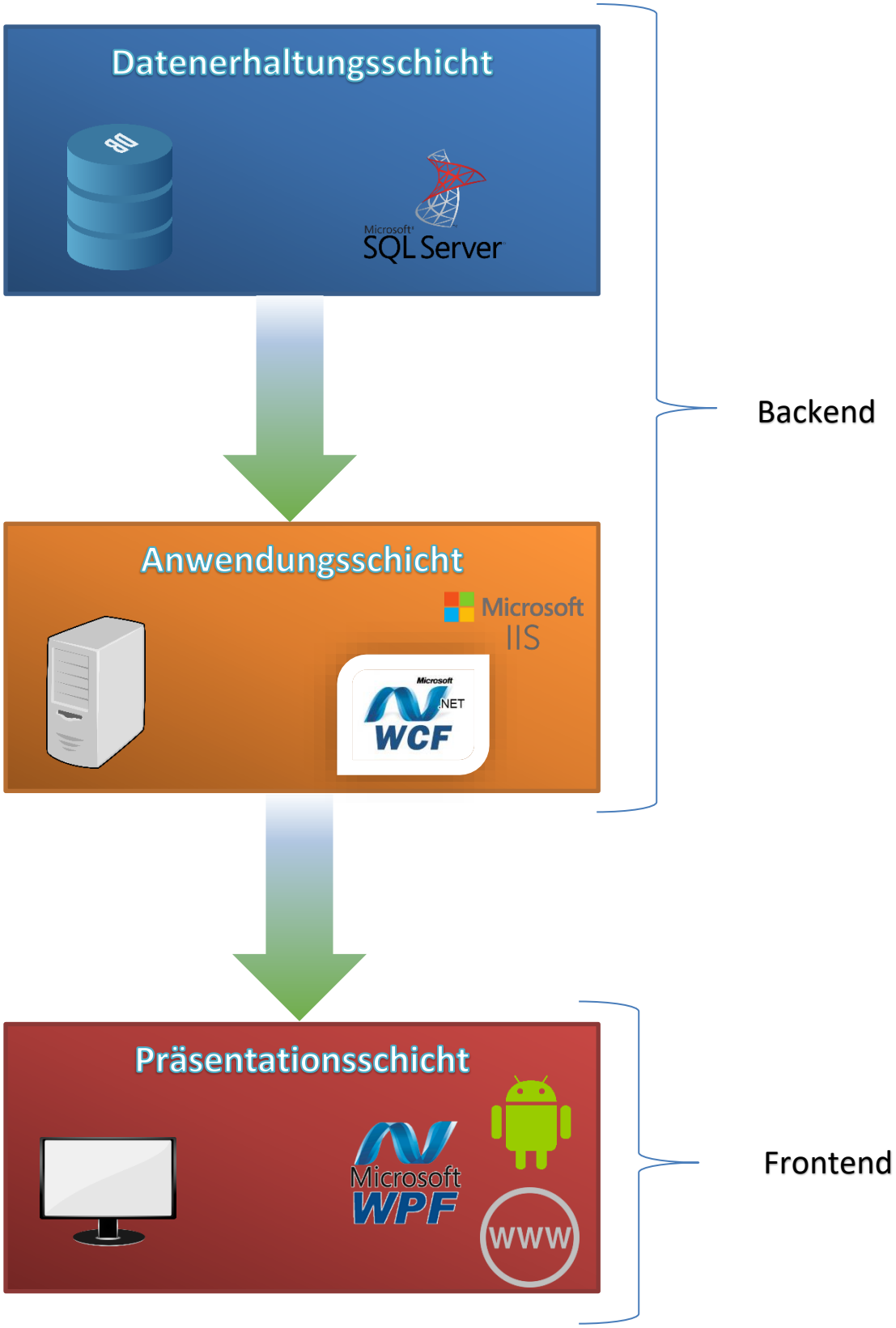

Abbildung 22 : Multi-Tier-Architektur

9.1.1 Kommunikation zwischen den Schichten

9.1.1.1 Datenerhaltungsschicht - Anwendungsschicht

Die Anwendungsschicht verbindet sich mit einen für die Datenbank angepassten Open-Data-Base-Connection-Treiber mit der Datenbank. Ein Object-Relational-Mapper erzeugt aus der Datenstruktur Objekte, welche anschließend mit den Daten aus Datenbank befüllt werden.

9.1.1.2 Anwendungsschicht – Präsentationschicht

Der Applikationsserver stellt eine API zur Verfügung. Diese wird von der Präsentationsschicht aufgerufen und zur Kommunikation verwendet. Über die API ist es möglich Daten zu versenden, aber auch Methoden in der Anwendungsschicht aufzurufen. Die Aufrufe, um Daten zu lesen oder ändern sind http-Anfragen, bei denen die eigentlichen Daten meistens in XML oder JSON codiert sind.

9.1.1.3 Präsentationschicht – Datenerhaltungsschicht

Die Präsentationsschicht kann nicht direkt mit der Datenerhaltungsschicht kommunizieren. Anfragen der Präsentationsschicht müssen zuerst von der Anwendungsschicht verarbeitet werden um zu gewährleisten, dass die Daten konsistent bleiben.

9.2 Backend

9.2.1 Microsoft Azure

Mircosoft Azure ist eine Cloud-Computing-Plattform welche seit 1. Februar 2010 verfügbar ist und verschieden Online-Dienste mit Fokus auf Softwareentwickler anbietet. Die Plattform nutz das Plattform-as-a-service-Konzept. Dies bedeutet, dass der Nutzer eine fertige Plattform mit entsprechender Infrastruktur als fertigen Service nutzen kann. Es bietet eine hohe Anzahl an Diensten, wie beispielsweise eine virtuelle Maschine mit vorinstalliertem Betriebssystem oder die Azure-Web-App, die mit einer Datenbank und sehr niedrigem Zeitaufwand einen Service bildet, von dem Daten konsumiert werden können.

9.2.2 Virtual Machine

Eine virtuelle Maschine ist ein Emulator eines physikalischen Computersystems. Sie ist auf der Architektur, eines echten Computer basiert und stellt auch die meisten Funktionen zur Verfügung.

9.2.3 VMware Workstation

VMware Workstation ist eine Software, die es ermöglicht, virtuelle Maschinen für Windows und Linux zu erstellen und zu konfigurieren.

9.2.4 Datenbank

Eine Datenbank ist eine Ansammlung von Daten.

Ein Datenbank Management System kurz DBMS ist ein Programm das sowohl mit Usern, anderen Applikationen als auch der Datenbank selbst interagieren. Das Hauptziel eines DBMS ist es die Erstellung und Wartung einer Datenbank zu erleichtern. Datenbanken werden meist in SQL oder relationale Datenbanken und NO-SQL Datenbanken unterteilt.

SQL-Datenbanken speichern die Daten in einer Sammlung von Tabellen. Diese Tabellen werden miteinander verknüpft um Beziehungen darzustellen.

NO-SQL-Datenbanken verfolgen im Gegensatz zu SQL-Datenbanken keinen relationalen Ansatz. Es gibt keine festgelegte Tabellenstruktur. Die Folge davon ist, dass Daten eventuell redundant, also mehrmals gespeichert werden, dafür sind Zugriffe um ein Vielfaches schneller.

9.2.5 Microsoft SQL Server

Der Microsoft SQL Server ist eine relationale Datenbanklösung von Microsoft. Es gibt viele verschiedene Ausführungen des Microsoft SQL Servers die sich in Preis und Funktionalität unterscheiden. Die meist verwendete Edition davon ist der SQL Server Express welcher zwar nur begrenzten Funktionen beinhaltet dafür aber ohne jegliche Kosten verwendbar ist. In RGHMS wird ein Mircrosoft SQL Server verwendet um die Daten zu speichern.

9.2.6 Datenmodell

Ein Datenmodell ist ein Modell welches einen Anwendungsbereich abbilden soll. Datenmodelle dienen dazu um die Struktur der Datenbank zu erstellen. Es gibt verschiedene Arten von Datenmodellen, die gebräuchlichste ist das ER-Diagramm, bei welchem die Gegenstände der realen Welt, auch Entitäten genannt, im Mittelpunkt stehen. Diese Entitäten werden dazu zueinander in Beziehung gesetzt, diese Verbindungen werden Relationships genannt.

9.2.7 Datenmodel von RGHMS

Die folgende Grafik stellt das Datenbankmodel vom Remserhof-Guesthouse-Management-System als ERD da.

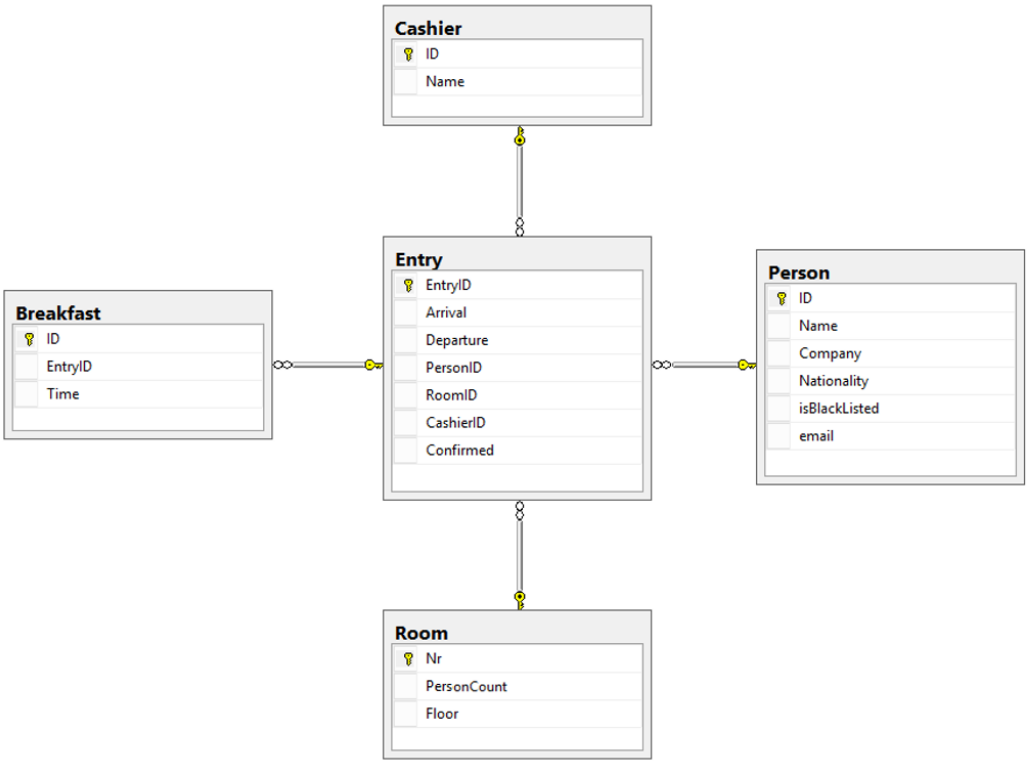

Abbildung 23 : Datenbankmodel

Tabelle	Attribut-Name	Beschreibung
Cashier	Name	Gibt den Namen des Kassiers an
Breakfast	EntryID	Gibt an, welcher Eintrag zu dem Frühstück gehört
	Time	Gibt an, wann das Frühstück geplant ist
Entry	Arrival	Gibt den Startzeitpunkt eines Aufenthaltes an
	Departure	Gibt den Endzeitpunkt eines Aufenthaltes an
	PersonID	Gibt die Person an die zum Eintrag gehört
	RoomID	Gibt an, welches Zimmer der Gast hat
	CashierID	Gibt an, wer das Geld für die Nächtigung kassiert hat
	Confirmed	Gibt an, ob der Gast bereits angekommen ist
Person	Name	Gibt den Namen der Person an
	Company	Gibt an, ob bzw. von welcher Firma der Gast ist
	Nationality	Gibt an, aus welchem Land der Gast kommt
	isBlacklisted	Gibt an, ob der Gast geblockt ist oder nicht
	Email	Gibt die E-Mail Adresse des Gastes an
Room	PersonCount	Gibt an, für wie viele Personen das Zimmer gedacht ist
	Floor	Gibt an, in welchem Stockwerk sich das Zimmer befindet

Tabelle 1 : Datenbankmodel

Jede Tabelle speichert abgesehen von ihren Attributen eine Identifikationsnummer (ID). Diese wird gebraucht um jede Eintrag eindeutig identifizieren zu können. In der Zimmertabelle bildet die Zimmernummer gleichzeitig die ID.

Die wichtigste Tabelle im System ist der Entry, also der Eintrag. Der Eintrag hat eine n:1 Beziehungen zu Kassier, Raum und Person, was bedeutet, dass ein Eintrag immer einen Kassier, Person und Raum haben, aber ein Raum, Person oder Kassier mehrere Einträge haben kann. Des Weiteren besteht eine 1: n Beziehung zu Frühstück, was bedeutet, dass ein Eintrag mehrere Frühstücke haben kann, wenn beispielsweise der Eintrag über mehrere Tage geht und der Gast des Eintrags jeden Tag ein Frühstück bekommt.

9.2.8 C#

C# ist eine objektorientierte Programmiersprache von Anderes Hejsberg, welcher diese für Microsoft entwickelte. C# wurde zwar ursprünglich fast nur für Windows entwickelt aber seit ASP.NET 5.0 wird auch Linux und MacOS unterstützt. Die neue Technologie von Microsoft die als Xamarin bezeichnet wird, ermöglicht außerdem Android, IOS, MacOS und UWP Apps in C# zu entwickeln.

Sowohl der Webservice von RGHMS als auch die Desktop Applikation werden C# programmiert.

9.2.9 ASP.NET

ASP. NET oder Active Server Pages .Net ist ein Framework von Microsoft für Web Applikationen. Es ist standardmäßig im .Net Framework enthalten. Des Weiteren gibt es noch ASP. NET Core, ein eigenständiges Framework mit ähnlichen Funktionen welches auch auf Linux und Mac OS verwendet werden kann.

9.2.10 WCF

Die Windows Communication Fundation ist eine Kommunikationsplattform von Microsoft. Sie stellt eine Sammlung von Netzwerkfunktionen für Applikationen zur Verfügung. Dabei werden viele Technologien zu einer einheitlichen Programmierschnittstelle zusammengefasst.

In RGHMS dient ein WCF als Schnittstelle zwischen Datenbank und Clients.

9.2.11 Webserver

Ein Webserver ist ein Server, der es ermöglicht Dateien an Clients, in den meisten Fällen einen Webbrowser, zu übertragen. Diese Dateien sind meist statische HTML-Dateien oder Bilder. Webserver ermöglichen aber auch dynamische Dokumente zu senden, welche erst beim Aufruf einer URI erzeugt werden. Diese dynamischen Dokumente werden meistens von Skriptsprachen erzeugt, können aber auch von Applikationen kommen, die von dem Webserver ausgeführt werden.

9.2.12 IIS

Der Internet Information Server ist ein Webserver von Microsoft, der in den meisten Windows Versionen beinhaltet ist. Neben den Standardfunktionen eines Webservers ermöglicht der IIS außerdem auch ASP.NET Applikation zu hosten.

RGHMS verwendet den IIS um sowohl den WCF als auch die Website zu hosten.

9.2.13 Entity Framework

Das Entity Framework ist Object Relationer Mapper in der ADO.NET Umgebung. Ein ORM erleichtert die Entwicklung von datenorientierten Software Applikationen. Dabei werden grundsätzlich zwei Vorgehensweisen unterstützt „Code First“ und „Database First“. Beim Code First Ansatz schreibt man, wie der der Name schon sagt, zuerst den Code mit den entsprechenden Klassen und lässt sich dann mit dem ORM die Datenbanktabellen erstellen. Bei Database First erstellt man zuerst ein Datenbankmodel und lässt sich mit dem ORM die entsprechenden Klassen der Applikation generieren.

RGHM verwendet das Entity Framework um die Entity Klassen der Datenbank für den WCF zu erzeugen.

9.2.14 API

Als API oder Application Programming Interface bezeichnet man einen Programmteil, welcher als Schnittstelle für andere Anwendungen dient.

9.2.15 SOAP Service

SOAP ist grundsätzlich ein Protokoll, welches den Datenaustausch zwischen zwei Systemen ermöglicht. Die Daten werden dabei in XML dargestellt. Die Übertragung erfolgt meistens über http/https und tcp/ip. Ein SOAP Service benutzt das SOAP Protokoll, um Daten zur Verfügung zu stellen. Die Request-URI bestimmt dabei, welche Daten als Response zurückgesendet werden. Ein SOAP Service stellt außerdem Metadaten zu Verfügung, welche einem zeigen, welche Daten man anfragen kann bzw. wie diese miteinander verknüpft sind. Die Metadaten ermöglichen auf diversen Frameworks wie z.B Restlet, .NET Framework das automatische Generieren von Klassen, welche die Daten repräsentieren.

RGHMS verwendet einen SOAP Service in Form der Technologie WCF von Microsoft.

9.2.16 SQL Server 2014 Management Studio

SQL Server 2014 Management Studio ist eine Softwareapplikation, die wie man bereits aus dem Titel entnehmen kann SQL Server, speziell den Microsoft SQL Server, verwalten lässt. Dies bedeutet, dass die Aufgaben eines DBMS übernommen werden.

Wir nutzen SQL Server 2014 Management Studio, um das zuvor entwickelte ERD-Modell im Microsoft SQL Server umsetzten.

9.3 Desktop-Applikation

9.3.1 LINQ

LINQ ist ein eigenes Verfahren von Microsoft, welches den Zugriff auf Daten definiert. Dieser Datenzugriff ist einheitlich innerhalb der .NET Umgebung, egal ob die Daten von Arrays, Collections, XML-Daten oder Quellen kommen.

9.3.2 WPF

Die Windows Presentation Foundation ist ein Grafikframework, welches Teil des .NET Framework ist, das seit Windows 7 standardmäßig mitgeliefert wird. Die Idee von WPF ist die strikte Trennung von Präsentation und Logik. Dabei wird in einer Xaml Datei das grafische Design definiert. Xaml ist eine Sprache, die mittels XML die graphische Darstellung in einer Baumstruktur deklariert. Die eigentliche Logik wird in einer cs-Datei designt. Jede Xaml-Datei besitzt eine dazugehörige cs-Datei, in der der Code-Behind definiert werden kann.

Die Desktop Applikation von RGHMS wird in Form einer WPF realisiert.

9.3.3 XML

XML oder Extensive Markup Language ist eine Auszeichnungssprache die es ermöglicht Daten, die in einer Baumstruktur vorliegen in einer Textdatei zu speichern. XML wird oft zum Austausch von Daten zwischen verschiedene System verwendet, vor allem über das Internet.

Der WCF von RGHMS stellt die Daten der Datenbank im XML-Format zur Verfügung.

9.3.4 Visual Studio

Visual Studio ist eine Entwicklungsumgebung von Microsoft. Visual Studio unterstützt die meisten gängigen Hochsprachen wie

- Visual Basic
- C
- C++
- C#
- TypeScript

Visual Studio stellt neben den Standardfunktionen von Entwicklungsumgebungen, wie das farbliche hervorheben von Schlüsselwörtern oder automatischer Syntaxprüfung auch graphische Interfaces zum Einbinden von Web-Services und diversen Bibliotheken. Eine weitere Funktion ist der eigene Server Explorer, welcher das Verwalten auf mehrere Datenquellen erheblich erleichtert. Sowohl die WPF als auch WCF sind in Visual Studio entwickelt worden.

9.3.5 .NET Framework

Das .NET Framework besteht aus vielen Klassenbibliotheken, APIs, Services als auch einer Laufzeitumgebung, welche die Programme ausführt. Es dient zur Entwicklung und Ausführung von .NET Programmen unter Windows. Das .NET Framework unterstützt viele Programmiersprachen da die .NET Programme beim Kompilieren in eine Zwischensprache namens „Common Intermediate Language“ übersetzt werden. Dieser Code wird dann beim Ausführen von der Laufzeitumgebung mit Hilfe eines „Just in Time Compiles“ in Maschinensprache übersetzt.

Das .NET Framework wird benötigt um die WPF Applikation als Endbenutzer zu verwenden.

9.4 Mobile Applikation

9.4.1 Android

Android ist ein Betriebssystem, welches speziell für mobile Geräte konzipiert ist. Das Betriebssystem wurde 2007 von Google veröffentlicht und ist seit 2013 das meist verbreitete Betriebssystem weltweit. Android ist unter der Open-Source-Lizenz Apache Lizenz 2.0 als Open-Source-Projekt verfügbar.

Android basiert auf einem modifizierten Linux-Kern der sich Monolith nennt. Der Kern selbst ist in der Programmiersprache C geschrieben, das Interface jedoch in Java. Applikation für Android werden wie das Interface in Java geschrieben.

Die neueste Android-Version, 7.1.1 – Nougat, ist seit dem 5. Dezember 2016 erhältlich.

9.4.2 RESTlet

Die RESTlet-Bibliothek ist ein open-source Framework, welches Funktionalität für Internetkommunikationen für APIs implementiert. Protokolle wie http, https aber auch Datenformate wie XML, JSON oder Atom werden unterstützt. RESTlet bietet Funktionalität sowohl für Client als auch für Serveranwendungen.

9.4.3 GraphView

Die GraphView ist eine exklusive für Android verfügbare Bibliothek, die Funktionalität zur einfachen Entwicklung dynamischer und flexibler Statistiken und Graphen bietet.

9.4.4 Android Studio

Android Studio ist eine IDE (integrated development environment), die speziell für Android entwickelt wurde. IDE bedeutet, dass es sich um eine Entwicklungsumgebung handelt, die Funktionen wie die Möglichkeit, Source Code zu bearbeiten, Debugger und Compiler inkludiert.

Android Studios ist seit Dezember 2014 unter der Apache Lizenz 2.0 gratis erhältlich.

Android Studio wurde verwendet um die Android App zu entwickeln.

9.5 Website

9.5.1 HTML

Hypertext Markup Language

HTML ist die Standardsprache für das Erstellen von Websites. Sie ist eine Auszeichnungssprache um digitale Dokumente mit Hyperlinks, Medien, Designs und weiteren Elementen zu erweitern und darzustellen.

Die visuelle Darstellung wird durch den Webbrowser und Vorlagen wie z.B. CSS bestimmt. HTML ist aktuell in der Version 5.1

9.5.2 HTML5

HTML5 führt verschiedene strukturierende Elemente ein, von welchen sich einige auch mit der Überschriftenhierarchie vereinen lassen. Des Weiteren wurden Elemente zur spezifischen Einbindung von Video- und Audiodateien, zur Textauszeichnung und zur Interaktion hinzugefügt. Mehrere Elemente wurden um Parameter erweitert, wie z.B. im Falle des input-Elements: die Eingabe von Suchbegriffen, Telefonnummern, E-Mail und Datums- wie Zeitangaben. Auch wurden einige Elemente, die zur Darstellung dienten, in ihrer Bedeutung geändert.

9.5.3 JavaScript

Die Skriptsprache für dynamisches HTML dient zum Verändern von Inhalten, Nachladen und Generieren von weiterem HTML und CSS und der Auswertung von Benutzereingaben.

Des Weiteren wird es ermöglicht, dynamisches HTML in Browsern zu verwenden. JavaScript wird den objektorientierten Programmierparadigmen in jedem Fall gerecht, wobei sich auch funktional oder prozedural programmieren lässt.

9.5.4 JQuery

Die freie JavaScript Bibliothek stellt Funktionen zur Manipulation von HTML-Dokumenten zur Verfügung. Es ist mit deutlichem Abstand die meistverwendete JavaScript Bibliothek.

9.5.5 Ajax

„Asynchronous JavaScript and XML“ ist die Bezeichnung des Konzeptes, welches Datenübertragung zu einem Server erlaubt. Es ist möglich über HTTP anzufragen ohne die angezeigte Seite neu zu laden, und währenddessen auch die Inhalte besagter HTML-Seite zu verändern

9.5.6 Bootstrap

Bootstrap ist ein frei zu verwendendes CSS-Framework, welches auf CSS und HTML basierende Gestaltungsmöglichkeiten beinhaltet. Mit starker Verbreitung bringt es mit nur wenigen Angaben in Klassennamen große visuelle Designfortschritte und Einheitlichkeit in ein Projekt.

9.5.7 CMS

Ein Content Management System erlaubt das gemeinsame Bearbeiten und Organisieren von Inhalten in Webseiten und wird häufig auch zum Erstellen von Websites verwendet. Es wird das Updaten von Inhalten über ein Interface ermöglicht und es verwendet meistens eine Datenbank. Bekannte Open Source Anbieter sind: Wordpress, Drupal, Joomla, TYPO3.

9.5.8 Notepad++

Notepad++ ist ein Texteditor, welcher auf mehreren Betriebssystemen funktioniert. Es ist möglich, ihn mit Myriaden von Plugins zu erweitern und Features wie Syntaxhervorhebung sind bereits in der Standardauslieferung integriert.

9.5.9 Mobirise

Die Website https://mobirise.com/de/ bezeichnet ihr Tool als Offline-Website-Baukasten für die Betriebssysteme Windows und Mac. Es eignet sich im Besonderen für kleine bis mittlere Webseiten mit wenig Wartungsbedarf und ist mit kostenlosen Standardthemes verwendbar.

Das Tool zeichnet sich durch Minimalistik und Responsivenes, eine angepasste Anzeige für alle Geräte, in seinen Endprodukten aus.

9.5.10 HTTP

Das Hypertext Transfer Protocol dient zur Übertragung von Daten auf der Anwendungsschicht. Die hauptsächliche Übertragungsmethode bei der kommunikation mit der Website ist die Get-Methode, welche ihre Attribute ausschließlich über die Adresszeile überträgt

10 Vorgehensweise

10.1 Auswahl der Ressourcen/Technologien

10.1.1 Backend

10.1.1.1 Server

Grundsätzlich gibt es zwei Möglichkeiten. Entweder man beschafft sich einen eigenen Server oder man mieten einen. Für das RGHMS kommt nur einen Server mieten in Frage, da es in der Pension nicht möglich, ist einen Server zu positionieren, auf Grund von zu geringer Bandbreite. Weitere Vorteile von einem gemieteten Server sind, dass man den Server nicht selber warten muss, viel geringere Initialkosten, Unabhängigkeit von Strom und Internetausfällen.

Als Nächstes stellt sich die Frage welchen Anbieter man wählen soll. Grundsätzlich gibt es eine Menge von Anbietern, doch nach längerer Überlegung kamen die Anbieter 1&1 und MS Azure in die engere Auswahl. Die Angebote der beiden Anbieter waren fast identisch. Beide bieten 1,75 GB Ram, ein CPU Core mit 1,8GHz und 50GB SSD mit Windows Server 2012R2 als Betriebssystem um rund 20 Euro pro Monat. Diverse Vorteile von Microsoft Azure Cloud, wie das nur tatsächlich genutzten Ressourcen bezahlten werden müssen oder das bereits sehr gut ausgereift Interface zur Verwaltung der zur Verfügung stehende Ressourcen machen MS Azure zum besseren Anbieter.

10.1.1.2 Die Anwendungsschicht

Um einen Webservice zu entwickeln gibt es 2 „best-practice“-Methoden. Eine Methode wäre, einen Java JAX-WS-Webservice zu entwickeln, die andere einen WCF-DataService. Java-WS-Webservice zeichnet sich dadurch aus, dass er mit weniger Ressourcen auskommt, doch RGHMS lastet die Ressourcen, die dem System zu Verfügung stehen nicht aus, somit spielt dieser Faktor eine untergeordnete Rolle. Ein Vorteil des WCFs ist, dass er mit Microsoft Produkten sehr hohe Kompatibilität bietet und einige Funktionen implementiert, die beispielsweise Fehlerbehebung einfacher machen als bei einen Java-Web-Serivce. In diesen Fall überwiegen die Vorteile eines WCFs und um von der hohen Kompatibilität des .Net-Frameworks zu profitieren wurden auch im Backend Microsoft Produkte verwendet. Der Server verwendet das Betriebssystem Windows Server 2012 R2, die Datenbank ist ein Microsoft SQL Server 2014 Express und der Webserver ist ein Microsoft IIS.

10.1.2 Frontend

10.1.2.1 Warum 3 verschiedene Applikationen anstatt Xamarin oder Cordova?

Die plattformübergreifende Technologie Xamarin von Microsoft ermöglicht es nur eine Applikation zu programmieren und diese dann in Android, Mac OS, Windows 8 und UWP bereitzustellen.

Apache Cordova ist ein weiteres Framework das es ermöglicht eine App für mehrere Plattformen auf einmal zu entwickeln. Dabei wird der Programmcode in HTML5, CSS3 und JavaScript geschrieben. In Cordova ist es wie auch in Xamarin nicht möglich, auf alle Funktionen der einzelnen Plattformen zugreifen. Auch das eine starke Verzögerungen bei neuen Betriebssystem entsteht, ist ein Nachteil, den sich Cordova mit Xamarin teilt.

Weiters ist es nicht möglich Anwendungen, mit komplexen User Interfaces zu bauen, da diese auf jeder Plattform anders sind. Ein weiterer Punkt ist, dass es nahezu unmöglich ist, für die neueste Version der einzelnen Plattformen, vor allem Android und Mac OS Apps zu entwickeln, da Xamarin und Cordova deren Features zuerst unterstützen muss und dann noch ein entsprechendes Update für diese Unterstützung veröffentlichen müssen. Außerdem besteht keine Möglichkeit, eine Website mit Xamarin zu entwickeln.

Des Weiteren sind die Ziele bzw. Funktionen der einzelnen Anwendungen unterschiedlich. Während die mobile App hauptsächlich dazu verwendet, wird Daten übersichtlich für den Benutzer anzuzeigen, wird die Desktop Applikationen dafür verwendet Einträge hinzuzufügen oder zu ändern. Es wäre zwar möglich beides in eine plattformübergreifende Technologie zu vereinenden jedoch würde damit das Userinterface und somit auch die Benutzerfreundlichkeit darunter leiden.

10.1.2.2 Warum Android?

Nach dem die Entscheidung gegen Xamarin und Cordova gefallen ist, musste ein Betriebssystem für die mobile Anwendung gewählt werden. Diese Entscheidung basiert auf die Kaufgewohnheit des Kunden, der exklusiv Android-Geräte verwendet.

10.1.2.3 Warum eine WPF?

Weil das Backend einen SOAP-Service in Form eines WCF Services zur Verfügung stellt, ist die beste Technologie für eine Benutzeranwendung mit GUI schnell gefunden. Eine WPF bietet die nötige Kompatibilität um sich von den Services, die Entity Klassen erzeugen zu lassen und eine einfache Art und Weise, das graphische Interface zu gestalten. Die einzige Voraussetzung um eine WPF Applikation öffnen, ist eine Installation des .NET Framework, welches standardmäßig mit Windows mit installiert ist.

10.1.2.4 Warum kein CMS?

Nach längerer Überlegung wurde sich, trotz der Einfachheit die bei der Erstellung einer Website geboten wird, gegen ein Content Management System entschieden. Der zusätzliche Funktionsumfang wurde nicht benötigt und die meist vorhandene Notwendigkeit einer weiteren Datenbank wäre eine Verschwendung von benötigten Ressourcen.

10.2 Implementierung des Systems

10.2.1 Backend

10.2.1.1 Microsoft Azure Server mieten

Um einen MS-Azure-Server zu mieten, wird zuerst ein gültiger Microsoft-Account benötigt. Dieser braucht zusätzliche Daten wie Kreditkarteninformationen, um später Zahlungen abwickeln zu können. Wenn dieser Account eingerichtet ist, so kann man im Azure-Portal, welches unter der Adresse https://portal.azure.com/ zu finden ist, eine neue Ressource auswählen. In Fall von RGHMS ist diese Ressource eine VM mit dem Betriebssystem Windows Server R2 Datacenter. Bei der Auswahl muss die VM-Kategorie bestimmt werden, welche die Leistung und den Preis der VM angibt. Für RGHMS wurde das Model „A1“ gemietet.

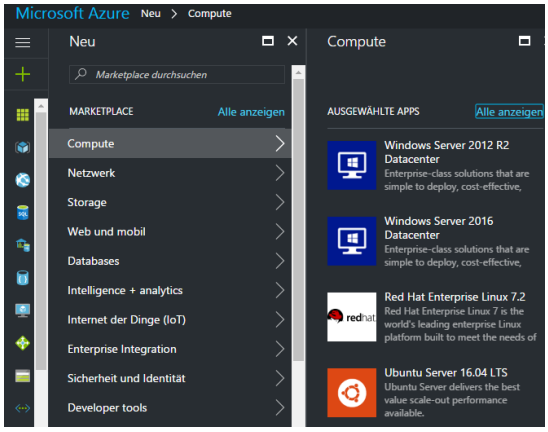

Abbildung 24 : MS Azure Ressourcenauswahl

INSTANCE	CORES	RAM	DISK SIZES [1]	PRICE
A0	1	0.75 GiB	20 GB	~€11.30/mo
A1	1	1.75 GiB	40 GB	~€22.59/mo
A2	2	3.50 GiB	60 GB	~€75.92/mo
A3	4	7.00 GiB	120 GB	~€203.29/mo
A4	8	14.00 GiB	240 GB	~€406.57/mo

Abbildung 25: MS Azure Serverklassen

10.2.1.2 Remote Verbindung zum Azure Server aufbauen

Die Remote Verbindung zum Server funktioniert durch Remote Desktop Protokoll kurz RDP welches in Windows integriert ist. Um eine Verbindung zum Server aufzubauen, muss nur eine „Remotedesktopverbindung“ öffnen und die IP des Zielservers eingeben.

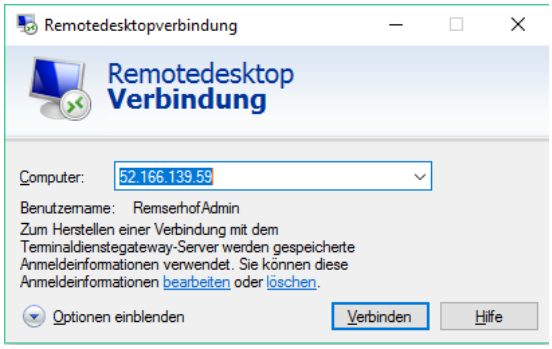

Abbildung 26 : Remoteverbindung

Nachdem man auf „Verbinden“ drückt kann man sich normal wie beim Starten von Windows mit Benutzername und Passwort anmelden.

10.2.1.3 Installation des Webservers

Um den Webserver IIS auf einen Windows Server 2012 zu installieren, öffnet man zuerst den Server Manager. Unter „Add Roles and Features“ findet man eine ziemlich große Auswahl mit Funktion die man hinzufügen kann, unteranderem auch der IIS.

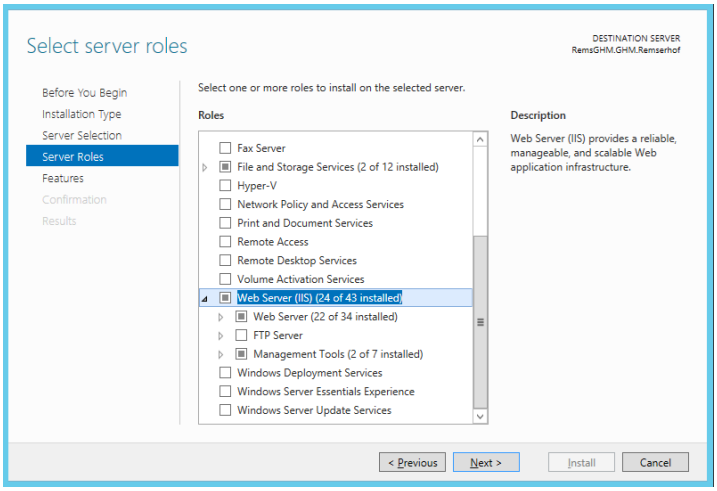

Abbildung 27 : Installation des Webservers

10.2.1.4 Installation des Microsoft SQL Servers Express

Die Installationsdatei des SQL Servers Express kann man auf der Website von Microsoft kostenlos herunterladen. Beim Ausführen wählt man „eine neue Instanz erstellen“. Nachdem man mit dem Erstellen der Instanz fertig ist, hat man eine funktionierte Datenbank auf dem System. Um die Verwaltung der Datenbank zu vereinfachen, ist es jedoch noch nötig, die Microsoft SQL Server Management Studio Datei von derselben Website herunterzuladen und zu installieren.

Mit dem Microsoft SQL Server Management Studio kann man sich dann problemlos auf die Datenbank verbinden und die Tabellen, wie im Datenbankmodell beschrieben, erstellen.

10.2.1.5 Erstellung des Web Services

Um den einen SOAP Service in .NET zu erstellen, erstelle man zunächst in Visual Studio ein neue "Web Project". Zu diesem Web Project fügt man mittels Rechtsklick "Add" ein eine "WCF DataService“ Datei hinzu. Sobald dies erledigt ist, fügt man ein Datenmodel hinzu bei welchen man "Create From Database" wählt. Dann wählt man die Datenbank des Remserhofs aus, und die Entity Klassen werden erstellt. Dann müssen einige Änderung in der Service Datei vorgenommen werden. Der Typ des „EntityFrameworkDataService“ muss auf den Namen der Entitys geändert werden in diesem Fall auf „remserhofEntities“. Die Zugriffoperationen müssen auf „*“ geändert werden um auch das hinzufügen, ändern und löschen von Daten zu erlauben.

```
15
16  namespace RemserWCF
17  {
18      [ServiceBehavior(IncludeExceptionDetailInFaults = true)]
        0 references
19      public class WcfDataService : EntityFrameworkDataService <remserhofEntities>
20      {
21          // This method is called only once to initialize service-wide policies.
            0 references
22          public static void InitializeService(DataServiceConfiguration config)
23          {
24              // TODO: set rules to indicate which entity sets and service operations are visible, updatable, etc.
25              // Examples:
26              config.SetEntitySetAccessRule("*", EntitySetRights.All);
27              config.SetServiceOperationAccessRule("*", ServiceOperationRights.All);
28              config.DataServiceBehavior.MaxProtocolVersion = DataServiceProtocolVersion.V3;
29              config.UseVerboseErrors = true;
30          }
31      }
32  }
33
```

Abbildung 28 : Code des WCF-Dataservices

Durch „UseVerboseErrors = true“ stellt man außerdem noch ein, dass bei Fehlern genaue Exceptiones angezeigt werden was das Testen vom einiges erleichtert.

Der WCF-Service kann nun mittels Rechtsklick auf das Projekt veröffentlich werden. Dabei ein neues Profil erstellen, bei diesem die Website auf dem IIS wählen auf welche der WCF veröffentlicht werden soll. Ab jetzt ist der Webservice über den IIS erreichbar. Jedoch wenn man sich eine, einzelne Entity ansehen möchte, bekommt man immer noch einen Fehler. Daraus kann man schnell schließen, dass noch einen User mit den Namen „IIS“APPPOLL\REMSERHOFWCF“ in der Datenbank anlegen und diesem Lese- und Schreibrechte geben muss. Um dies zu tun geht man in SQL Server Management Studios wählt unter „Security“, „Logins“. Mit einem Rechtsklick auf „Logins“ kann man nun „Add Login“ wählen. Nun gibt man einfach den Namen ein und wählt unter „User Mapping“ die Datenbank, in diesen Fall „Remserhof“, und anschließend wählt man die Rechte „dbreader“ und „dbwriter“ auswählen. Nun kann man auch die einzelnen Entitys ohne Fehler aufrufen. Ein Problem steht allerdings immer noch aus und zwar der Webservice ist nicht von außerhalb des Servers erreichbar. Um das zu ändern, muss unter den Firewall-Einstellungen eine „Inbound-Rule“ Erstellen, bei welcher man den Port der Website, auf welcher der WCF veröffentlicht ist, einträgt.

10.2.2 Mobile App

10.2.2.1 Verbindung zum Backend

Nach dem Anlegen der App, wird ein neuer Thread in Form eines AsyncTask gebraucht. Dieser wird für die Verbindung zum Backend benötigt, da Android eine Netzwerkverbindung im Haupt-Thread nicht zulässt. Die Funktionalität des AsyncTasks wird mit der RESETlet-Bibliothek implementiert. Weiters ist es über die Bibliothek möglich Klassen zu generieren, welche der Struktur der Daten entspricht, die vom Backend übergeben werden. Die konkrete Vorgehensweise ist in der Dokumentation von RESTlet, auf ihrer Website https://restlet.com/, in detaillierter Ausführung zu finden.

10.2.2.2 Graphische Oberflächen

In Android werden sogenannte Layouts entwickelt, die den Code für die Oberfläche in XML speichern. Die Entwicklung wird in Android Studios mit einem Designer erleichtert. Diese Layouts können dann über Objekte aufgerufen werden und werden dem entsprechend angezeigt.

10.2.2.3 Kommunikation zwischen Ansichten

Neue Ansichten werden mit einen *Intent* aufgerufen. Diesem *Intent* kann man Daten übergeben werden, die von der neu geladenen Ansicht genutzt werden können. Diese Daten werden allerdings *Call-By-Value* übergeben. Dies bedeutet die übergebenen Daten werden kopiert und Änderungen wirken sich nicht auf andere Ansichten auf. Um diesen Problem entgegen zu wirken, können Daten auch im Applikationskontext gespeichert werden. Dieser Kontext ist über jede Ansicht abrufbar und Änderungen wirken sich auf alle Daten aus.

10.2.2.4 Anzeigen von Daten

Die von RESTlet generierten Klasse speichern alle Werte, diese sind jedoch nicht zur Anzeige geeignet. Um solch eine Klasse für die korrekte Anzeige anzupassen, wird eine Adapterklasse benötigt, welche die generierten Klassen zu graphischen Elementen adaptiert.

10.2.3 Desktop Applikation

Ein WPF-Projekt lässt sich in Visual Studio recht einfach erstellen. Man muss nur ein neues Projekt anlegen und unter „Windows/Classsic Desktop“, eine WPF-Applikation auswählen.

Um nun mit dieser WPF auf die Daten welche über den WCF zur Verfügung gestellt werden, zugreifen zu können, muss man eine Referenz hinzufügen und die Entitiy Klassen erstellen. Dies mach man mit einem Rechtsklick auf Projekt bei dem man „Add Service-Reference“ auswählt. Nun muss man nur noch die Adresse des WCFs eintragen und Visual Studio erzeugt automatisch die Entity Klassen.

Jetzt kann man sich mittels folgendem Code alle Einträge in eine Collection laden.

```
remserhofEntities entities = new remserhofEntities(uri);
var Query = from o in Entities.Entries orderby o.RoomID select o;
DataServiceCollection<Entry> entries= new DataServiceCollection<Entry>(Query);
```

Abbildung 29 : Konsum von Daten eines Webservices

Um einen neuen Eintrag hinzuzufügen muss man diesen anlegen, die benötigten Attribute setzten und ihn zur Collection hinzufügen. Mittel „SaveChanges“ wird das Ganze an den WCF weitergeleitet.

```
Entry en=new Entry();
en.Arrival = (DateTime)DatePicker_Arrival.SelectedDate;
en.Departure = (DateTime)DatePicker_Departure.SelectedDate;
en.Confirmed = ComboBox_Arrived.SelectedItem as String;
entities.AddToEntries(en);
entities.SaveChanges();
```

Abbildung 30 : Übergabe von Daten an den Webservice

Löschen und Ändern von Einträgen funktioniert nach demselben Prinzip mit den Methoden „UpdateObject“ und „DeleteFromEntries“.

10.2.4 Website

10.2.4.1 Aufsetzen auf dem Azure Server

Zuallererst muss der Webseitencontent mit allen Inhalten auf den Server migriert werden. Dort wird der Internet Information Server geöffnet und eine neue Seite angelegt. In dessen Managementtool wird eine neue Seite angelegt, und diese auf den Port 80 geschalten. Das Verzeichnis der Website sollte nun auf das Verzeichnis welches die index.html beinhaltet zeigen.

10.2.4.2 Darstellung

Um eine Webseite zu erstellen und sie darzustellen benötigt man nur einen Texteditor und einen Browser. Für die Einbindung verschiedenster CSS Frameworks und Bibliotheken zur optischen Verbesserung wurde ein visueller Designer verwendet, in dessen Exportprodukt die Funktionalität und die eigenen Hintergründe und Bilder wortwörtlich eingesetzt wurde.

10.2.4.3 WCF

Zur Implementierung von gewissen Funktionen, musste der WCF-Service um einige handgeschriebenen Methoden erweitert werden. Diese werden mit Annotationen markiert welche Beschreiben über welche URIs und mit welchen Methoden sie Aufrufbar sind.

```
[WebInvoke]
[WebGet]
0 references
public int[] getRooms()
{
    remserhofEntities ent = new remserhofEntities();
    Room[] entRoom = ent.Rooms.ToArray();
```

Abbildung 31 : manuell konfigurierte Methode des Backends

Das Manipulieren und Auslesen der Daten erfolgt über das Entity Framework auf welches man im Service direkten Zugriff hat. Die Methoden zur Bearbeitung dieser sind die exakt Selben wie auch oben in dem Bereich der Desktop Applikation beschrieben.

Darüber hinaus müssen diese Methoden mit dem von WCF zur Verfügung gestellten Zusatzparameter $format=json aufgerufen werden, um die Daten in JavaScript problemlos verwenden zu können. Somit muss keine zusätzliche Bibliothek verwendet werden um Daten konsumieren zu können. Über die Rückgabeparameter werden Fehlercodes und Daten übertragen um dem Benutzer über die Website Feedback und Informationen zu seinen Eingaben zu geben.

11 Organisation

11.1 Projektorganisation

Für die Organisation wurde vom Team ein Projektleiter bestimmt. Dieser hat projektinterne Treffen, Treffen mit dem Auftragsgeber und Treffen mit dem Betreuungslehrer organisiert. Weiters wurden alle Dokumente der Projektgruppe für die Diplomarbeit vom Projektleiter aufbewahrt und organisiert. David Sedlak übernahm die Rolle des Projektleiters in RGHMS.

Das Backend wurde vom ganzen Team erarbeitet, da das Backend die Basis für jedes Mitglied bildet.

Jedes der 3 Frontendkomponenten wurde von je einem Mitglied entwickelt. Somit war Georg Aschauer für die Desktop-Applikation, David Sedlak für die mobile Applikation und Dominik Mühlbachler für die Website verantwortlich.

Datum	Meilenstein	Beschreibung	Ergebnis
07.10.2016	Pflichtenheft	Ein Pflichtenheft das alle Anforderungen von RGHMS regelt	Fertiges Pflichtenheft, sowohl vom Auftraggeber als auch von den Projektmitglieder unterschrieben worden ist
31.10.2016	Backend fertigstellen	Fertigung eines Backendes, das online über eine API abrufbar ist	Ein Azure-Server, der die RGHMS-DB über einen WCF konsumieren lässt
31.01.2017	Erste Prototypen	Prototypen der Frontend-Komponenten mit Grundfunktionalität	Siehe Individuelle Meilensteinliste
14.02.2017	Integrationstest	Testen des System in praktischen Anwendungsfällen	Findung Designfehler und Bugs in der mobilen App und Desktop-Anwendung
05.04.2017	Projektabgabe	Abgabe der fertigen Diplomarbeit in fertiger Form	

Tabelle 2 : Meilensteinliste der Organisation

11.1.1 Fortschrittanalyse

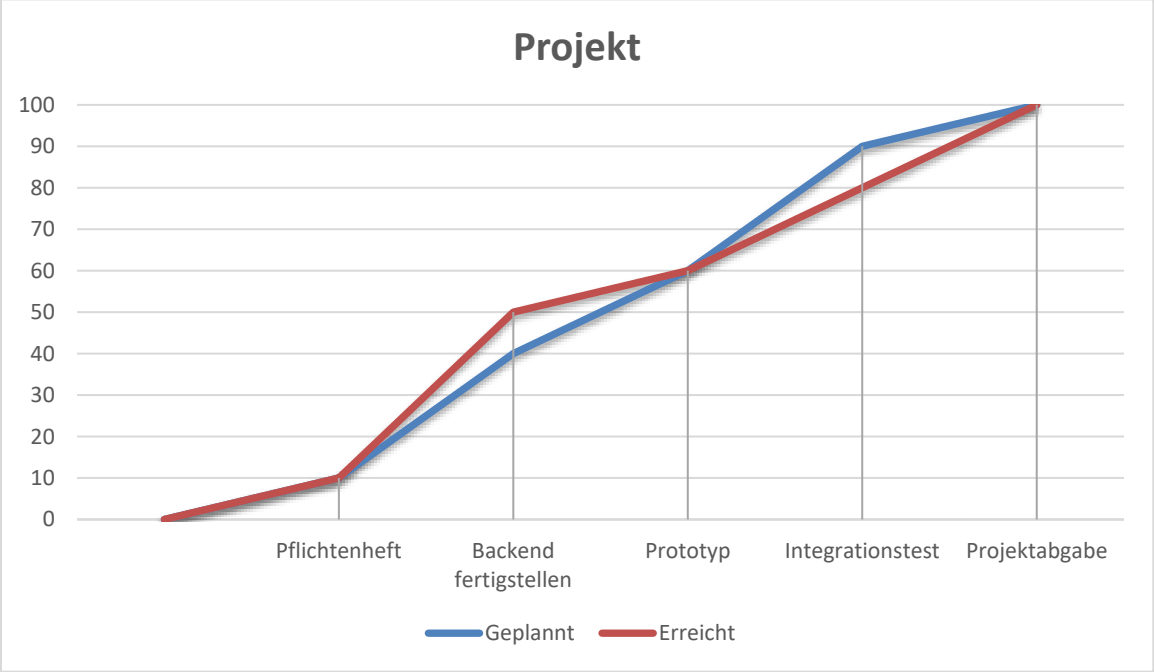

Abbildung 32 : Fortschrittanalyse des Projekts

Die ersten 3 Meilensteine der Diplomarbeit konnten alle zeitgerecht und planmäßig erfüllt werden. Durch relativ frühen Beginn der Entwicklung des Backends, wurde dieser Meilenstein sogar früher fertig als erwartet. Der Integrationstest konnte nicht termingerecht durchgeführt werden. Dies lag vor allem an den unterschiedlichen Fortschritt der individuellen Komponenten des Frontends. Der Integrationstest für die mobile Applikation lief ohne Probleme, der für die Desktop-Applikation ebenfalls, trotz der zeitlichen Verzögerung. Für die Website konnte kein Test durchgeführt werden.

11.2 Backend

Da alle Komponenten die gleichen Daten verwenden sollen, wurde das Backend vom gesamten Team erarbeitet. Vor allem bei der Entwicklung des Datenbankmodells wurde besonderer Wert auf die Zusammenarbeit alle Mitglieder gelegt, da Änderungen in der Datenbank Auswirkungen auf jede Komponente im System haben. Zur Erarbeitung des Backend wurden außerschulische und wöchentliche Treffen organisiert, um eine zeitgerechte Fertigstellung zu ermöglichen.

11.2.1 Meilensteinliste

Datum	Meilenstein	Beschreibung	Ergebnis
15.9.2016	Server aussuchen	Serveranbieter vergleichen und geeigneten auswählen	Entscheidung fiel auf einen Microsoft Azure Cloud Server
31.9.2016	Datenbank in Testumgebung aufsetzten	Datenbank in einer Windowsserver VM einrichten.	Windows Server 2012 R2 mit einer Instanz des Microsoft SQL Servers 2012 Express. Die Tabellen wurden entsprechend dem Modell unter 7.2.5 erstellt
11.10.2016	Webservice erstellen und auf IIS in der Testumgebung veröffentlichen	Einen SOAP Service für die Datenbank in der Windowsserver VM erstellen und auf des IIS des Windowsservers veröffentlichen	Ein WCF-Dataservice welche die Bearbeitung der Daten ermöglich wurde auf dem Webserver IIS erstellt
18.10.2017	Migration der Datenbank und Webservices auf den Azure Server	Die Datenbank und den Webservice auf der VM exportieren und im Azure Server importieren	Der Webservice und die Datenbank wurden auf den Azure Server migriert und sind von überall aus erreichbar

Tabelle 3 : Meilensteinliste des Backends

11.2.2 Fortschrittanalyse

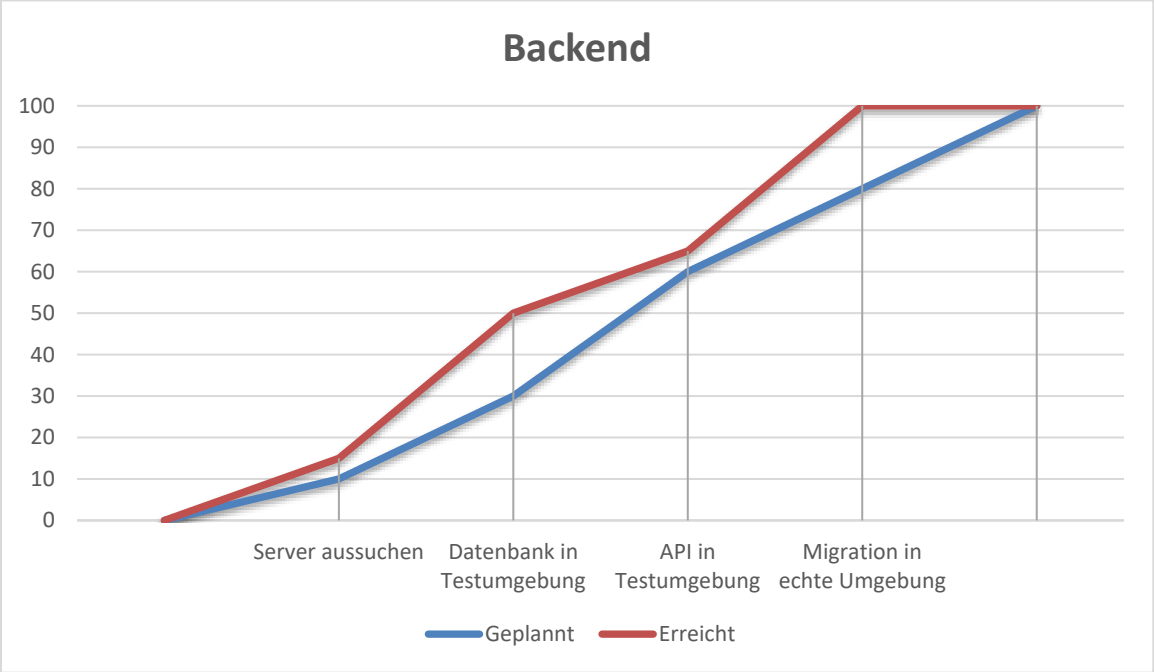

Abbildung 33 : Fortschrittanalyse des Backends

Wie bereits in der Analyse des Projekts erwähnt wurde, konnten die Meilensteine des Backends alle frühzeitig erreicht werden. Probleme in der Entwicklung der API nahmen mehr Zeit in Anspruch als erwartet und verringerten den relativ hohen Vorsprung Meilensteinliste. Die unerwartet einfache Migration von Datenbank und API in die echte Umgebung entwickelten wieder einen größeren Vorsprung.

11.3 Die Desktop-Applikation

11.3.1 Meilensteinliste

Datum	Meilenstein	Beschreibung	Ergebnis
31.10.2016	Basis Applikation	Applikation mit Verbindung zum Azure Server	WPF Applikation mit einer Verbindung zum WCF
30.11.2016	Applikation die Daten anzeigt	Applikation kann über den WCF Service Daten erhalten	Applikation kann Daten anzeigen und hinzufügen
31.12.2016	Applikation die Daten bearbeiten kann	Applikation kann die Daten mittels eines GUIs Bearbeiten	Applikation kann die Daten bearbeiten nur das Löschen von Einträgen deren Personen mehrere Eintrage haben funktioniert nicht
07.1.2017	Erster Prototyp	Applikation die Daten anzeigen, bearbeiten und filtern kann. Diverse Ansichten für freie Zimmer, Frühstücke und Personen.	Applikation kann alle Daten bearbeiten und löschen. Daten können gefiltert werden. Es sind verschiedene Ansichten für Zimmer, Frühstücke, Personen und Kassiere vorhanden
31.1.2017	Fertige Desktop Applikation	Applikation die den oben genannten Funktionen und einen ansprechenden Design	Applikation mit den genannten Funktionen und Design passend zu der App und Website

Tabelle 4 : Meilensteinliste der Desktop-Applikation

11.3.2 Fortschrittanalyse

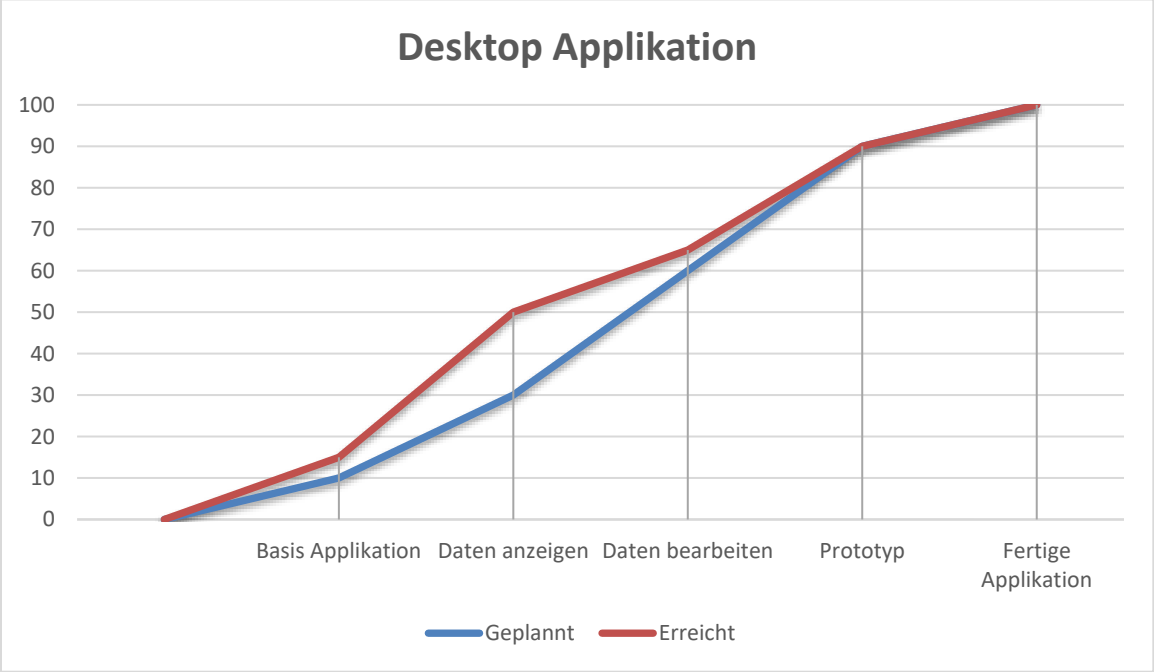

Abbildung 34 : Fortschrittanalyse der Desktop Applikation

Wie bereits in der Analyse des Projekts erwähnt wurde, konnten die Meilensteine der Desktop Applikation erreicht werden. Die Vollendung der Prototypen hat mehr Zeit in Anspruch genommen und verringerte damit den relativ hohen Vorsprung in der Meilensteinliste. Da die Erstellung der Entity Klassen zu Beginn einwandfrei funktionierte, kam dadurch ein ziemlicher Vorsprung zustande.

11.4 Mobile Applikation

11.4.1 Meilensteinliste

Datum	Meilenstein	Beschreibung	Ergebnis
31.10.2016	Einlesen die Sprache Xamarin und Anlegen der Applikation	Einarbeiten in die Sprache Xamarin und Erstellen einer Applikation in Xamarin	Probleme von Xamarin gefunden und zur besser passenden Lösung, Android gewechselt
30.11.2016	Kommunikation mit WCF und ungefähres Design	Android-App kann Daten konsumieren und Grobkonzept für das spätere Design	RESTlet-Bibliothek eingebunden um Kommunikation zu ermöglichen und Grobkonzept entwickelt
31.12.2016	Umsetzung des Designs	Praktische Anwendung des Grobkonzepts	Android mit diversen Ansichten, die echte Daten enthalten
07.1.2017	Push-Notifikation für Reservierungen	Push-Notifikation falls eine Reservierung eingeht und diese bestätigt werden muss.	Verwerfung des Designs der Push-Notifikationen durch den Auftragsgeber, Verbesserte Navigation und Reservierungsfunktion
31.1.2017	Fertige Mobile App	Fehlende Funktionen ergänzen	Fertige Mobil-App

Tabelle 5 : Meilensteinliste der mobilen Applikation

11.4.2 Fortschrittanalyse

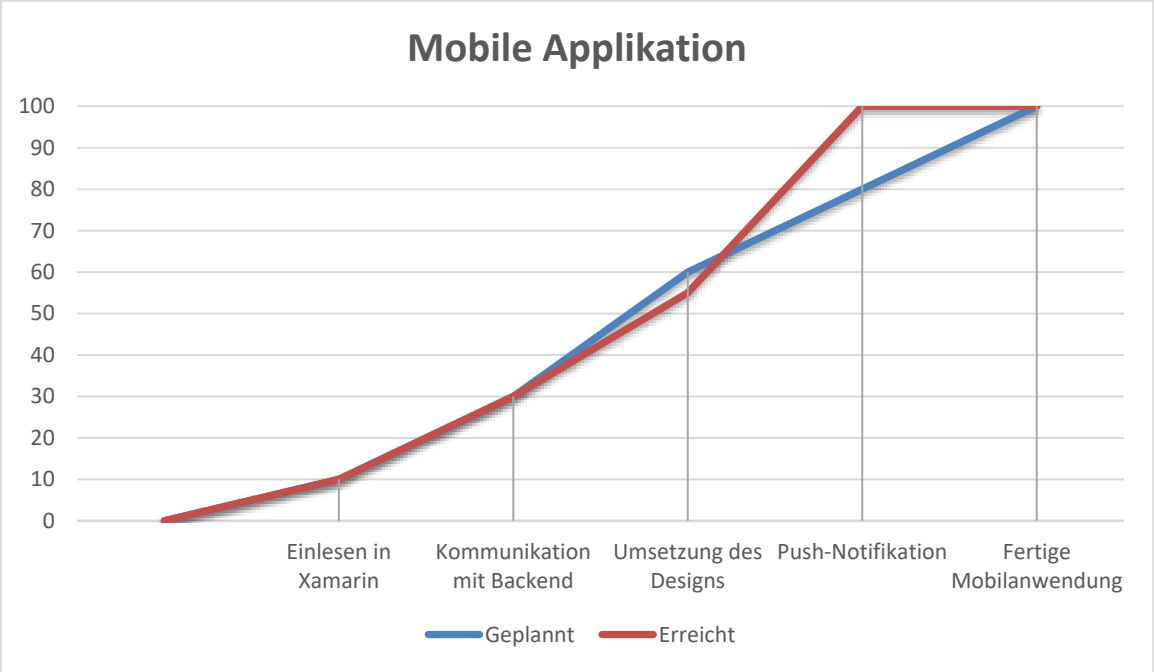

Abbildung 35 : Fortschrittanalyse der mobilen Applikation

Durch frühe Umstellung von Xamarin auf Android ging nur sehr wenig Zeit verloren und die Entwicklung verzögerte sich nicht.

Kommunikation mit den Backend erwies sich als nicht einfach da das Android-SDK die Kommunikation mit einer API, wie die von RGHMS, nicht unterstützt. Mit erhöhten Zeitaufwand konnte das Problem mit einer zusätzlichen Bibliothek gelöst werden während der Meilenstein im Zeitplan erreicht wurde.

Durch das Versagen einer Festplatte gingen Änderung an der Applikation verloren und stießen denn Erfolg zurück. Da der Versuch der Datenrettung erfolglos blieb, wurde dieser Rückschlag durch mehr Arbeitszeit und Engagement kompensiert.

Der Push-Notifikation-Meilenstein ist von einer konzeptionellen Änderung gezeichnet, weswegen der Name Navigation und Überarbeitung der Reservierung akkurater wäre. Diese Änderung verzögerte das Fertigwerden der App allerdings nicht und die App konnte durch das bereits zuvor erwähnte Arbeitsverhalten frühzeitig fertiggestellt werden.

11.5 Website

11.5.1 Meilensteinliste

Datum	Meilenstein	Beschreibung	Ergebnis
31.10.2016	Eruierung welche Sprache/Technologie	Bestehende Frage ob ASP.Net, HTML oder ein CMS verwendet wird	CMS wurde als kaum geeignet erachtet, der Prototyp wurde verworfen und eine Website aus simplen HTML mit vorgegebener CSS gestaltet
30.11.2016	Basiswebsite erstellt	Das Erstellen einer statischen Website	Nicht erreicht, Festlegung auf Responsiveness
31.12.2016	Erste Features die mit dem WCF Kommunizieren	Auslesen von Daten des WCF und/oder raufschreiben dieser	Statische Website mit Google Maps Plugin, Aufgrund von Cross-Site Exception, ein Meilenstein zurück
07.01.2017	Sämtliche Charts mit chart.js und Kalenderauswahl	Die Visualisierung von Daten welche aus dem WCF ausgelesen wurden und Simpleres Eingeben von einem Datum	Hinzugefügtes Formular mit responsiven Webdesign Noch immer Cross Site Exception
31.01.2017	Fertige Website und am Server gehostet	Vollständige Funktionalität mit allen Informationen und von überall Erreichbar	Hosten auf dem Server funktioniert, jedoch fehlt etwas an Funktionalität bzgl. Frühstück, auslesen von Freien Räumen etc.
17.02.2017	Fertige Website und am Server gehostet	Vollständige Funktionalität mit allen Informationen und von überall Erreichbar	Auslesen von Freien Räumen und die Vollständige Reservierung funktionieren
31.03.2017	Fertige Website und am Server gehostet	Vollständige Funktionalität mit allen Informationen und von überall Erreichbar	Die Darstellung und restliche Funktionalität existieren im vollen Umfang und die Website ist fertig

Tabelle 6 : Meilensteinliste des Website

11.5.2 Fortschrittanalyse

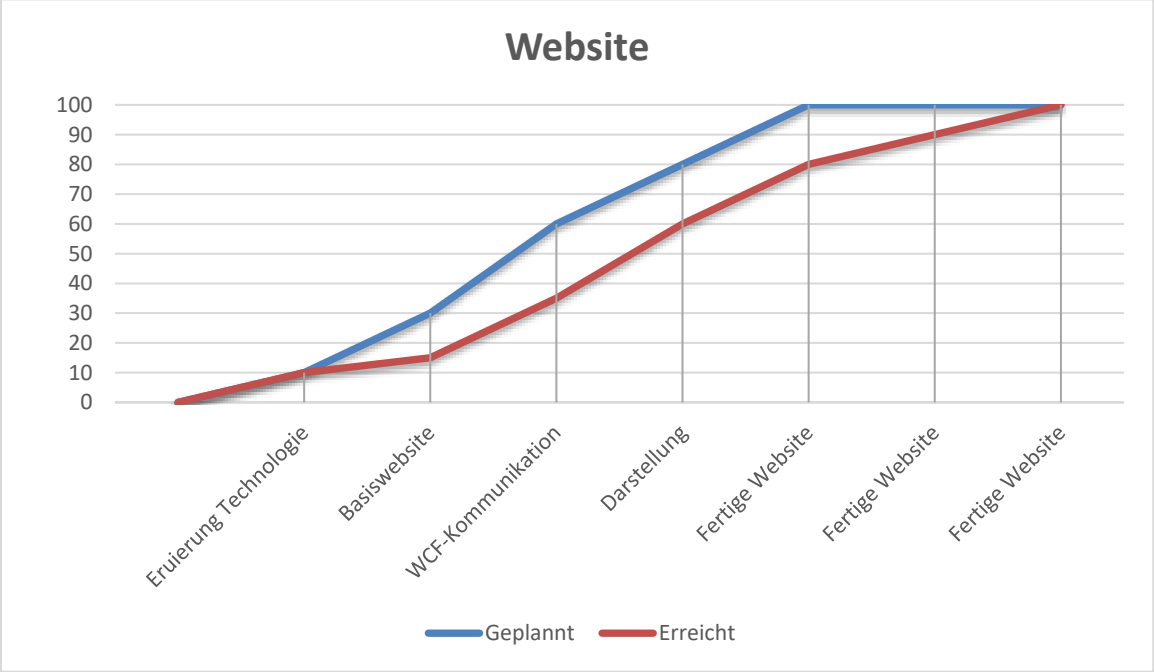

Abbildung 36 : Fortschrittanalyse der Website

Trotz ordnungsgemäßen Starts fiel die Entwicklung der Website bereits beim zweiten Meilenstein um kontinuierlich ein bis zwei Meilensteine zurück. Bei dem Versuch diese Zeit aufzuholen stieß man auf mehrere Probleme, die sich nur eingeschränkt lösen ließen da diese bereits von den anderen Applikationen verwendet wurden und diese auf die Funktionalität vertrauten. Somit war eine Gratwanderung vonnöten und man wurde vor mehrere Problematiken gestellt. Letztendlich wurde eine aktuelle Lösung für die Kommunikation mit dem Backend gefunden die dessen Funktionalität weder abändert noch einschränkt, sondern ausschließlich erweitert.

Aufgrund der Bekanntheit der folgenden auftretenden Probleme und einem ausreichendem Maß an Erfahrung mit diesen, sei es von früheren Projekten oder von nebenbei laufenden, konnten weitere zeitliche Einbußen vermieden werden und die Website wurde dank des zeitlichen Spielraums nach hinten noch rechtzeitig vor dem Abgabetermin fertiggestellt.

12 Probleme

12.1 Backend

12.1.1 Begrenzte Server Ressourcen

Die 1.75 GB Arbeitsspeicher des Azure Servers machen das Arbeiten darauf ziemlich langsam vor allem da über 1.25 GB des Rams bereits durch das Betriebssystem und die Datenbank belegt sind. Es ist dadurch, unmöglich eine Entwicklungsumgebung wie Visual Studio auf dem Server zu installieren. Diese wird allerdings benötigt, um den Webservice zu erstellen. Man könnte Visual Studio nun natürlich einfach auf einem Windows-Gerät mit genug Ressourcen installieren, jedoch hat man dann das Problem, dass man nicht auf die Datenbank zugreifen kann. Der Zugriff auf die Datenbank wird allerding benötigt um sich mit Hilfe des Entity Frameworks die Entity Klassen auf der Datenbank generieren zu lassen. Man könnte nun natürlich einfach den Port der Datenbank durch die Firewall lassen, das ist aber aus sicherheitstechnischen Gründen nicht sehr geeignet.

Die bessere Lösung ist eine mit dem Server möglich identische Umgebung in einer VM aufzusetzen. Man erstellt auf einem PC mit reichlich Leistung eine Windows Server 2012 R2 VM. Dieser VM ordnet man dann die benötigten Ressourcen zu, um sowohl Visual Studio als auch die Datenbank zu installieren. Dann erstellt man die gewünschten Tabellen in der VM und generiert sich die Entity Klassen daraus. Danach veröffentlich man dies auf dem IIS der VM und migriert die veröffentlichten Dateien von der lokalen VM auf den Azure Server.

12.1.2 Datenbank funktioniert nach geändertem Password nicht mehr

Ein Windows Server verlangt es, alle 3 Monate, dass Password zu ändern. Dies führt allerdings manchmal zu Problemen. In dem Fall vom Microsoft Azure Server von RGMS hat plötzlich jeder Zugriff auf die Datenbank einen Fehler geliefert. Grund dafür ist, dass der Dienst, der Datenbankzugriffe ermöglicht, noch das alte Password verwendet, um sich bei der Datenbank anzumelden. Um das zu beheben, muss man sich unter Dienste den „SQL Server(Express)“ Dienst suchen und unter Optionen das Password auf das neue Password ändern.

12.2 Mobile App

12.2.1 Kommunikation mit dem Backend

Das Android-SDK unterstützt die Kommunikation mit Webservices standardmäßig nicht. Neben der Verbindung zur Kommunikation werden für App Klassen benötigt, welche die vom Service übertragenen Daten in Form von Objekten speichern können.

Die RESTlet-Bibliothek schafft für beide Probleme Abhilfe. Mit der Bibliothek ist es möglich, über einen Servicelink Klassen generieren zu lassen, welche die später übertragenen Daten repräsentieren können und eine Service-Klasse, welche die Verbindung zum Service bilden kann. Zusätzlich zu den Klassen, muss die Bibliothek natürlich in die Applikation eingebunden werden.

12.2.2 Netzwerkkommunikation in Android

Der Versuch Netzwerkkommunikation im Main-Thread einer Android-Applikation durchzuführen ist programmiertechnisch nicht erlaubt. Um solch eine Kommunikation durchzuführen, muss ein Thread oder wie in RGHMS ein AsyncTask implementiert werden. Dieser muss dann die Funktionalität für die Netzwerkkommunikation beinhalten.

12.3 Website

12.3.1 Kaum Kommunikationsmöglichkeiten zu WCF

Es war nicht möglich direkt und Asynchron auf die Standardoperationen eines WCF zuzugreifen, weshalb ein Workaround mit offengelegten Methoden gefunden werden musste. Dieser Umweg hat auch einen weiteren Vorteil bezüglich der Übertragung der Daten, welche vom Volumen betrachtet, minimiert wurden. Die gesamte Logik und das Anlegen der Objekte und Einträge wird nun im Backend geregelt, welches nur die Identifikationsinformationen und wenige Grunddaten entgegennimmt. Darüber hinaus werden über diese Methoden Fehlercodes in den Returnparametern gesendet um die Website über etwaige Probleme zu informieren.

12.3.2 Cross Site Aufrufe

Standardmäßig ist das Aufrufen eines WCF von einer anderen Domain oder Website nicht gestattet, weswegen dies Explizit in der Konfigurationsdatei erlaubt werden muss. Dies erforderte ergiebige Recherche da sich in den benötigten XML Direktiven sich in den vergangenen Jahren vieles änderte.

13 Resümee

Rückblickend gesehen, wurde in dieser Diplomarbeit ein System entwickelt, welches die Verwaltungsarbeit in der Pension Remserhof digitalisiert. Auch wenn an dem einen oder anderen Ende die Komplexität einer Aufgabe überschätzt wurde, war das Diplomarbeitsteam in der Lage die Arbeit erfolgreich fertig zu stellen.

Zusammenfassend war die Diplomarbeit, trotz einem nicht zu unterschätzenden Zeitaufwand, überaus wertvoll für unsere Zukunft. Wir sammelten viele Erfahrungen mit den verwendeten Technologien und lernten neben mehreren technischen auch Myriaden von Lektionen auf der organisatorischen- und sozialen Seite, welche uns wohl noch lange begleiten werden.

14 Verfassungsnachweis

14.1 David Sedlak

2 Danksagung

4 Kurzbeschreibung

5 Einführung 5.1 Auftraggeber

5 Einführung 5.2 Ausgangssituation

5 Einführung 5.3 Problemstellung

5 Einführung 5.4 Verbesserung durch RGHMS

6 Funktionen von RGHMS 6.1 Desktop-Applikation

6 Funktionen von RGHMS 6.2 Mobile Applikation

6 Funktionen von RGHMS 6.3 Website

7 Funktionalität aus Sicht des Nutzers 7.2 Mobile Applikation

7 Funktionalität aus Sicht des Nutzers 7.2.1 Installation

7 Funktionalität aus Sicht des Nutzers 7.2.3 Startseite

7 Funktionalität aus Sicht des Nutzers 7.2.4 Navigation in der App

7 Funktionalität aus Sicht des Nutzers 7.2.5 Neue Reservierungen

7 Funktionalität aus Sicht des Nutzers 7.2.6 Übersicht: Frühstück

7 Funktionalität aus Sicht des Nutzers 7.2.7 Übersicht: Reservationen

7 Funktionalität aus Sicht des Nutzers 7.2.8 Übersicht: Gäste

7 Funktionalität aus Sicht des Nutzers 7.2.9 Übersicht: Blacklist

7 Funktionalität aus Sicht des Nutzers 7.2.10 Übersicht: Zimmer

8 Technische Umsetzung 8.2 Mobile Applikation

9 Grundlagen 9.1 Multi-Tier-Architektur

9 Grundlagen 9.1.1 Kommunikation zwischen den Schichten

9 Grundlagen 9.5 Mobile Applikation

9 Grundlagen 9.5.1 Android

9 Grundlagen 9.5.2 RESTlet

9 Grundlagen 9.5.3 GraphView

9 Grundlagen 9.5.4 Android Studio

10 Vorgehensweise 10.1 Auswahl der Ressourcen/Technologien

10 Vorgehensweise 10.1.1 Backend

10 Vorgehensweise 10.1.2 Frontend

10 Vorgehensweise 10.2.2 Mobile App

11 Organisation 11.1 Projektorganisation

11 Organisation 11.1.1 Meilensteinliste

11 Organisation 11.1.2 Fortschrittanalyse

11 Organisation 11.4 Mobile Applikation

11 Organisation 11.4.1 Meilensteinliste

11 Organisation 11.4.2 Fortschrittanalyse

12 Probleme 12.2 Mobile App

12 Probleme 12.2.1 Kommunikation mit dem Backend

12 Probleme 12.2.2 Netzwerkkommunikation in Android

14.2 Georg Aschauer

3 Abstract

7 Funktionalität aus Sicht des Nutzers 7.1 Desktop-Applikation

7 Funktionalität aus Sicht des Nutzers 7.1.1 Übersicht

7 Funktionalität aus Sicht des Nutzers 7.1.2 Menüleiste

7 Funktionalität aus Sicht des Nutzers 7.1.3 Kassier

7 Funktionalität aus Sicht des Nutzers 7.1.4 Blacklist

7 Funktionalität aus Sicht des Nutzers 7.1.5 Frühstücke

7 Funktionalität aus Sicht des Nutzers 7.1.6 Räume

7 Funktionalität aus Sicht des Nutzers 7.1.7 Export

8 Technische Umsetzung 8.1 Desktop-Applikation

8 Technische Umsetzung 8.3 Website

9 Grundlagen 9.2 Backend

9 Grundlagen 9.2.2 Virtual Machine

9 Grundlagen 9.2.3 VMware Workstation

9 Grundlagen 9.2.4 Datenbank

9 Grundlagen 9.2.5 Microsoft SQL Server

9 Grundlagen 9.2.6 Datenmodell

9 Grundlagen 9.2.7 Datenmodel von RGHMS

9 Grundlagen 9.2.8 C#

9 Grundlagen 9.3.1 ASP.NET

9 Grundlagen 9.3.2 WCF

9 Grundlagen 9.3.3 Webserver

9 Grundlagen 9.3.4 IIS

9 Grundlagen 9.3.5 Entity Framework

9 Grundlagen 9.3.6 API

9 Grundlagen 9.3.7 SOAP Service

9 Grundlagen 9.3.8 SQL Server 2014 Management Studio

9 Grundlagen 9.4 Desktop-Applikation

9 Grundlagen 9.4.1 LINQ

9 Grundlagen 9.4.2 WPF

9 Grundlagen 9.4.3 XML

9 Grundlagen 9.4.4 Visual Studio

9 Grundlagen 9.4.5 .NET Framework

10 Vorgehensweise 10.1 Auswahl der Ressourcen/Technologien

10 Vorgehensweise 10.1.1 Backend

10 Vorgehensweise 10.1.2 Frontend

10 Vorgehensweise 10.2 Implementierung des Systems

10 Vorgehensweise 10.2.1 Backend

10 Vorgehensweise 10.2.3 Desktop Applikation

11 Organisation 11.2 Backend

11 Organisation 11.2.1 Meilensteinliste

11 Organisation 11.2.2 Fortschrittanalyse

11 Organisation 11.3 Die Desktop-Applikation

11 Organisation 11.3.1 Meilensteinliste

11 Organisation 11.3.2 Fortschrittanalyse

12 Probleme 12.1 Backend

12 Probleme 12.1.1 Begrenzte Server Ressourcen

12 Probleme 12.1.2 Datenbank funktioniert nach geändertem Password nicht mehr

14.3 Dominik Mühlbachler

7 Funktionalität aus Sicht des Nutzers 7.3 Die Website

7 Funktionalität aus Sicht des Nutzers 7.3.1 Navigation

7 Funktionalität aus Sicht des Nutzers 7.3.2 Orientierung

7 Funktionalität aus Sicht des Nutzers 7.3.3 Reservierung

7 Funktionalität aus Sicht des Nutzers 7.3.4 Freie Plätze

7 Funktionalität aus Sicht des Nutzers 7.3.5 Annullierung irrtümlicher Reservierungen

8 Technische Umsetzung 8.3 Website

9 Grundlagen 9.6 Website

9 Grundlagen 9.6.1 HTML

9 Grundlagen 9.6.2 HTML5

9 Grundlagen 9.6.3 JavaScript

9 Grundlagen 9.6.4 JQuery

9 Grundlagen 9.6.5 Ajax

9 Grundlagen 9.6.6 Bootstrap

9 Grundlagen 9.6.7 CMS

9 Grundlagen 9.6.8 Notepad++

9 Grundlagen 9.6.9 Mobirise

9 Grundlagen 9.6.10 HTTP

10 Vorgehensweise 10.2.4 Website

10 Vorgehensweise 10.2.4.1 Aufsetzen auf dem Azure Server

10 Vorgehensweise 10.2.4.2 Darstellung

10 Vorgehensweise 10.2.4.3 WCF

11 Organisation 11.5 Website

11 Organisation 11.5.1 Meilensteinliste

11 Organisation 11.5.1 Fortschrittsanalyse

12 Probleme 12.3 Website

12 Probleme 12.3 Kaum Kommunikationsmöglichkeiten zu WCF

12 Probleme 12.3 Cross Site Aufrufe

13 Resümee

15 Textquellen

https://en.wikipedia.org/wiki/Microsoft_Azure

https://en.wikipedia.org/wiki/Virtual_machine

https://en.wikipedia.org/wiki/VMware_Workstation

https://en.wikipedia.org/wiki/Database

https://en.wikipedia.org/wiki/Microsoft_SQL_Server

https://en.wikipedia.org/wiki/Data_model

https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

https://en.wikipedia.org/wiki/ASP.NET

https://en.wikipedia.org/wiki/Windows_Communication_Foundation

https://en.wikipedia.org/wiki/Web_server

https://en.wikipedia.org/wiki/Entity_Framework

https://en.wikipedia.org/wiki/Application_programming_interface

https://en.wikipedia.org/wiki/Soap

https://en.wikipedia.org/wiki/Android_(operating_system)

https://en.wikipedia.org/wiki/Restlet

https://en.wikipedia.org/wiki/Android_Studio

https://en.wikipedia.org/wiki/Language_Integrated_Query

https://en.wikipedia.org/wiki/Windows_Presentation_Foundation

https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

https://en.wikipedia.org/wiki/.NET_Framework

https://en.wikipedia.org/wiki/HTML

https://en.wikipedia.org/wiki/HTML5

https://en.wikipedia.org/wiki/JQuery

https://en.wikipedia.org/wiki/Ajax_(programming)

https://en.wikipedia.org/wiki/Bootstrapping

https://en.wikipedia.org/wiki/Content_management_system

https://en.wikipedia.org/wiki/Multitier_architecture

https://en.wikipedia.org/wiki/Business_logic#Business_logic_layer

https://en.wikipedia.org/wiki/Data_access_layer

https://en.wikipedia.org/wiki/Service_layer

https://restlet.com/documentation/client/

https://developer.android.com/develop/

16 Abbildungsverzeichnis

Abbildung 1 : Desktop-Applikation Übersicht 16
Abbildung 2 : Desktop-Applikation Menüleiste 16
Abbildung 3 : Desktop-Applikation Kassier 17
Abbildung 4 : Desktop-Applikation Blacklist 17
Abbildung 5 : Desktop-Applikation Früstücke 18
Abbildung 6 : Desktop-Applikation Räume 18
Abbildung 7 : Desktop-Applikation Exportmeldung 19
Abbildung 8 : Desktop-Applikation Export 19
Abbildung 9 : mobile Applikation Startseite 20
Abbildung 10 : mobile Applikation Navigation 21
Abbildung 11 : mobile Applikation neue Reservierung 22
Abbildung 12 : mobile Applikation Frühstücke gruppiert 23
Abbildung 13 : mobile Applikation Frühstücke im Detail 23
Abbildung 14 : mobile Applikation Reservationen 24
Abbildung 15 : mobile Applikation Gäste 25
Abbildung 16 : mobile Applikation Blacklist 26
Abbildung 17 : mobile Applikation Zimmer 27
Abbildung 18 : Website Navigation 28
Abbildung 19 : Website Google Maps 28
Abbildung 20 : Website Reservierung 29
Abbildung 21 : Website Reservierungsbestätigung 30
Abbildung 22 : Multi-Tier-Architektur 33
Abbildung 23 : Datenbankmodel 37
Abbildung 24 : MS Azure Ressourcenauswahl 50
Abbildung 25: MS Azure Serverklassen 50
Abbildung 26 : Remoteverbindung 51
Abbildung 27 : Installation des Webservers 51
Abbildung 28 : Code des WCF-Dataservices 53
Abbildung 29 : Konsum von Daten eines Webservices 55
Abbildung 30 : Übergabe von Daten an den Webservice 55
Abbildung 31 : manuell konfigurierte Methode des Backends 56
Abbildung 32 : Fortschrittanalyse des Projekts 58
Abbildung 33 : Fortschrittanalyse des Backends 60
Abbildung 34 : Fortschrittanalyse der Desktop Applikation 62
Abbildung 35 : Fortschrittanalyse der mobilen Applikation 64
Abbildung 36 : Fortschrittanalyse der Website 66

17 Tabellenverzeichnis

Tabelle 1 : Datenbankmodel 38
Tabelle 2 : Meilensteinliste der Organisation 57
Tabelle 3 : Meilensteinliste des Backends 59
Tabelle 4 : Meilensteinliste der Desktop-Applikation 61
Tabelle 5 : Meilensteinliste der mobilen Applikation 63
Tabelle 6 : Meilensteinliste des Website 65