

# PROJEKTARBEIT



Thema:

Verwaltungssoftware für Hotels und  
Unternehmensorganisation

ABGABE BEI:

Herr Zimmermann

Herr Röther

Abgabeort:

CSS – Neckarsulm

Abgabetermin:

22.06.22

Autoren: Youssef Sbai & Ramon Grothe

## 1 Inhaltsverzeichnis

<b>2</b>	<b>Einleitung – Teil I Unternehmensorganisation</b>	<b>3</b>
2.1	Projektziel	3
2.2	Projektbegründung und Lösungsweg	3
2.3	Ist-Zustand	3
2.4	Soll-Zustand	3
<b>3</b>	<b>Hauptteil – Teil I Unternehmensorganisation</b>	<b>4</b>
3.1	Unternehmensziele	4
3.2	Aufgabenanalyse	5
3.3	Aufgabensynthese - allgemein	7
3.3.1	Aufgabensynthese – Produktentwicklung	8
3.4	Organisationsstruktur & Organigramm	8
3.5	Stellenbeschreibung	9
3.6	Führungsstil & Führungstechnik	11
3.7	Unternehmensleitbild	11
3.8	Corporate Identity	12
<b>4</b>	<b>Fazit – Teil I Unternehmensorganisation</b>	<b>13</b>
4.1	Soll-/Ist-Vergleich	13
4.2	Reflexion	13
4.3	Ausblick	13
<b>5</b>	<b>Einleitung – Teil II Softwaredokumentation</b>	<b>14</b>
5.1	Projektumfeld	14
5.2	Projektziel	14
5.3	Projektbegründung & Lösungsweg	14
5.4	Ist-Zustand	14
5.5	Soll-Zustand	15
<b>6</b>	<b>Hauptteil – Teil II Softwaredokumentation</b>	<b>15</b>
6.1	Planungsphase	15
6.1.1	Programm	15
6.1.2	Datenbankstruktur	16
6.2	Designphase	17
6.3	WPF-Implementierung	18
6.3.1	Login-Oberfläche	18
6.3.2	Admin-Oberfläche	19

6.4	Implementierung der Geschäftslogik .....	20
6.4.1	Filterfunktionen.....	20
6.4.2	Suchfunktion .....	21
6.5	Datenbank .....	22
6.5.1	Tabellen .....	22
6.5.2	Stored Procedures.....	22
6.5.3	Datenbankzugriff.....	23
6.6	Unit Tests.....	24
<b>7</b>	<b>Fazit – Teil II Softwaredokumentation .....</b>	<b>24</b>
7.1	Soll-/Ist-Vergleich .....	24
7.2	Rückblick .....	25
7.3	Ausblick.....	25
<b>8</b>	<b>Abbildungsverzeichnis.....</b>	<b>26</b>
<b>9</b>	<b>Quellen- und Literaturverzeichnis .....</b>	<b>27</b>

## 2 Einleitung – Teil I Unternehmensorganisation

Die folgende Projektarbeit beschreibt den Ablauf der Unternehmensumstrukturierung und nachfolgend die Entwicklung einer Software für Hotelmanagement. Wir unterteilen die Arbeit in die Unternehmensorganisation und die Software. Im Folgenden beginnen wir mit dem Unternehmen.

### 2.1 Projektziel

Ziel dieses Projekts ist es, unser fiktives Unternehmen „YR Technology“ mit 16 Mitarbeitern, umzustrukturieren und zukunftssicher zu machen. Mit dieser nachhaltigen Planung ist es das Ziel das Unternehmen wachsen lassen zu können und sich eine stabile Position im Markt zu sichern.

### 2.2 Projektbegründung und Lösungsweg

Die Projektarbeit wurde im Rahmen einer Schularbeit zur Aufgabe gestellt. Die Motivation dahinter ist, neben einer guten Benotung, das gelernte Wissen anzuwenden und auch während der Arbeit neues zu erlangen.

Zur Lösungsfindung überlegen wir zunächst, wie die generelle Struktur des Unternehmens aussehen soll. Dazu erstellen wir auch Grafiken, um diese Überlegungen anschaulich darzustellen. Anschließend arbeiten wir die einzelnen gegebenen Punkte ab und fügen dies im Gesamten zur Dokumentation hinzu.

### 2.3 Ist-Zustand

Der Ausgangspunkt in diesem Projekt ist, dass wir ein junges Unternehmen mit 16 Mitarbeitern führen, welches sich aktuell im Wachstum befindet und stetig neue Mitarbeiter dazukommen. Die bisherige Struktur ist jedoch nicht auf ein expandierendes Unternehmen ausgelegt und benötigt dementsprechend eine Umstrukturierung.

### 2.4 Soll-Zustand

Nach der Umstrukturierung soll das Unternehmen eine nachhaltige Organisation haben, welche auf eine stetig wachsende Mitarbeiterzahl vorbereitet ist. Auch soll erreicht werden, dass das Unternehmen einen steten Umsatzzuwachs hat, da die Mitarbeiter effizienter arbeiten können.

### 3 Hauptteil – Teil I Unternehmensorganisation

Im Folgenden werden die einzelnen Arbeitsschritte aufgelistet und in die einzelnen Anforderungen unterteilt.

#### 3.1 Unternehmensziele

Um für die Organisation auch klare Ziele stecken zu können, haben wir 10 Unternehmensziele erarbeitet, welche wir mit der neuen Umstrukturierung erreichen wollen. Wir kategorisieren wir diese noch zur jeweiligen Zielebene:

- Leistungsziele:
  - Im 2. Quartal des Geschäftsjahres 2023 soll eine neue Software auf den Markt gebracht werden, welche ein Rundumpaket für alle Hotelmanagement-Tätigkeiten werden soll. Die Projektplanung und -umsetzung beginnt in Q3 2022.
  - Bis zum Ende des 3. Quartals des aktuellen Geschäftsjahres soll das Unternehmen eine neue Social-Media Präsenz aufbauen und bereits 200 Follower über verschiedene Plattform verteilt vorweisen.
- Finanzielle Ziele:
  - Zum Ende des aktuellen Geschäftsjahres 2022 soll der Umsatz im Vergleich zum Vorjahr um 8% gesteigert werden.
  - Durch Erlangung zusätzlicher Qualifikationen für Mitarbeiter sollen extern anfallende Kosten (z.B. für Lohnbuchhaltung) um 40 % bis Beginn 2023 reduziert werden.
- Organisations- und Führungsziele:
  - Die Anzahl der Mitarbeiter soll bis zum Ende des Geschäftsjahres 2024 auf 30 erhöht werden.
  - Um das Unternehmen für Arbeitnehmer attraktiv zu machen, wird ab dem folgenden Quartal eine monatliche, anonyme Mitarbeiterumfrage durchgeführt. Das Ergebnis soll hier bis Ende des aktuellen Geschäftsjahres in der Gesamtzufriedenheit um 10% erhöht werden.

- Soziale Ziele:
  - o Im 1. Quartal des nächsten Geschäftsjahres findet der Umzug in neue Büroräume statt, um auf den Mitarbeiterwachstum vorzubereiten. Dort soll gleichzeitig ein kleines Fitnessstudio für alle Mitarbeiter eingerichtet werden.
  - o Beginnend mit dem aktuellen Geschäftsjahr sollen alle Mitarbeiter mit dem Novembergehalt zusätzlich ein Weihnachtsgeld von 40% des regulären Monatsgehalts bekommen.
- Ökologische Ziele:
  - o Der bestehende Fuhrpark aus 3 Firmenwagen soll diese nach Ablauf der jeweiligen Leasing-Zeiten gegen Elektrofahrzeuge ausgewechselt werden.
  - o Im Geschäftsjahr 2023 beteiligen wir uns an 3 gemeinnützigen Projekten rund um den Umweltschutz. Wir werden Bäume pflanzen, uns an einer Müllsammlung beteiligen und eine Spendenaktion für den regionalen Umweltschutzverband unterstützen.

### 3.2 Aufgabenanalyse

Im nächsten Schritt führen wir eine Aufgabenanalyse, welche bereits im Unternehmen anfallen und in Zukunft anfallen werden. Diese Aufgabenanalyse wird uns folgend dabei helfen, welche Tätigkeitsbereiche für das Unternehmen relevant sind und wie wir die aktuelle Belegschaft aufteilen. Vorab erstellen wir eine Liste der anfallenden Aufgaben und haben diese dann zur besseren Einteilung in Aufgabenbereiche eingeteilt.

Personalwesen:

- Lohnbuchhaltung
- Recruiting
- Administration von Personaldaten
- Onboarding
- Weiterbildungsorganisation (geplant)
- Office Management

Produktentwicklung:

- Projektplanung
- Ressourcenplanung
- Scrum (geplant)
- Backend Strukturplanung
- Frontend Strukturplanung & Design
- Anwendungsentwicklung Backend
- Anwendungsentwicklung Frontend
- Testing

- QS
- Deployment
- Monitoring

#### Finanzen:

- Finanzbuchhaltung
- Budgetierung, Finanz- und Kostenplanung
- Jahresabschlüsse erstellen
- Cashflow überwachen
- Ansprechpartner für Steuer- und Wirtschaftsprüfer

#### Sales:

- Neukundenakquise
- Bestandskundenmanagement
- Produktpräsentation erstellen
- Angebotserarbeitung
- Rechnungsstellung
- Kundenberatung

#### Marketing:

- Social-Media Management (geplant)
- Marketingstrategie ausarbeiten (geplant)
- Konzept und Pflege der firmeneigenen Website

Die hier dargestellten Tätigkeitsbereiche werden wir in den folgenden Schritten benötigen, um eine ordentliche Organisationsstruktur zu erstellen.

### 3.3 Aufgabensynthese - allgemein

Die aus der Aufgabenanalyse gewonnenen Tätigkeitsbereiche nutzen wollen wir direkt in unserer Organisationsplanung einfließen lassen. Hier ergibt sich für uns eine neue Abteilungsstruktur (*siehe Abb. 1*), unter der wir nachfolgend die Mitarbeiter unterordnen können.

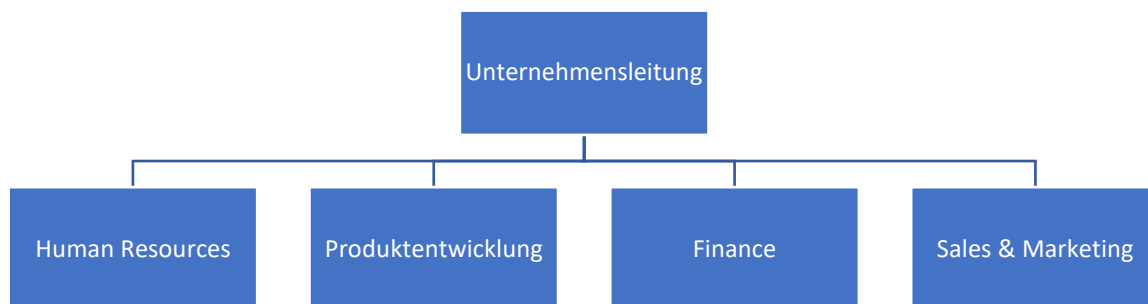


Abbildung 1: Unternehmensstruktur

Human Resources übernimmt hierbei die Tätigkeiten vom Personalwesen und wird bei eventuellem Ausfall (da aktuell nur 2 Mitarbeiter) durch die Assistenz der Geschäftsführung und ggf. Geschäftsführung unterstützt.

Die Aufgaben unter dem Punkt Finanzen werden von der Finance-Abteilung übernommen. Hier findet eine enge Zusammenarbeit mit der Geschäftsführung statt, besonders bei der Planung des Budgets und anfallenden Kosten.

Für Sales & Marketing haben wir uns entschieden die Funktionen in einer Abteilung zusammenzufassen. Die Abteilung wird auch hier durch die Assistenz der Geschäftsführung und die Geschäftsführung selbst unterstützt, besonders bei der Marketingstrategie. Bei der Kundenberatung steht die Sales Abteilung auch in engem Kontakt mit der Produktentwicklung.



### 3.3.1 Aufgabensynthese – Produktentwicklung

In der Produktentwicklung haben wir die meisten Mitarbeiter beschäftigt. Um hier langfristig den Arbeitsablauf zu ordnen, ist eine detailliertere Aufteilung zur jeweiligen Stelle notwendig (siehe Abb.2).

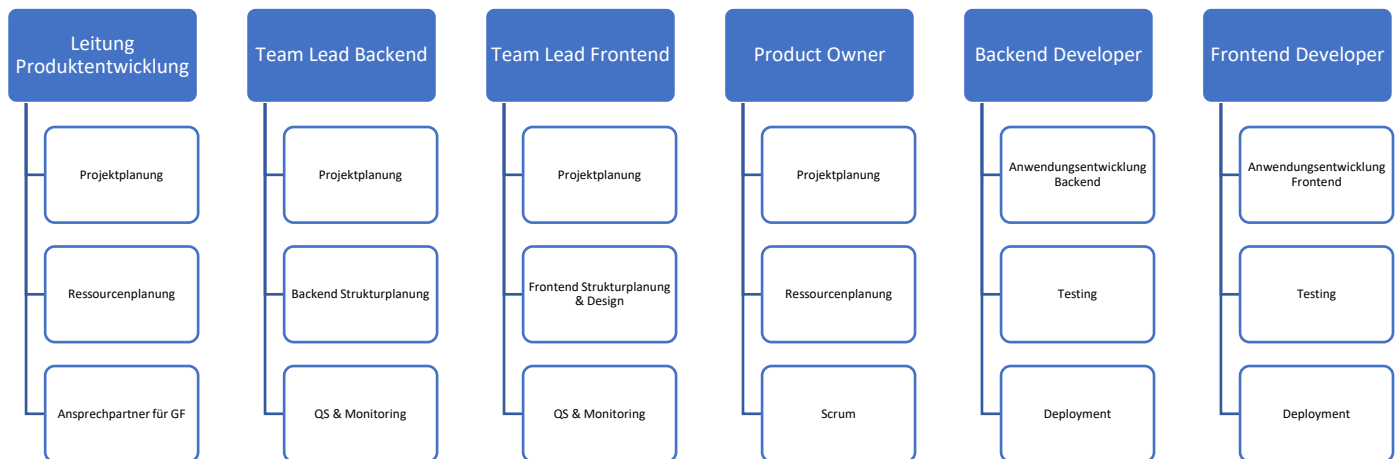


Abbildung 2: Aufgabenteilung Produktentwicklung

Mit dieser ausgearbeiteten Struktur können wir auch besser feststellen, in welchen Bereichen wir vorrangig neues Personal benötigen und dies dementsprechend priorisieren.

## 3.4 Organisationsstruktur & Organigramm

Anhand der Aufgabenanalyse fällt die Entscheidung zur Organisationsstruktur auf die funktionale Organisation. Bei dieser Struktur kann die Fachkompetenz der einzelnen Mitarbeiter optimal genutzt und ausgebaut werden.

Durch die klare Abgrenzung des Tätigkeitsbereichs kann jeder Mitarbeiter seine Expertise für die jeweilige Einzelaufgabe optimal nutzen.

Einen weiteren Vorteil sehen wir darin, dass diese organisatorisch auch gut skalierbar ist, wenn die Firma weiter wächst. So ist eine neue Umstrukturierung definitiv in den nächsten 5 Jahren nicht notwendig.

Um die Struktur deutlich aufzuzeigen, erstellen wir ein Organigramm (siehe Abb. 3) mit allen Mitarbeitern.

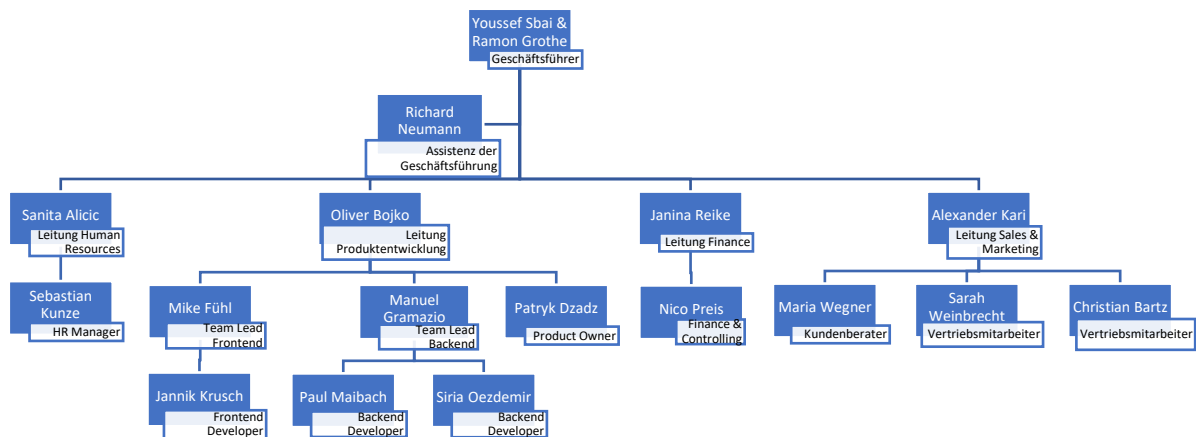


Abbildung 3: Organigramm

### 3.5 Stellenbeschreibung

Unsere Überlegung ist, dass wir eine detaillierte Stellenbeschreibung für eine Position aufsetzen, welche wir zeitnah besetzen möchte. Dadurch ist die Wahl auf den HR-Manager gefallen.

HR-Manager – Administration und Personalentwicklung

Unser HR-Manager mit Schwerpunkt Administration und Personalentwicklung ist in folgenden Bereichen tätig:

Administration:

- Anlegen und Führen von Personalakten
- Vorbereitung der Lohnbuchabrechnungsdaten für Dienstleister
- Prüfung der Zeiterfassung und Abwesenheiten und ggf. Rücksprache mit jeweiliger Führungskraft
- Personalbeurteilung erstellen (Empfehlungsschreiben, Zwischenzeugnisse, Abschlusszeugnisse)
- Unterstützung im Einstellungsprozess (Bewerbungsgespräche, Bewerbungsunterlagen prüfen)

Personalentwicklung:

- Führungskräfte in der Personalentwicklung beraten
- Personalentwicklungsmaßnahmen bzgl. Qualität, Nachhaltigkeit und Kosten prüfen
- Kontinuierliche Analyse von Personalentwicklungsbedarf und Umsetzung der Maßnahmen
- Unterstützung bei internen Workshops/Seminaren als Referent/Trainer
- Mitarbeitergespräche führen
- Abteilungen beim Onboarding neuer Mitarbeiter unterstützen

Der Stelleninhaber hat folgende Befugnisse:

- Einsicht und Änderungen in Personalakten aller Mitarbeiter vornehmen
- Mitbestimmungsrecht beim Einstellungsprozess, wenn dieser daran beteiligt ist
- Entscheidung und Genehmigung, welche Weiterbildungsangebote die Mitarbeiter wahrnehmen können
- Schlägt Mitarbeitern und Führungskräften Weiterbildungsmaßnahmen vor und kann diese auch verordnen

Folgende Voraussetzungen werden vom Stelleninhaber erfüllt:

- Erfolgreich abgeschlossene kaufmännische Ausbildung oder abgeschlossenes Studium im Bereich BWL / Personalwesen oder vergleichbares
- Selbstständige, strukturierte und eigenverantwortliche Arbeitsweise
- Sehr gute MS Office Kenntnisse
- Erkennen von Qualifikationsdefiziten und Weiterbildungspotenzial bei Mitarbeitern aus diversen Abteilungen
- Kann Weiterbildungsangebote beurteilen, ob diese die entsprechende Qualifikation vermitteln

Der Stelleninhaber ist dem Abteilungsleiter der Human Resources Abteilung direkt unterstellt.

Des Weiteren ist dem Stelleninhaber kein festangestellter Mitarbeiter unterstellt. Die Betreuung und Führung von Praktikanten, Auszubildenden oder Werkstudenten kann nach Rücksprache mit der Führungskraft erfolgen.

Der Stelleninhaber soll stets das Ziel verfolgen, die Mitarbeiter des Unternehmens in Ihrer Weiterentwicklung zu unterstützen, damit diese auf die vielseitigen Anforderungen unserer Kunden reagieren können.

Im Fall des eigenen Ausfalls wird der Stelleninhaber je nach aktueller Kapazität und Kompetenz durch Mitarbeiter in der Human Resources Abteilung vertreten. Gleichzeitig vertritt dieser die Kollegen der Abteilung, wenn diese ausfallen.

### 3.6 Führungsstil & Führungstechnik

Nach der Organisationsstruktur müssen wir uns entscheiden, welchen Führungsstil wir zukünftig anwenden wollen. In YR Technology werden wir den situativen Führungsstil ausüben. Gleiches gilt für alle Führungskräfte.

Durch diesen wird eine stete Weiterentwicklung unserer Mitarbeiter gefördert. Wir fordern und fördern diese, geben aber auch Unterstützung, sollte der Mitarbeiter mal nicht weiterwissen. Ziel davon ist es, dass diese durch generelle Eigenverantwortung motiviert werden, gleichzeitig aber auch die Sicherheit haben Hilfe in Anspruch nehmen zu können.

Wir werden hierfür mit unseren Führungskräften einen ausführlichen Workshop durchführen, damit jeder an einem Strang zieht.

Aufbauend darauf werden die Führungstechnik „Management by Delegation“. Mit dieser Technik soll den Mitarbeitern Verantwortung an die Führungskräfte übertragen werden. Dies hat sowohl den Effekt, dass die Führungskraft entlastet wird, als auch, dass der Mitarbeiter mit Eigenverantwortung motiviert wird.

### 3.7 Unternehmensleitbild

Wir möchten sowohl unseren Kunden als auch unseren Mitarbeitern klar aufzeigen, welche Werte wir in YR Technology vermitteln. Dazu erstellen wir auch anhand unserer Unternehmensphilosophie (siehe Abb. 4) ein Unternehmensleitbild, welche kurz und prägnant die wichtigsten Leitsätze aufzeigt.



Abbildung 4: Unternehmensphilosophie

Unser Unternehmensleitbild setzt sich aus 5 Leitsätzen zusammen:

1. Wir sehen unsere Innovation als Erfolgsfaktor an.
  - a. Heißt, unsere Innovation soll uns und unsere Kunden überzeugen.
  - b. Wir wollen daher auch nie stillstehen, unser Produkt ständig verbessern.
2. Wir wollen als Team gewinnen und uns weiterentwickeln.
  - a. Heißt, dass unser Erfolg nicht am Einzelnen liegt, sondern wir alle dazu beitragen.
  - b. Jeder soll am Erfolg beteiligt werden.
3. Wir wollen unseren Teil zum Umweltschutz beitragen.
  - a. Heißt, wir nehmen an gemeinnützigen Umweltprojekten in der Region teil.
  - b. Wir wollen auch intern unser Umweltbewusstsein zeigen, z.B. durch E-Autos,
4. Wir stellen bei unserer Technologie den Menschen in den Mittelpunkt.
  - a. Heißt, dass unsere Technologie die Arbeit erleichtern soll, nicht verkomplizieren.
  - b. Wir wollen unsere Technologie auch barrierefrei gestalten.
5. Wir wollen die Eigeninitiative und Motivation unser Mitarbeiter fördern.
  - a. Heißt, dass unsere Mitarbeiter den nötigen Freiraum bekommen, den sie benötigen.
  - b. Wir wollen dafür auch eine offene Feedback-Kultur schaffen.

### 3.8 Corporate Identity

Um YR Technology auch dementsprechend zu präsentieren, setzen wir auch auf für uns wichtige Corporate Identity Elemente.

Unser Corporate Design hat unser Firmenlogo und die Farbe Blau als Mittelpunkt. Das zeigen wir bereits in unserem Produkt und auch unsere neuen Büroräume nächstes Jahr sollen dies widerspiegeln. Die Firmenwagen werden auch das Logo von YR Technology präsentieren.

In der Kommunikation untereinander pflegen wir eine Du-Kultur. Genauso möchten wir das nach außen hin präsentieren. Dabei zählen nicht nur unsere Kunden und Partner, sondern auch die öffentliche Kommunikation, z.B. Social Media oder Werbung.

Unser Corporate Behavior soll das widerspiegeln. Wir möchten einen offenen Umgang aller Mitarbeiter und Führungskräften, unabhängig, ob Unternehmensleitung oder Praktikant. Auch wollen wir durch kleine Rituale, wie ein gemeinsames Frühstück einmal freitags im Monat den Austausch und Zusammenhalt fördern.

## 4 Fazit – Teil I Unternehmensorganisation

Zusammenfassend können wir sagen, dass wir eine solide Grundlage für YR Technology geschaffen haben. Wir haben die Struktur des Unternehmens von Grund auf neu überarbeitet und eine nachhaltige, zukunftsorientierte Organisation geschaffen. Unsere Mitarbeiter sind dabei auch sinnvoll zu Ihrer Tätigkeit in die passenden Bereiche eingeteilt worden. Auch haben wir aufgearbeitet, welche Aufgabe für die jeweilige Stelle zu tun ist.

### 4.1 Soll-/Ist-Vergleich

Rückblickend auf den vorherigen Ist-Zustand hat sich die Struktur zu unseren Wünschen verbessert. Die Struktur ist organisierter, die Aufgaben sind klar zugeteilt und es gibt ein einheitliches Auftreten. Damit sind die von uns gewünschten Punkte erfüllt.

### 4.2 Reflexion

Insgesamt ist der wirtschaftliche Teil dieser Projektarbeit gut gelaufen. Wir haben die Themen sorgfältig ausgearbeitet und diese nach bestem Wissen und Gewissen in unser Projekt einfließen lassen.

Entgegen der vorangegangenen Erwartung diesen Teil im Zeitraum von ca. 8 Unterrichtseinheiten abschließen zu können, hat der Teil des Projekts doch deutlich mehr Zeit in Anspruch genommen. Dies lag hier teils am Umfang der Aufgabe aber auch daran, dass noch ausführliche Recherche notwendig war, um die Aufgabenstellung zu unserer Zufriedenheit zu erfüllen. Hier werden wir zukünftig mehr auf den Umfang achten und uns vorab eine bessere Planung überlegen.

### 4.3 Ausblick

Auch wenn das Projektziel erreicht wurde, kann hier auf den geschaffenen Grundlagen zukünftig aufgebaut werden. Man könnte z.B. einen Börsengang simulieren oder einen Marketingplan für ein neues Produkt erstellen. Die Möglichkeiten sind dabei recht vielfältig.

Im Rahmen der Ausbildung werden wir auf diese Projektarbeit zurückschauen und aus unseren Fehlern lernen als auch Dinge, die positiv waren, in zukünftige Arbeiten einfließen lassen. Die Vorbereitung besonders im Hinblick auf die Abschlussprüfung ist hierbei von sehr großem Wert.

## 5 Einleitung – Teil II Softwaredokumentation

### 5.1 Projektumfeld

Die folgende Projektdokumentation beschreibt die Entwicklung eines Prototyps für eine Hotelsoftware zum Buchungsmanagement. Sie wird im Rahmen der Projektarbeit für die Fächer BT-B und BT-S erstellt, bei welcher sowohl die Unternehmensorganisation des fiktiven Unternehmens YR Technology beleuchtet wird als auch den Entwicklungsprozess einer Applikation für die Verwaltung eines Hotels.

In diesem Teil der Dokumentation wird nur die Entwicklung der Software beschrieben. Die Unternehmensorganisation findet sich im ersten Teil ab Seite 2.

### 5.2 Projektziel

Ziel des Projekts ist es, eine funktionierende Software für das Raumbuchungsmanagement eines Hotels zu erstellen. Dabei sollen die Daten in einer Datenbank gespeichert und abgerufen werden. Zusätzlich soll der Code objektorientiert aufgebaut werden und als Oberfläche entweder Windows Forms oder WPF (Windows Presentation Foundation) genutzt werden.

### 5.3 Projektbegründung & Lösungsweg

Die Entscheidung auf das Raumbuchungsmanagement fiel, weil dies für uns die naheliegendste Anwendung im Hotelbereich war und wir bei der Auswahl der Themen direkt darauf gestoßen sind. Die Software macht es für den Nutzer einfacher Räume zu buchen und die Buchungen einzusehen, zu bearbeiten und zu löschen.

Um dies zu erreichen, werden wir uns erst darauf festlegen, welches Framework wir für die Benutzeroberfläche nutzen wollen. Danach werden wir mithilfe eines ER-Modells unsere Datenbankstruktur skizzieren. Im Anschluss wird die Arbeit am Frontend der Software beginnen, welches später mit dem Backend inklusive Datenbank zusammengebracht wird.

### 5.4 Ist-Zustand

Es gibt bisher keine bestehende Software, auf welcher der Prototyp aufbauen soll. Innerhalb der Projektarbeit wird dieser neu entwickelt. Eine genauere Analyse des Ist-Zustands kann daher nicht erstellt werden.

## 5.5 Soll-Zustand

Am Ende dieser Projektarbeit soll eine funktionierende Software für das Buchungsmanagement entstehen. Diese soll sowohl einfach zu bedienen sein als auch die gegebenen Anforderungen mindestens erfüllen. Die Funktionen sollen dann auch präsentierbar werden können.

# 6 Hauptteil – Teil II Softwaredokumentation

## 6.1 Planungsphase

### 6.1.1 Programm

Bevor wir mit genauem Design-Konzept beginnen, machen wir uns erst grundlegende Gedanken über die Struktur unserer Software. Hier entscheiden wir uns dafür, die Solution in 3 Projekte (siehe Abb. 5) zu unterteilen.

In „Core“ findet sich später die Geschäftslogik zur Datenverarbeitung. In „Configuration“ hinterlegen wir Interfaces, Entitäten, Repositories und Zugriff auf die Datenbank. In „UI“ soll später das gesamte Frontend mit WPF abgelegt werden.

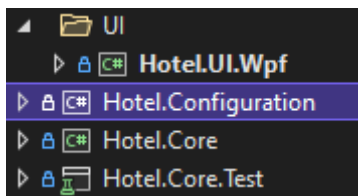


Abbildung 5: Solution Struktur

Zusätzlich haben wir noch ein Projekt für Unit-Tests erstellt, welche wir später planen zu implementieren, um unsere Backend-Methoden und Datenspeicherung zu testen.

Zuletzt wollen wir bereits die erste Logik hier in hinterlegen und haben eine Überprüfung erstellt, mit welcher das Programm sehen kann, ob die eingegebene E-Mailadresse dem korrekten Format entspricht. Hierfür haben wir ein Regex (siehe Abb. 6) genutzt, da dies für uns die einfachste Lösung war. Auch haben wir diese mit einem Unit-Test überprüft.

```
public class EmailCheckService : ICheckService<string>
{
    Regex _regexEmail = new Regex(@"[\w\d\.\_\-]+@[\w]+\.[\w]{2,}");

    /// <summary>
    /// This method checks if the input email is in the correct format of abc@xyz.efg with a Regex.
    /// </summary>
    /// <param name="email"></param>
    /// <returns>True if it matches the Regex or false if not</returns>
    public bool CheckIfValid(string email)
    {
        return _regexEmail.IsMatch(email);
    }
}
```

Abbildung 6: EmailCheckService mit Regex



### 6.1.2 Datenbankstruktur

Als nächstes befassen wir uns mit der Datenbankstruktur. Dafür überlegen wir zuerst welche Daten wir speichern müssen, um diese nachher zu verarbeiten. Hierfür haben wir uns mit einem Entity-Relationship-Modell geholfen, um die Struktur und Beziehungen der Daten grafisch darzustellen.

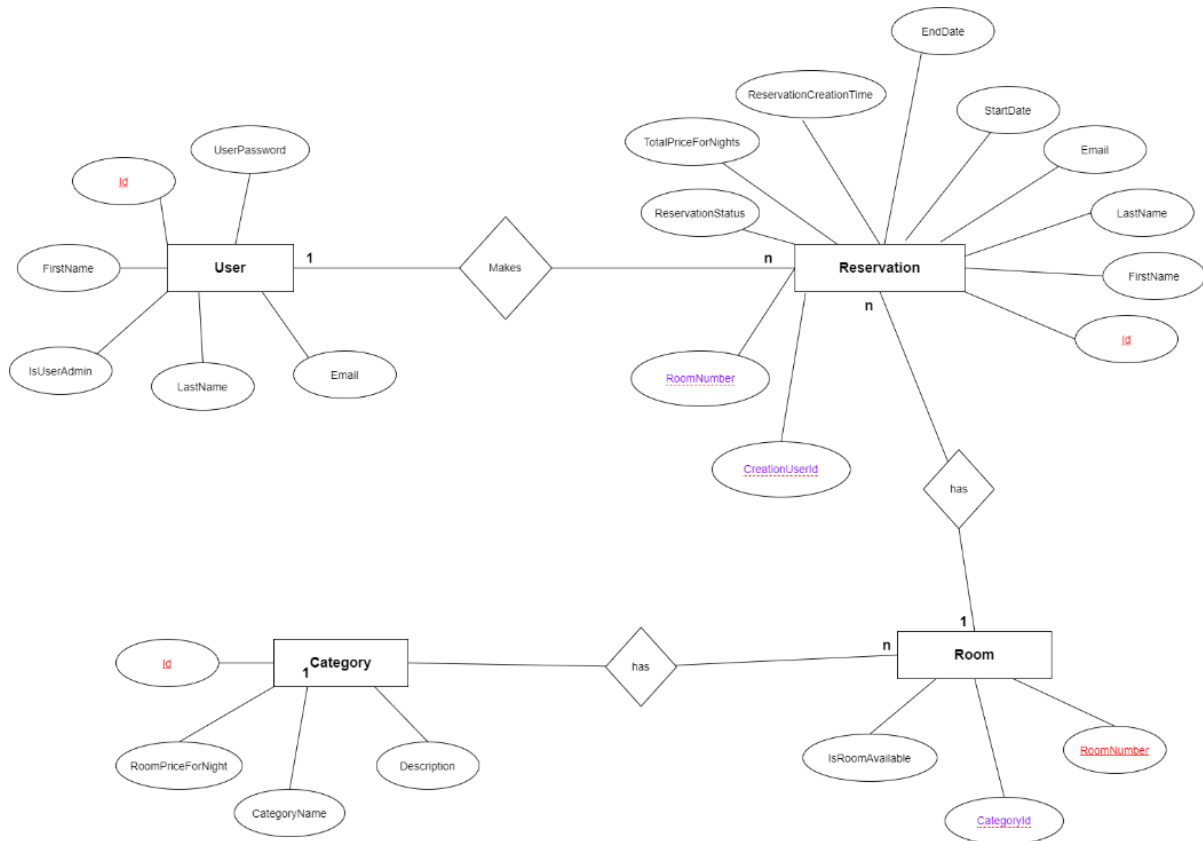


Abbildung 7: Entity-Relationship-Modell (Lila = Foreign Key, Rot = Primary Key)

Dazu gehörig erstellen wir noch das Relationsschema:

Reservation (Id, RoomNumber, CreationUserId, ReservationStatus, TotalPriceForNights, ReservationCreationTime, EndDate, StartDate, Email, LastName, FirstName)

User (Id, UserPassword, Email, LastName, FirstName, IsUserAdmin)

Category (Id, RoomPriceForNight, Categoryname, Description)

Room (RoomNumber, CategoryId, IsRoomAvailable)

Diese Daten haben wir im Anschluss als Entitäten in unseren Code unter „Configuration“ abgelegt.

## 6.2 Designphase

Im nächsten Schritt befassen wir uns mit dem Design und der WPF-Struktur unserer Anwendung.

Wir haben einige Prototypen erstellt, mit welchen wir uns ein erstes Bild machen wollen. Jedoch stellt sich hier bereits das Problem, dass es mit unserem aktuellen Wissensstand zu zeitaufwändig wird, ein für uns zufriedenstellendes Design zu erstellen. Ursprünglich haben wir uns aber genau deswegen für WPF entschieden, da es hier viel mehr Möglichkeiten gibt die Applikation auch ansprechend aussehen zu lassen (verglichen mit Windows Forms).

Nach einigen Versuchen haben wir uns entschieden nach einer Bibliothek für WPF Design Elementen zu schauen. Hier sind wir auf das „Material Design in XAML Toolkit“ gestoßen. Anhand der Demos können wir sehen, dass die Bibliothek genau die Styling-Möglichkeiten beinhaltet, die wir uns erhofft haben.

Nachdem wir uns nun eingelese haben, welche Möglichkeiten wir mit der Bibliothek haben, erstellen wir ein Design mithilfe von „Figma“ (siehe Abb. 8). Dies ist ein Designtool spezialisiert auf User Interface. In weiteren Verlauf dieses Projekt werden wir unser Figma-Design mit den gegebenen Änderungen erweitern.

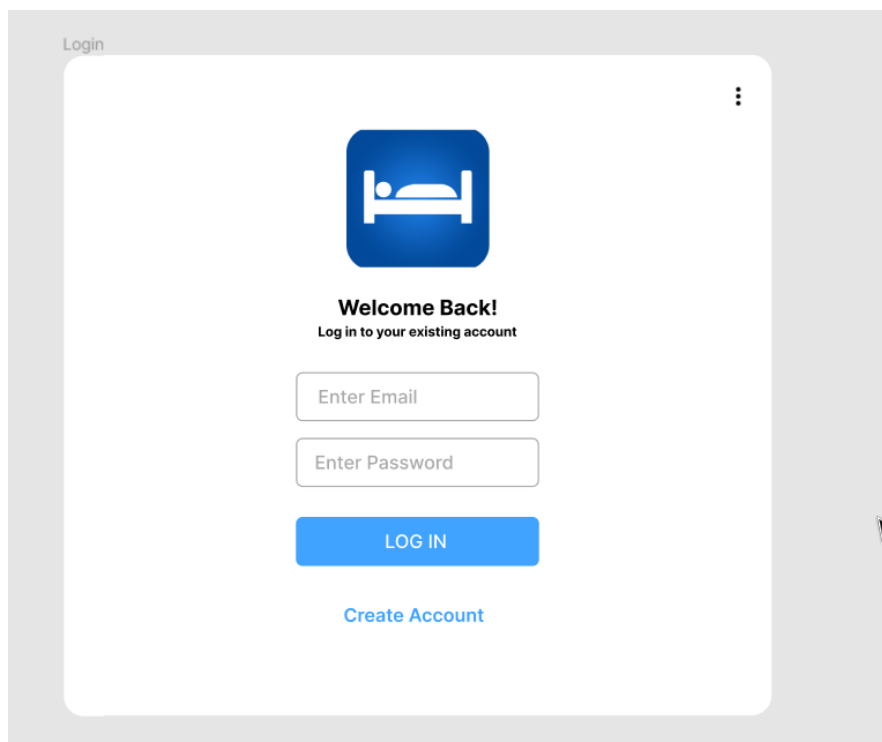


Abbildung 8: Erstes Figma-Design

## 6.3 WPF-Implementierung

Im ersten Schritt der Implementierungsphase müssen wir zuerst überlegen, wie wir unseren Code strukturieren wollen. Wir haben uns hier für „Model View ViewModel“ (kurz MVVM) entschieden. Hierdurch wird es für später einfacher sein, die einzelnen Designelemente (Views) von der Präsentationslogik (ViewModel) und der Geschäftslogik (Model) zu trennen.

### 6.3.1 Login-Oberfläche

Mit unserem bereits erstellen Design (*siehe Abb. 8*) haben wir Login-Oberfläche erstellt und zugehörigen „DataBindings“ erstellt, um die Views mit dem ViewModel zu verknüpfen.

Wir erkennen nun nach der Erstellung, dass unser Design nicht optimal gestaltet ist, da zum Beispiel die Applikation über ein Overflow-Menü (Drei-Punkte-Menü) und einem weiteren Klick geschlossen werden muss. Außerdem wollen wir das Firmenlogo und die Firmenfarbe von YR Technology nutzen und hervorheben.

Daher haben wir unser Design geändert und die Login-Oberfläche erneut bearbeitet (*siehe Abb. 9*).

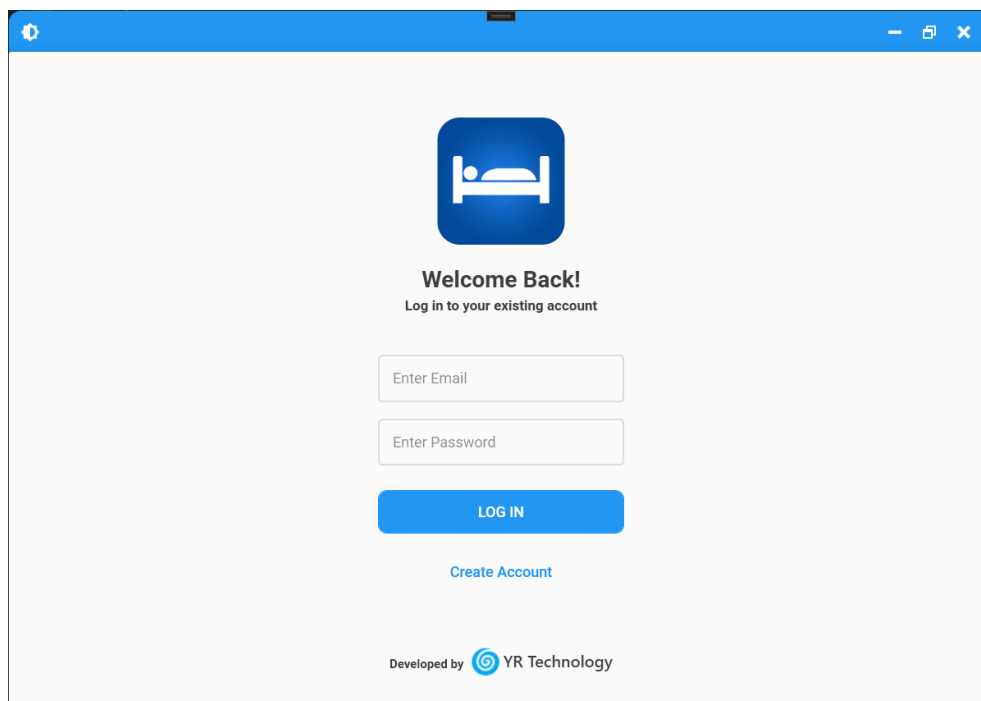


Abbildung 9: Neues Design der Login-Oberfläche

### 6.3.2 Admin-Oberfläche

Da die Applikation vorerst nur im Verwaltungsbereich genutzt werden soll, beschäftigen wir uns als nächstes mit der Admin-Oberfläche. Über diese wird später die Buchung von Zimmern möglich sein. Hier sind die Überlegungen entstanden, was wir noch inkludieren wollen. Dabei haben wir uns noch für ein User-Management entschieden, bei dem registrierte Benutzer verwaltet und gelöscht werden können. Als zweites wollen wir eine grafische Oberfläche implementieren, die eine grafische Übersicht der verfügbaren und belegten Räume liefern soll.

Für eine bessere Nutzerfreundlichkeit brauchen wir mit mehreren Features noch eine Navigation, in der der User zu der jeweiligen Oberfläche wechseln kann und sich auch ohne Schließen des Programms ausloggen kann. Dafür erstellen wir wieder zuerst ein Design und implementieren dies im Anschluss (siehe Abb. 10 und 11).



Abbildung 11: Design  
der Navigationsleiste

```
<ListViewItem Margin="0 10" >
  <Button HorizontalAlignment="Left" Height="auto" ToolTip="Reservation Management"
    BorderBrush="{x:Null}" Background="{x:Null}" Foreground="White"
    Command="{Binding OpenReservationCommand}" Style="{StaticResource MaterialDesignToolButton}">
    <StackPanel Margin="-5" Orientation="Horizontal">
      <materialDesign:PackIcon Kind="CalendarClock" Margin="20"/>
      <TextBlock Text="Reservations" VerticalAlignment="Center" Margin="10 10" FontSize="16"/>
    </StackPanel>
  </Button>
</ListViewItem>
<ListViewItem Margin="0">
  <Button Height="auto" ToolTip="User Management"
    BorderBrush="{x:Null}" Background="{x:Null}" Foreground="White"
    Command="{Binding OpenUserManagerCommand}" Style="{StaticResource MaterialDesignToolButton}">
    <StackPanel Margin="-5" Orientation="Horizontal">
      <materialDesign:PackIcon Kind="AccountSupervisor" Margin="20"/>
      <TextBlock Text="Users" VerticalAlignment="Center" Margin="10 10" FontSize="16"/>
    </StackPanel>
  </Button>
</ListViewItem>
```

Abbildung 10: Codeausschnitt der Navigationsleiste

Im Anschluss zu der Navigation wenden wir uns dem Reservierungsmanager zu. Auch hier erstellen wir erst ein Design-Konzept in Figma und beginnen erst danach mit der Umsetzung im Code.

Da sich unsere Buchungsdaten am besten tabellarisch darstellen und später auch sortieren lassen, haben wir uns hier für die GridView entschieden, um die Daten anzuzeigen.

Im Code richten wir zuerst wieder die Views ein und erstellten Bindings zu den ViewModels. Um Reservierungen hinzuzufügen, kommt hier noch der „Add Reservation“-Button hinzu. Der Reservierungsmanager (siehe Abb. 12) konnte nun abgeschlossen werden und hat für die einzelnen Buchungen bereits auch die Logik implementiert.

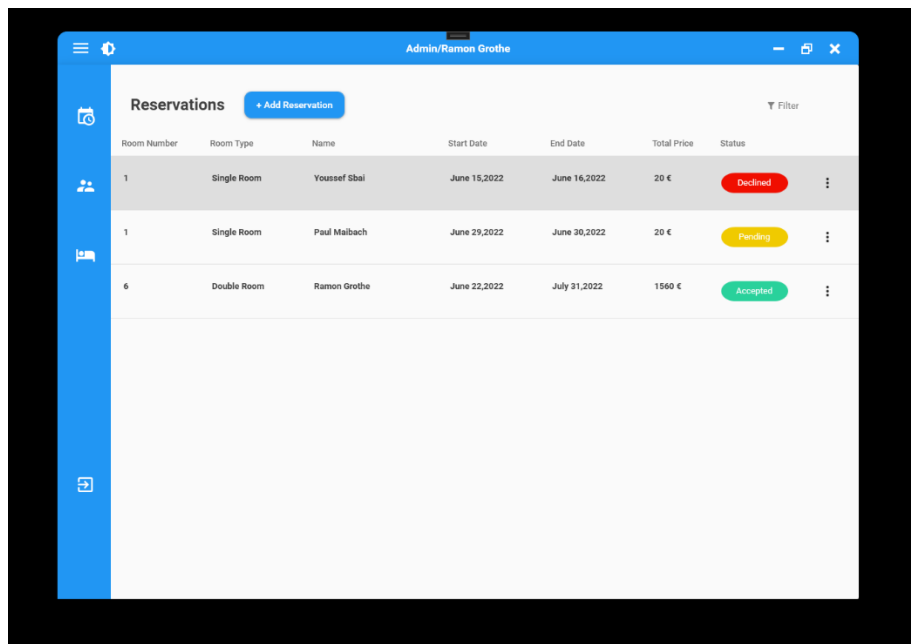


Abbildung 12: Reservierungsmanagement

Für das Usermanagement gehen wir einen ähnlichen Weg, da hier die Daten auch am besten über eine tabellarische Ansicht dargestellt werden können. Wie bereits in den Schritten davor wird hier erst Design in Figma erstellt und danach in den Code implementiert.

Unsere Grundidee im Login-System war, dass wir beim Neustart der App den vorherigen User eingeloggt lassen und keine neue Authentifizierung notwendig ist. Wir haben uns aber nach erneuten Überlegungen dagegen entschieden, da vor allem an Rezeptionen im Hotel oder generell im Büro an einem PC über den Tag verteilt mehrere Nutzer arbeiten. Daher sollte aus Sicherheitsgründen beim Schließen des Programms immer wieder ein neuer Login stattfinden, damit immer sichtbar ist, wer genau welche Buchung getätigt hat.

## 6.4 Implementierung der Geschäftslogik

### 6.4.1 Filterfunktionen

In der Nutzeroberfläche haben wir bereits einige Filteroptionen eingebaut, die man mit einer Checkbox anhaken kann. So soll es möglich sein auf die Gesamtliste an angezeigten Elementen mehrere Filter gleichzeitig zu nutzen.

Die einzelnen Filtermethoden (*siehe Abb. 13*) iterieren mithilfe von LINQ-Methoden anhand der gegebenen Parameter durch die Liste und geben eine neue, gefilterte Liste zurück.

```
/// <summary>
/// This method filters a list of IHotelReservation by the name with using LINQ's Where() to extract all reservations that contain the input first or last name.
/// </summary>
/// <param name="hotelReservations"></param>
/// <param name="name"></param>
/// <returns>A filtered list of IHotelReservation that is filtered by the name.</returns>
2 references | Ramon, 2 hours ago | 2 authors, 2 changes
public List<IHotelReservation> GetReservationsFromName(IList<IHotelReservation> hotelReservations, string name)
{
    return hotelReservations.Where(x => x.FirstName.ToLower().Contains(name.ToLower()) || x.LastName.ToLower().Contains(name.ToLower())).ToList();
}

/// <summary>
/// This method filters a list of IHotelReservation by the date with using LINQ's Where() to extract all dates that are in the range of the start date to end date.
/// </summary>
/// <param name="hotelReservations"></param>
/// <param name="startDate"></param>
/// <param name="endDate"></param>
/// <returns>A filtered list of IHotelReservation where the reservations are in the range of the start and end date.</returns>
2 references | Ramon, 2 hours ago | 2 authors, 2 changes
public List<IHotelReservation> GetReservationsFromDateRange(IList<IHotelReservation> hotelReservations, DateOnly? startDate, DateOnly? endDate)
{
    Func<DateOnly, DateOnly?, bool> isEndDateApplying = (endDate, targetedEndDate) =>
    {
        if (targetedEndDate == null)
        {
            return true;
        }
        else
        {
            return endDate <= targetedEndDate;
        }
    };
    return hotelReservations.Where(x => x.StartDate >= (startDate ?? new DateOnly()) && isEndDateApplying(x.EndDate, endDate)).ToList();
}
```

Abbildung 13: Codebeispiel Filtermethoden

Um die Funktionalität der Filter zu gewährleisten, haben wir einige Unit-Tests geschrieben, welche vorerst mit hard coded Listen arbeiten. Sobald wir die Datenbank aufgesetzt haben, werden wir mithilfe von „Mocking“ die Unit-Tests anpassen.

#### 6.4.2 Suchfunktion

Nach der Implementierung der Filtermethoden merken wir, dass diese nicht optimal zu dem passen, was wir im Usermanagement abbilden wollen.

Wir haben uns hier noch für eine Suchleiste entschieden, nach der wir Nutzer anhand des Vor- oder Nachnamens oder der Emailadresse suchen können. Zusätzlich haben wir noch eine

Combobox hinzugefügt, um dem Endnutzer die Möglichkeiten zu geben alle Ergebnisse nach der Rolle zu filtern.

Für die Backend-Logik nutzen wir hier die String Contains() Methode und LINQ (siehe Abb. 14).

```
/// <summary>
/// This static method takes <see cref="IList<<see cref="IUser"/>>"> of users and returns a filtered list from the search content
/// </summary>
/// <param name="searchContent"></param>
/// <param name="userList"></param>
/// <returns> empty <see cref="IList<<see cref="IUser"/>>"> if no user is found</returns>
1 reference | Youssef Sbai, 11 minutes ago | 1 author, 1 change
public static IList<IUser> GetUsersFromNameOrEmail(string searchContent, IList<IUser> userList)
{
    var filterSearchList = new List<IUser>(userList.Where(x =>
    {
        var name = x.FirstName + " " + x.LastName;
        return name.Contains(searchContent, StringComparison.OrdinalIgnoreCase)
        || x.Email.Contains(searchContent, StringComparison.OrdinalIgnoreCase);
    }));
    return filterSearchList;
}
```

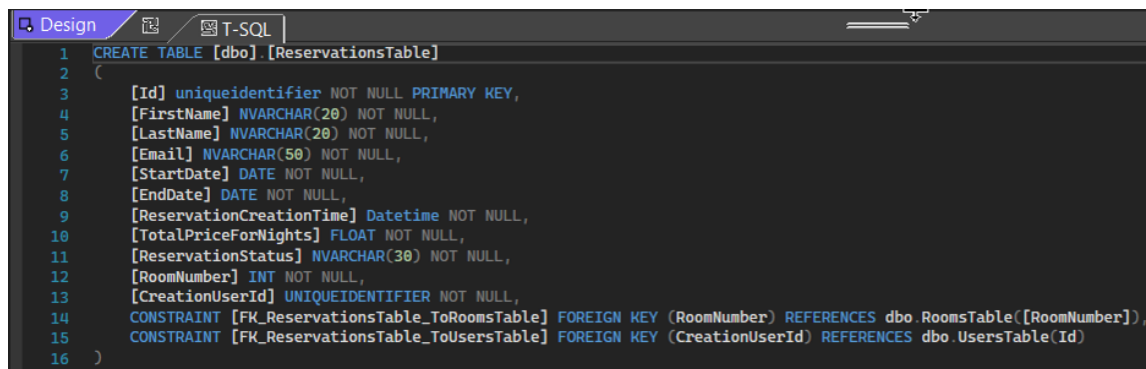
Abbildung 14: Suchmethode

## 6.5 Datenbank

Um die Datenbank besser über GitHub verwalten zu können und insgesamt eine sauberere Lösung zu finden, werden wir die Datenbank direkt in Visual Studio mittels der zu Verfügung stehenden Tools zu erstellen. Diese basiert auf SQL Server 2016 und wird direkt in der Solution mit abgebildet.

### 6.5.1 Tabellen

Zu Beginn erstellen wir mithilfe des zuvor angefertigten Entity-Relationship-Modells die einzelnen Tabellen mit den zuvor definierten Datentransferobjekten. Die Tabellen sind mit SQL-Befehlen (siehe Abb. 15) erstellt worden.



```
1 CREATE TABLE [dbo].[ReservationsTable]
2 (
3     [Id] uniqueidentifier NOT NULL PRIMARY KEY,
4     [FirstName] NVARCHAR(20) NOT NULL,
5     [LastName] NVARCHAR(20) NOT NULL,
6     [Email] NVARCHAR(50) NOT NULL,
7     [StartDate] DATE NOT NULL,
8     [EndDate] DATE NOT NULL,
9     [ReservationCreationTime] Datetime NOT NULL,
10    [TotalPriceForNights] FLOAT NOT NULL,
11    [ReservationStatus] NVARCHAR(30) NOT NULL,
12    [RoomNumber] INT NOT NULL,
13    [CreationUserId] UNIQUEIDENTIFIER NOT NULL,
14    CONSTRAINT [FK_ReservationsTable_ToRoomsTable] FOREIGN KEY ([RoomNumber]) REFERENCES dbo.RoomsTable([RoomNumber]),
15    CONSTRAINT [FK_ReservationsTable_ToUsersTable] FOREIGN KEY ([CreationUserId]) REFERENCES dbo.UsersTable([Id])
16 )
```

Abbildung 15: SQL-Befehl, um Tabelle zu erstellen

### 6.5.2 Stored Procedures

Um die Datenbankbefehle nicht direkt im Code eingeben zu müssen und diese eventuell zu wiederholen, schreiben wir Stored Procedures.

Hier müssen wir uns erst überlegen, wie wir Passwörter in der Datenbank hinterlegen wollen. Diese einfach als String einzuspeichern wäre trotz Einfachheit keine gute Lösung.

Daher haben wir die SQL Hash-Funktion genutzt, um das Passwort bei Eintrag in der Datenbank zu verschlüsseln. Diese wurde mit in die Stored Procedure (siehe Abb. 16) eingefügt.

```
CREATE PROCEDURE [dbo].[InsertUser]
    @Id uniqueidentifier,
    @FirstName nvarchar(30),
    @LastName nvarchar(30),
    @Email nvarchar(50),
    @Password nvarchar(50),
    @IsUserAdmin Bit
AS
Begin
    SET NOCOUNT ON
    Insert into dbo.UsersTable (Id, FirstName, LastName, Email, UserPassword, IsUserAdmin)
    Values (@Id, @FirstName, @LastName, @Email, HASHBYTES('SHA2_512', @Password), @IsUserAdmin);
End
```

Abbildung 16: Stored Procedure mit Hash-Funktion

Da wir das Passwort nicht mehr im Datentransferobjekt speichern, haben wir auch noch eine Stored Procedure für die Authentifizierung beim Login geschrieben. Diese prüft, ob das eingegebene Passwort zu der Emailadresse valide ist.

Zum Schluss haben wir noch ein „Post Deployment“ Skript geschrieben, welche die Datenbank mit Dummy-Daten und Usern füllt. Dies ist erforderlich, da wir keine ausgelagerte Datenbank haben, welche über den Dienst online erreichbar ist. So können wir die lokal instanziierte Datenbank direkt mit Daten füllen.

### 6.5.3 Datenbankzugriff

Die Verbindung unserer Datenbank zum Backend übernimmt bei uns die Bibliothek „Dapper“. Mit dieser können wir recht einfach unsere Stores Procedures und den Connection-String verwenden, um auf die Datenbank zuzugreifen.

Hier bemerken wir aber ein Problem mit unserer Datenstruktur. Wir verwenden in den Datentransferobjekten den neuen Typ „DateOnly“, welcher mit .NET 6 hinzugefügt wurde. Dieser wird aber noch nicht von Dapper unterstützt.

Da wir definitiv bei Dapper bleiben wollen, da es den Datenbankzugriff sehr einfach gestaltet, versuchen wir hier eine Lösung zu finden.

Hierfür haben wir einen TypeHandler (siehe Abb. 17) erstellt, welcher beim Speichern in die Datenbank DateOnly in DateTime umwandelt und beim Lesen der Datenbank DateTime in DateOnly zur Verwendung im Programm umwandelt.

```
public class DapperSqlDateOnlyTypeHandler : SqlMapper.TypeHandler<DateOnly>
{
    0 references | Youssef Sbai, 10 days ago | 1 author, 1 change
    public override void SetValue(IDbDataParameter parameter, DateOnly date)
    => parameter.Value = date.ToDateTime(new TimeOnly(0, 0));

    0 references | Youssef Sbai, 10 days ago | 1 author, 1 change
    public override DateOnly Parse(object value)
    => DateOnly.FromDateTime((DateTime)value);
}
```

Abbildung 17: TypeHandler für DateOnly



## 6.6 Unit Tests

Wir haben bereits während der Implementierung der Geschäftslogik Unit-Tests geschrieben. Diese sind aber noch darauf ausgelegt mit Listen zu arbeiten, nicht mit der Datenbank.

Wir müssen nun die Unit Tests refaktorisieren und dem Datenbankzugang anpassen. Dabei ist zu beachten, dass wir die Tests nicht direkt über eine Datenbank laufen lassen wollen.

Um das zu Umgehen nutzen wir „Mocking“, damit wir zum Test ein Dummy-Repository (siehe Abb. 23) erstellen. Hierfür kommt die Bibliothek „Moq“ zum Einsatz.

```
private IRepository<T> GetMockedModelRepo<T>(List<T> repoModelList, Func<T,T,bool> matchingFunc) where T : class
{
    var mockedHotelRoomRepo = new Mock<IRepository<T>>();
    mockedHotelRoomRepo.Setup(x => x.GetAll()).Returns(repoModelList);
    mockedHotelRoomRepo.Setup(x => x.CreateNewModel(It.IsAny<T>())).Callback<T>(x => repoModelList.Add(x));
    mockedHotelRoomRepo.Setup(x => x.DeleteModel(It.IsAny<T>())).Callback<T>(x => repoModelList.RemoveAll(m => matchingFunc(m, x)));
    mockedHotelRoomRepo.Setup(x => x.UpdateModel(It.IsAny<T>())).Callback<T>(x =>
    {
        var modelIndex = repoModelList.FindIndex(m => matchingFunc(m, x));
        if (modelIndex == -1)
        {
            throw new ArgumentNullException(nameof(x));
        }
        repoModelList[modelIndex] = x;
    });
    return mockedHotelRoomRepo.Object;
}
```

Abbildung 18: Dummy Repository im Unit Test mit Moq erzeugt

## 7 Fazit – Teil II Softwaredokumentation

### 7.1 Soll-/Ist-Vergleich

Gehen wir auf den ursprünglichen Ist-/Soll-Zustand zurück, haben wir die an uns selbst gestellten Anforderungen voll umfänglich erfüllt.

Während der Laufzeit des Projektes brachten wir aber immer wieder neue Pläne und Features ein, die außerhalb des Rahmens dieses Projekts lagen. Hier kamen das Usermanagement, eine Registrierungsmaske mit Logik, eine grafische Zimmerübersicht und ein durchdachtes Login-Management hinzu.

Bis auf die grafische Zimmerübersicht fügten wir auch diese neuen Feature-Ideen unserem Projekt hinzu.

## 7.2 Rückblick

Rückblickend hat die Arbeit am Projekt viel Spaß gemacht. Wir haben beide viel Spaß daran gehabt unsere Ideen in das Projekt einzubringen und umzusetzen. Auch konnten wir durch das gegenseitige Unterstützen viel dazulernen. Dabei hat uns besonders Youssefs Expertise in WPF geholfen, welche er bereits abseits von diesem Projekt erlangt hat.

Was uns letztlich aber oft zurückgeworfen hat war, dass wir sehr oft neue Ideen hatten und sich der Umfang des Projekts ständig vergrößert hat. Bis zu dem Punkt, dass der letzte Stand unseres Projektes über 30000 Codezeilen umfasst.

## 7.3 Ausblick

Da das Projekt im Rahmen einer schulischen Aufgabe entstanden ist, werden wir sporadisch noch Features zu Übungszwecken hinzufügen oder den Code nochmal als Referenz nehmen. Darüber hinaus ist die Arbeit am Projekt jedoch beendet.

Für die Zukunft werden wir nicht nur das erlernte Fachwissen aus diesem Projekt mitnehmen, sondern auch das Arbeiten an diesem selbst. Damit ist gemeint, dass wir auch viel über Projektplanung und Arbeiten im Team gelernt haben. Diese Erkenntnisse werden uns nicht nur im Schulalltag bei zukünftigen Projektarbeiten weiterhelfen, sondern auch im Arbeitsalltag selbst.

## 8 Abbildungsverzeichnis

Abbildung 1: Unternehmensstruktur .....	7
Abbildung 2: Aufgabenteilung Produktentwicklung .....	8
Abbildung 3: Organigramm .....	9
Abbildung 4: Unternehmensphilosophie .....	11
Abbildung 5: Solution Struktur .....	15
Abbildung 6: EmailCheckService mit Regex .....	15
Abbildung 7: Entity-Relationship- Modell .....	16
Abbildung 8: Erstes Figma-Design .....	17
Abbildung 9: Neues Design der Login-Oberfläche .....	18
Abbildung 10: Codeausschnitt der Navigationsleiste.....	19
Abbildung 11: Design der Navigationsleiste.....	19
Abbildung 12: Reservierungsmanagement .....	20
Abbildung 13: Codebeispiel Filtermethoden.....	21
Abbildung 14: Suchmethode .....	21
Abbildung 15: SQL-Befehl, um Tabelle zu erstellen .....	22
Abbildung 16: Stored Procedure mit Hash-Funktion .....	23
Abbildung 17: TypeHandler für DateOnly .....	23
Abbildung 18: Dummy Repository im Unit Test .....	24

## 9 Quellen- und Literaturverzeichnis

Für Teil I - Unternehmensorganisation:

<https://moodle.css-nsu.de/moodle/course/view.php?id=744> - Schulstoff aus BT-B

<https://www.business-wissen.de/artikel/leitbild-entwickeln-beispiele-und-anleitung/>

<https://gruenderplattform.de/ratgeber/unternehmensziele>

<https://www.business-wissen.de/hb/inhalte-einer-stellenbeschreibung/>

<https://keyperformance.de/corporate-identity>

Für Teil II Softwareentwicklung:

<https://softchris.github.io/pages/dotnet-moq.html#references>

<https://www.mssqltips.com/sqlservertip/4037/storing-passwords-in-a-secure-way-in-a-sql-server-database/>

<https://github.com/MaterialDesignInXAML/MaterialDesignInXamlToolkit>

<https://www.learndapper.com/>

<http://materialdesigninxaml.net/>

<https://stackoverflow.com/> - oft während Entwicklung genutzt, um kleinere Codeprobleme zu lösen