Software Engineering Übung, LVNr: 8_H2

Übungsleiter: Hans Moritsch

Dokument: Design 1 v.1.0

Projekttitel: Orbit Hotel

Projekthomepage:

http://www.unet.univie.ac.at/~a1200993/

Gruppenmitglieder:

MatNr:	Nachname:	Vorname:	e-mail:
1203094	Christl	Korbinian	korbson@gmx.net
1206994	Lazarus	Michael	michaellazarus@hotmail.de
1200993	Lomasow	Veronika	veronika.lomasow@gmx.at
1247136	Chlup	Sebastian	Seb.c21@gmx.at

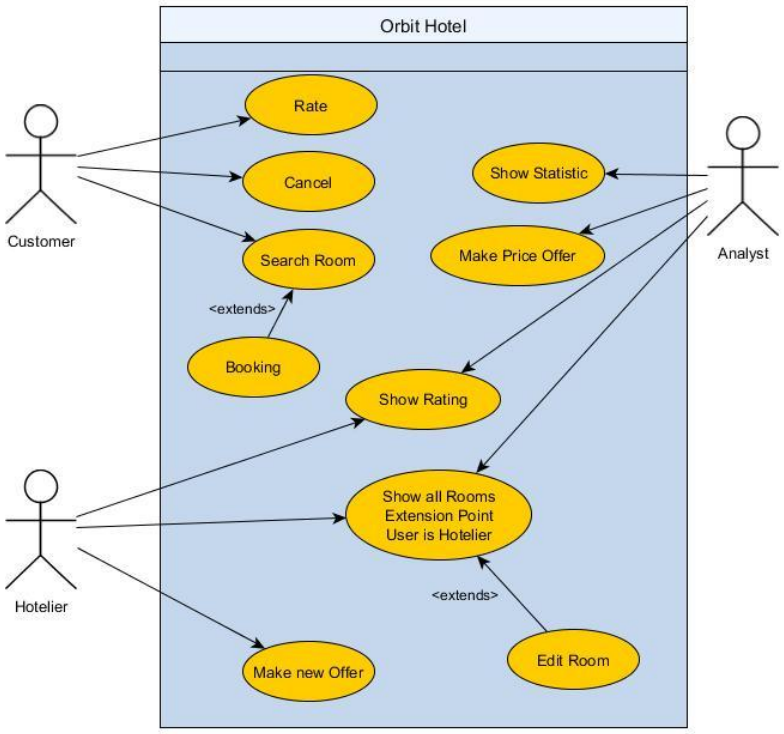
Git Repository:

https://github.com/MichiLazarus/Hotelreservierung

Datum: 23.11.2013

1 Use Cases, Sequenzdiagramme und Use Cases Realization Design

1.1 Use Cases



1.1.1 Use Case 1

- Use Case Name: „Search Room”
- Ziel:
  - Ausgabe der verfügbaren Zimmer, Zimmerpreise und Ausstattung dieser.
- Kategorie:
  - Primär
- Vorbedingung:
  - Der Customer muss zum Ausführen des Use Cases eingeloggt bzw. registriert sein.
- Nachbedingung bei Erfolg:
  - Customer kann entweder ein Zimmer buchen oder eine erneute Zimmerabfrage erstellen.
- Nachbedingung bei Fehlschlag:
  - Customer kann eine erneute Zimmerabfrage starten.
- Beteiligte Akteure:
  - Customer

1.1.2 Use Case 1 Beschreibung

- Auslösendes Ereignis:
  - Klick auf den Suchbutton inkl. Eingabe der Suchkriterien.
- Beschreibung Basisablauf:
  - Customer gibt Suchkriterien wie Preis, Personenanzahl oder Ausstattung ein und klickt auf den Suchbutton. Danach werden treffende Zimmer, welche gerade nicht belegt sind angezeigt. Weiters werden Informationen über das jeweilige Zimmer angegeben. Der Customer hat darauf die Möglichkeit das jeweilige Zimmer zu buchen.

1.1.3 Use Case 2

- Use Case Name:
  - "Rate Booking"
- Ziel:
  - Bewertung der Zimmer.
- Kategorie:
  - Primär
- Vorbedingung:
  - Customer muss registriert und eingeloggt sein.
- Nachbedingung bei Erfolg:
  - Customer kann Wünsche, Anregungen, Beschwerden mitteilen.
- Nachbedingung bei Fehlschlag:
  - Erneuter Versuch Bewertung abzugeben.
- Beteiligte Akteure:
  - Customer

1.1.4 Use Case 2 Beschreibung

- Auslösendes Ereignis:
  - Nach eingeben, der Buchungsnummer und des Bewertungstextes kann man auf "Rate" klicken.
- Beschreibung Basisablauf:
  - Customer gibt Buchungsnummer ein und verfasst einen Bewertungstext. Im Anschluss an den Klick auf den "Rate" Button wird die Bewertung abgeschickt und in ein Textdokument des zugehörigen Zimmers gespeichert.

1.1.5 Use Case 3

- Use Case Name:
  - "Cancel Booking"
- Ziel:
  - Stornierung eines zuvor gebuchten Zimmers.

- Kategorie:
    - Primär
- Vorbedingung:
    - Customer muss registriert und eingeloggt sein und ein Zimmer gebucht haben.
- Nachbedingung bei Erfolg:
    - Zimmer wird storniert.
- Nachbedingung bei Fehlschlag:
    - Zimmer wurde nicht storniert und erneuter Versuch muss gestartet werden.
- Beteiligte Akteure:
    - Customer

1.1.6 Use Case 3 Beschreibung

- Auslösendes Ereignis:
    - Customer gibt Buchungsnummer ein und drückt den "Cancel" Button.
- Beschreibung Basisablauf:
    - Customer Buchungsnummer des zu stornierenden Zimmers ein und klickt den "Cancel" Button. Daraufhin wird bei gültiger Eingabe die Buchung storniert. Zimmerstatus wird auf "Verfügbar" gesetzt.

1.1.7 Use Case 4 Beschreibung

- Use Case Name:
    - "Booking"
- Ziel:
    - Buchung eines Zimmers
- Kategorie:
    - Optional
- Vorbedingung:
    - Der Customer muss registriert und eingeloggt sein. Das Zimmer muss gesucht und ausgewählt werden.
- Nachbedingung bei Erfolg:
    - Das Zimmer ist gebucht. Buchungsbestätigung mit Buchungsnummer erscheint.
- Nachbedingung bei Fehlschlag:
    - Zimmer wurde nicht gebucht und muss erneut gebucht werden.
- Beteiligte Akteure:
    - Customer

1.1.8 Use Case 4 Beschreibung

- Auslösendes Ereignis:
    - Nach Eingabe der Suchkriterien und klick auf den "Search" Button, wird Liste mit allen zur Verfügung stehenden Räumen, welche den

Suchkriterien entsprechen ausgegeben. Der Customer wählt eines dieser Zimmer und kommt anschließend in die Buchungsmaske wo die Daten zur Buchung bestimmt werden.

- Beschreibung Basisablauf:
    - In der Buchungsmaske ist es essenziell, dass der User alle Informationen wie Datum, Zahlungsart und Personenanzahl eingibt. Im Anschluss klickt er auf den "Buchen" Button, wodurch eine Buchungsseite mit Buchungsnummer, Gesamtpreis und eingegebenen Informationen erscheint. Klickt der Customer nun auf "Buchen", wird der Status des Zimmers auf "belegt" geändert.

1.1.9 Use Case 5

- Use Case Name:
    - "Show Rating"
- Ziel:
    - Einsehen der Bewertungen der Customer.
- Kategorie:
    - Primär
- Vorbedingung:
    - Hotelier oder Analyst muss eingeloggt sein und Bewertungen müssen vorhanden sein.
- Nachbedingung bei Erfolg:
    - Hotelier oder Analyst kann Bewertungen eines bestimmten Zimmers einsehen.
- Nachbedingung bei Fehlschlag:
    - Keine Bewertung vorhanden oder neuer Versuch Bewertung einzusehen.
- Beteiligte Akteure:
    - Hotelier, Analyst

1.1.10 Use Case 5 Beschreibung

- Auslösendes Ereignis:
    - Klick auf "Show" Button bei Show Rating nach Eingabe der Raumnummer.
- Beschreibung Basisablauf:
    - Hotelier oder Analyst gibt im Feld "Show Rating" eine bestimmte Raumnummer ein und klickt auf den "Show" Button. Im Anschluss werden die Bewertungen für dieses Zimmer angezeigt.

1.1.11 Use Case 6

- Use Case Name:
  - "Alle Zimmer anzeigen"
- Ziel:
  - Ausgabe aller Zimmer inklusive Preise und Ausstattung dieser.
- Kategorie:
  - Primär
- Vorbedingung:
  - User muss eingeloggt sein.
- Nachbedingung bei Erfolg:
  - Dem User werden alle Zimmer angezeigt.
- Nachbedingung bei Fehlschlag:
  - Erneuter Versuch Zimmer einzusehen.
- Beteiligte Akteure:
  - Hotelier, Analyst

1.1.12 Use Case 6 Beschreibung

- Auslösendes Ereignis:
  - Klick auf den "Show" Button im Feld "Show all Rooms"
- Beschreibung Basisablauf:
  - User klickt auf den "Show" Button im Feld "Show all Rooms". Er bekommt eine Tabelle zurückgeliefert mit dem Inhalt "Rommnumber", "Show", "Number of Persons", "Status", "Edit". Er hat daraufhin noch die Möglichkeit die Zimmer zu editieren.

1.1.13 Use Case 7

- Use Case Name:
  - "Zimmer Editieren"
- Ziel:
  - Editieren der "Roomnumber", "Price", "Equipment", "Number of Persons", "Status", "Rating"(unangebrachte Kommentare entfernen) eines Zimmers.
- Kategorie:
  - Optional
- Vorbedingung:
  - Hotelier muss eingeloggt sein.
- Nachbedingung bei Erfolg:
  - Zimmer wurde editiert und weiter Zimmer können editiert werden.
- Nachbedingung bei Fehlschlag:
  - Erneuter Versuch Zimmer zu editieren.
- Beteiligte Akteure:
  - Hotelier

1.1.14 Use Case 7 Beschreibung

- Auslösendes Ereignis:
  - Klick auf den "Edit" Button in der Maske "Show all Rooms".
- Beschreibung Basisablauf:
  - Über das Interface, welches Alle Räume anzeigt, kann der Hotelier auf den "Edit" Button eines bestimmten Zimmers klicken, wodurch er in die Maske zum Editieren des Zimmers gelangt und kann dadurch Roomnumber", "Price", "Equipment", "Number of Persons", "Status", "Rating"(unangebrachte Kommentare entfernen) editieren. Zusätzlich wird ihm der Preisvorschlag des Analysten angezeigt. Die veränderten Informationen werden gespeichert.

1.1.15 Use Case 8

- Use Case Name:
  - "Make new Offer"
- Ziel:
  - Erstellen eines neues Angebotes
- Vorbedingung:
  - Hotelier muss eingeloggt sein.
- Nachbedingung bei Erfolg:
  - Angebot wird erfolgreich erstellt.
- Nachbedingung bei Fehlschlag:
  - Angebot konnte nicht erstellt werden, erneuter Versuch.
- Beteiligte Akteure:
  - Hotelier

1.1.16 Use Case 8 Beschreibung

- Auslösendes Ereignis:
  - Beim Feld "New Offer", "Finish" Button klicken.
- Beschreibung Basisablauf:
  - Hotelier gibt "Rommnumber", "Price", “Equipment“ und "Number of Persons" ein und klickt auf den "Finish" Button. Das Angebot wurde erstellt.

1.1.17 Use Case 9

- Use Case Name:
  - „Show Statistic“
- Ziel:
  - Einblick in öffentliche Statistik.
- Kategorie:

    - Primär
- Vorbedingung:
    - Login, Vorhandensein von Reservierungs- und Kundendaten
- Nachbedingung bei Erfolg:
    - Statistik vorhanden
- Beteiligte Akteure:
    - Analyst

1.1.18 Use Case 9 Beschreibung

- Auslösendes Ereignis:
    - Eingeben der Statistiknummer und Klick auf „Show“ Button im Statistikfeld.
    - Beschreibung Basisablauf: Analyst gibt Nummer der gewünschten Statistik ein und klickt auf den „Show“ Button. Falls abrufbare Daten vorhanden sind, wird eine passende Statistik zu Beliebtheit des Angebotes oder saisonalen Reservierungen autogeneriert.

1.1.19 Use Case 10

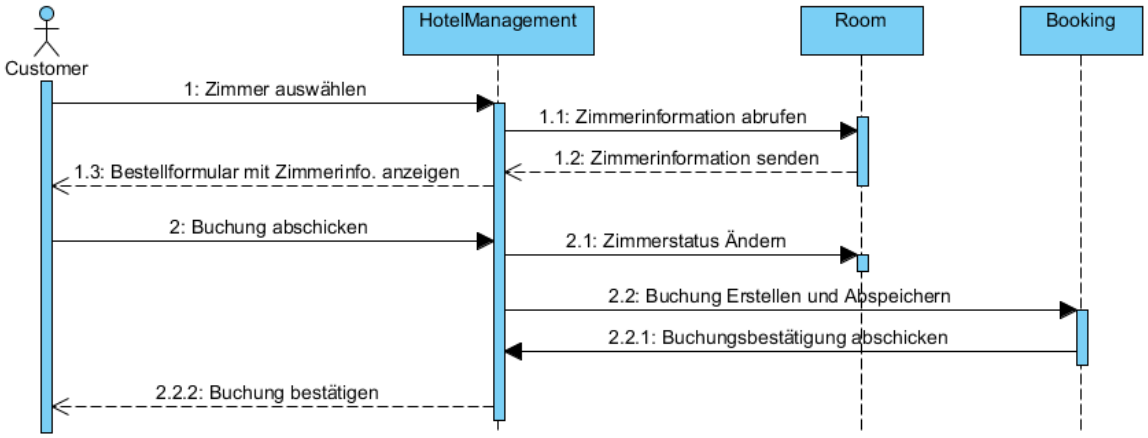
- Use Case Name:
    - "Make Price Offer”
- Ziel:
    - Erstellen eines Preisvorschlages
- Kategorie:
    - Primär
- Vorbedingung:
    - Analyst muss eingeloggt sein
- Nachbedingung bei Erfolg:
    - Preisvorschlag wurde erstellt
- Nachbedingung bei Fehlschlag:
    - Erneuter Versuch einen Preisvorschlag zu erstellen.
- Beteiligte Akteure:
    - Analyst

1.1.20 Use Case 10 Beschreibung

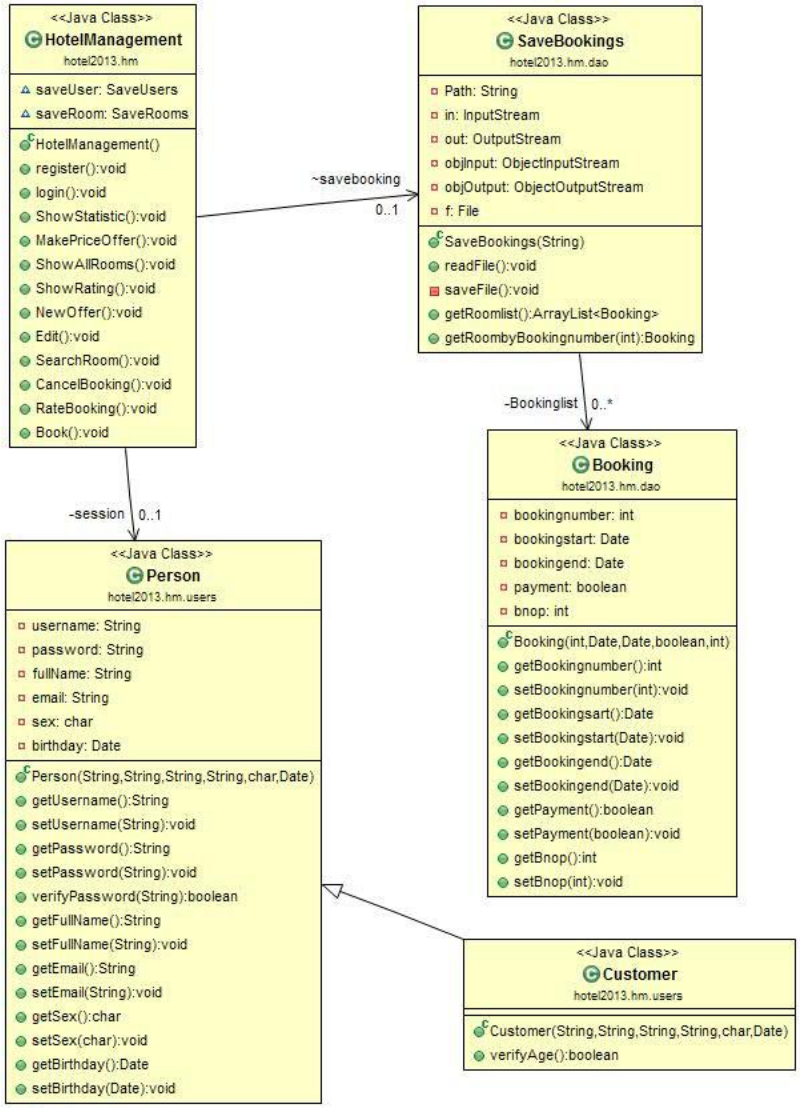
- Auslösendes Ereignis:
    - Klick auf den "Offer" Button.
- Beschreibung Basisablauf:
    - Der Analyst gibt "Roomnumber" und "Price", den er für dieses Zimmer als angebracht empfindet ein und erstellt so einen Preisvorschlag.

1.2 Use Case Realization Design

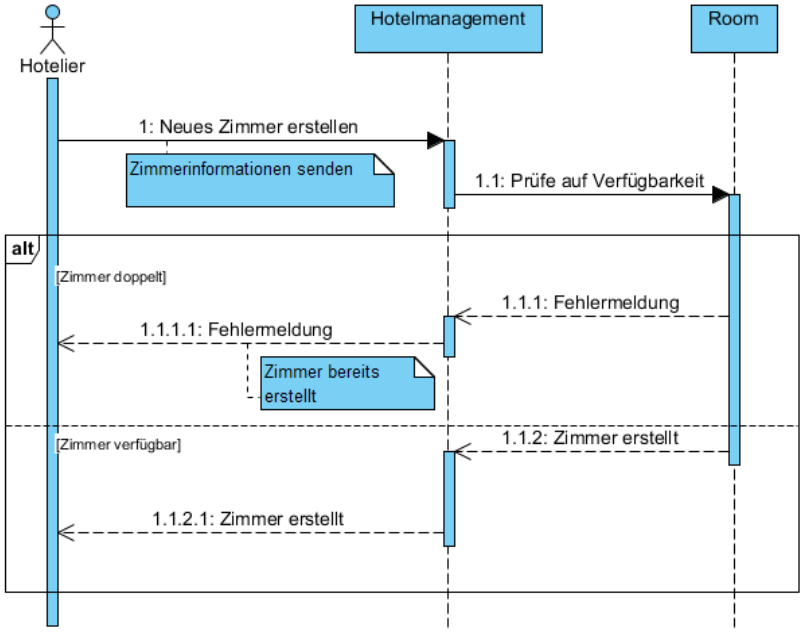
1.2.1 Sequenzdiagramm 1 (UseCase „Booking“)



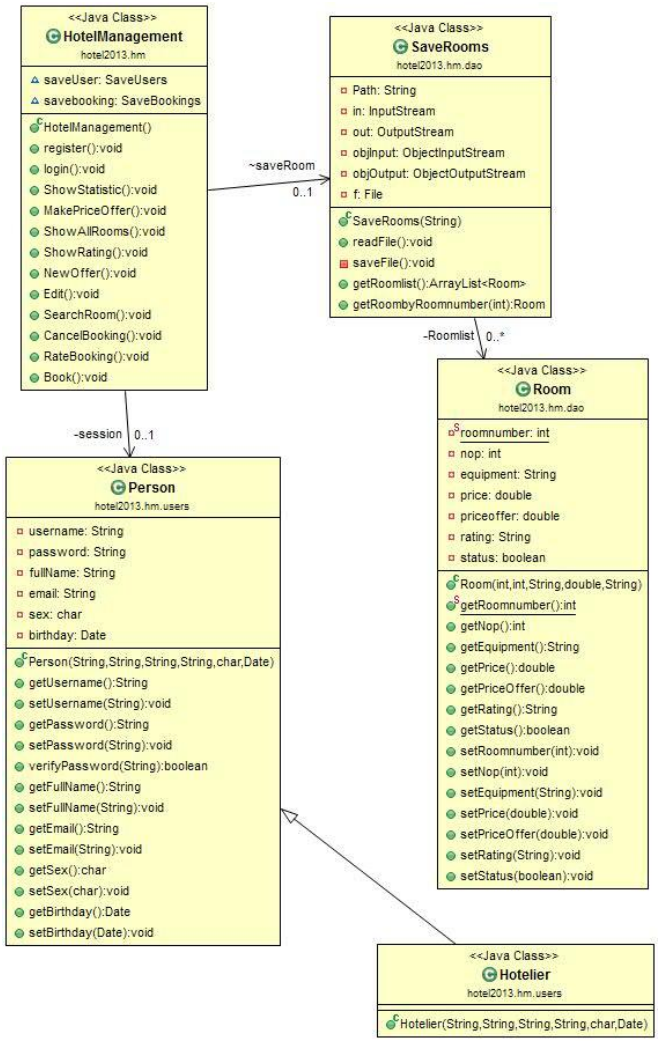
1.2.2 Klassendiagramm 1 (Use Case „Booking“)



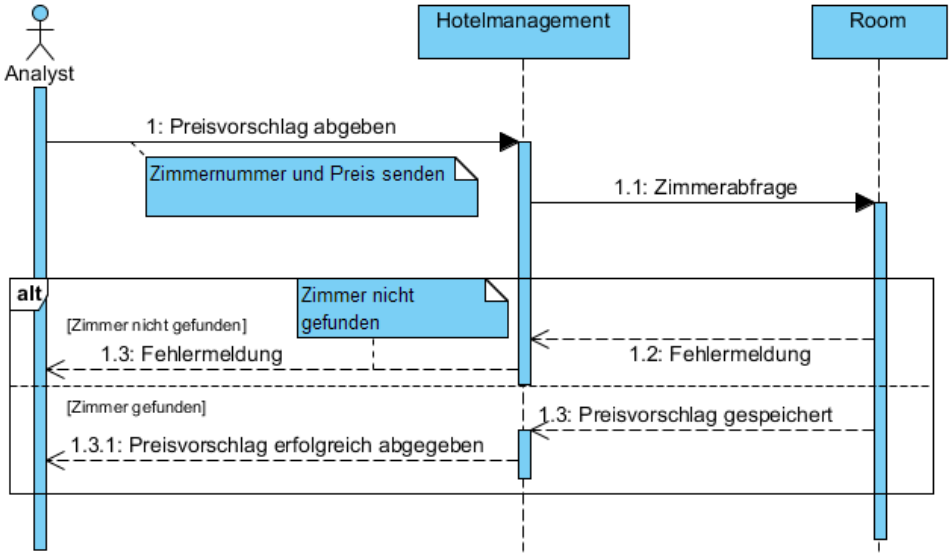
1.2.3 Sequenzdiagramm 2 (Use Case Make new Offer)



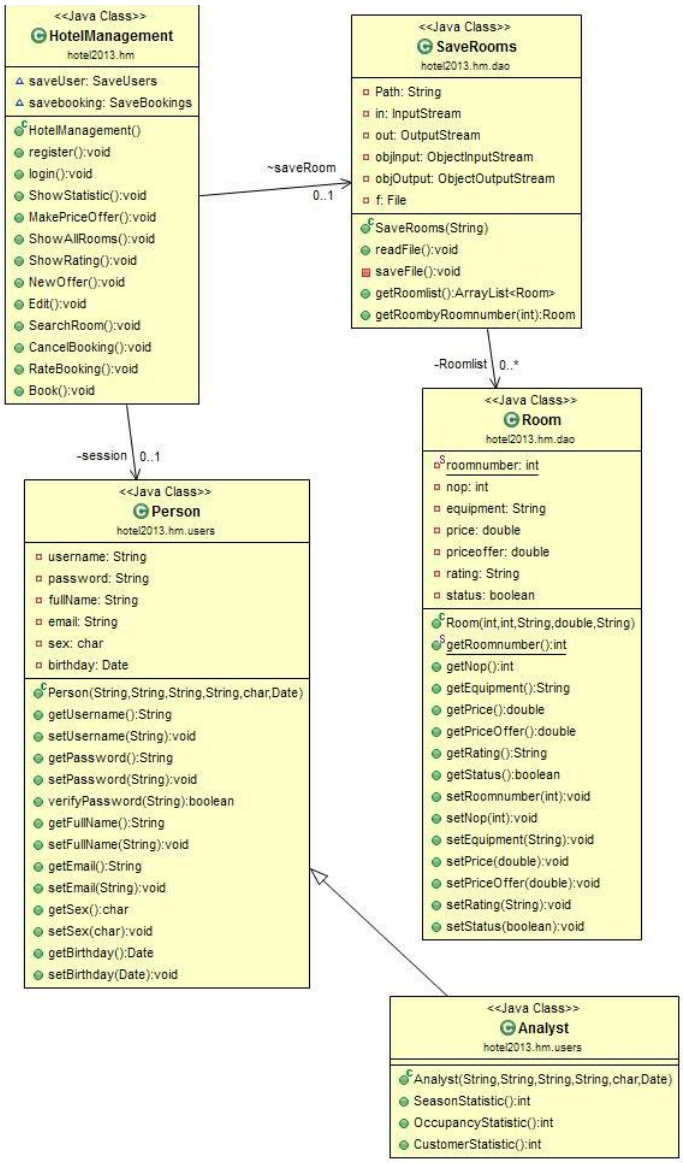
1.2.4 Klassendiagramm 2 (Use Case Make new Offer)



1.2.5 Sequenzdiagramm 3 (Use Case Make Price Offer)



1.2.6 Klassendiagramm 3 (Use Case Make Price Offer)



1.3 Use Case Realization Beschreibung

1.3.1 Beschreibung 1

Sequenzdiagramm 1 (Use Case „Booking“)

Am Beginn der Interaktion „Booking“ sendet der Benutzer „Customer“ eine Nachricht an die Klasse „HotelManagement“, um ein Zimmer auszuwählen. Die Klasse „HotelManagement“ will daraufhin die Zimmerinformationen von der Klasse „Room“ abrufen. Diese antwortet mit den gewünschten Zimmerinformationen. „HotelManagement“ sendet eine Antwort-Nachricht an „Customer“ mit dem Bestellformular. „Customer“ schickt darauf eine Nachricht mit der Buchung, die alle benötigten und notwendigen Buchungsinformationen des „Customer“ enthält. „Hotelmanagement“ informiert parallel „Room“ den Zimmerstatus zu ändern und die Klasse „Booking“ die Buchung zu erstellen und abzuspeichern. Als Antwort-Nachricht bekommt er von „Booking“ die Buchungsbestätigung. Die Interaktion endet mit der Buchungsbestätigung vom „HotelManagement“ an den „Customer“.

Klassendiagramm 1 (Use Case „Booking“)

Die Person Hotelier greift mittels der Klasse Hotelmanagement auf die public Funktion „New Offer “ zu. Diese Methode erstellt einen neues Objekt der Klasse „Room“, die Attribute dieser Klasse sind alle private auf diese Attribute kann jedoch mittels den public Getters und Setters zugegriffen werden und der Konstruktor erstellt nach Eingabe aller notwendiger Parameter einen neuen Raum. Die Klasse „SaveRooms“ ist dazu da die ArrayList aller Räume auszulesen und zu speichern. So fügt sie nun den neuen Raum in ihre private ArrayList hinzu und mit der public Methode „SaveRooms“ speichert sie erneut alle Räume in das File, dessen Dateipfad aus dem privaten String „Path“ ausgelesen wird.

1.3.2 Beschreibung 2

Klassendiagramm 2 (Use Case „Make new offer“)

Die Person Hotelier greift mittels der Klasse Hotelmanagement auf die public Methode „NewOffer “ zu. Diese Methode erstellt einen neues Objekt der Klasse „Room“, die Attribute dieser Klasse sind alle private auf diese Attribute kann jedoch mittels den public Getters und Setters zugegriffen werden und der Konstruktor erstellt nach Eingabe aller notwendiger Parameter einen neuen Raum. Die Klasse „SaveRooms“ ist dazu da die ArrayList aller Räume auszulesen und zu speichern. So fügt sie nun den neuen Raum in ihre private ArrayList hinzu und mit der public Methode „SaveRooms“ speichert sie erneut

alle Räume in das File, dessen Dateipfad aus dem privaten String „Path“ ausgelesen wird.

Sequenzdiagramm 2 (Use Case „Make new offer“)

Die Interaktion „Make new offer“ beginnt mit einer neuen Zimmererstellung des Benutzers „Hotelier“, der eine Nachricht an die Klasse „HotelManagement“ mit allen Zimmerinforamtionen sendet. Daraufhin sendet „Hotelmanagement“ eine Nachricht an die Klasse „Room“, die die Verfügbarkeit des Zimmers überprüfen soll. Falls dann ein Zimmer doppelt erstellt wurde, sendet „Room“ eine Fehlermeldung an „HotelManagement“, der dann dem „Hotelier“ ausgibt, dass das Zimmer bereits erstellt wurde. Wenn ein Zimmer aber verfügbar ist, teilt „Room“ dem „HotelManagement“ mit, dass ein neues Zimmer erstellt wurde, welcher dann dem „Hotelier“ eine Antwort-Nachricht zusendet, in der er genau dasselbe nochmal dem „Hotelier“ mitteilt und damit die Interaktion beendet.

1.3.3 Beschreibung 3

Sequenzdiagramm 3 (Use Case „Make price offer“)

Die Person Hotelier greift mittels der Klasse Hotelmanagement auf die public Methode „MakePriceOffer “ zu. Diese Methode ruft die Public Methode „getRoombyRoomnumber“ der Klasse „SaveRooms“ auf. So wird aus deren RoomsArrayList ein bestimmtes Objekt der Klasse „Room“ ausgelesen. Diesem Objekt wird nun ein weiterer privater Parameter übergeben „PriceOffer“ dieser wird mittels der public „setPriceOffer“ Methode dieser Parameter hinzugefügt. Danach speichert die Klasse „SaveRooms“ das neue Raumobjekt in die Arraylist mittels der public Methode „SaveRooms“. Diese Arraylist wird ebenfalls wieder in das dazugehörige File gespeichert dessen Pfad mittels privatem Attribut „Path“ festgelegt ist.

Klassendiagramm 3 (Use Case „Make price offer“)

Die Person Hotelier greift mittels der Klasse Hotelmanagement auf die public Methode „MakePriceOffer“ zu. Diese Methode ruft die Public Methode „getRoombyRoomnumber“ der Klasse „SaveRooms“ auf. So wird aus deren RoomsArrayList ein bestimmtes Objekt der Klasse „Room“ ausgelesen. Diesem Objekt wird nun ein weiterer privater Parameter übergeben, „PriceOffer“. Dieser wird mittels der public „setPriceOffer“ Methode diesem Objekt hinzugefügt. Danach speichert die Klasse „SaveRooms“ das neue Raumobjekt in die Arraylist mittels der public Methode „SaveRooms“. Diese Arraylist wird ebenfalls wieder in das dazugehörige File gespeichert dessen Pfad mittels privaten Attribut „Path“ festgelegt ist.

2 Klassendesign

2.1 Klassenbeschreibungen

2.1.1 Klasse 1

Person:

Person ist die Upperclass von Analyst, Hotelier und Customer. Sie beinhaltet alle notwendigen privaten Attribute, um sich einzuloggen oder zu registrieren. Des Weiteren sind hier auch alle notwendigen public get- und set-Methoden implementiert, um auf die Variablen zugreifen zu können bzw. sie zu verändern. Dazu kommt noch die Methode „verifyPassword“, welche zur Überprüfung der Gültigkeit der Eingabe verwendet wird. Diese Klasse wird benötigt, damit alle unterschiedlichen Subclassen von ihr erben können.

2.1.2 Klasse 2

Customer:
Customer ist eine SubClass der Klasse Person. Dies bedeutet, dass sie sämtliche Methoden und Variablen dieser erbt. Mit dieser Klasse wird der Otto-Normalverbraucher dargestellt. Zusätzlich enthält er noch die public Methode „verifyAge“, welche sicherstellt, dass der sich registrierende Benutzer volljährig ist.

2.1.3 Klasse 3

Analyst:
Customer ist eine SubClass der Klasse Person. Dies bedeutet, dass sie sämtliche Methoden und Variablen dieser erbt. Diese Klasse repräsentiert den Mitarbeiter eines Unternehmens, welches sich auf Analysen für Hotelstatistiken spezialisiert hat. Um die verschiedenen Statistiken zu erstellen, besitzt er die public Methoden „SeasonStatistic“, „OccupancyStatistic“ und „CustomerStatistic“. Diese greifen auf die bereits gespeicherten Daten zu und autogenerieren je nach Anforderung die gewünschte Statistik.

2.2.4 Klasse 4

Hotelier:
Hotelier ist eine SubClass der Klasse Person. Dies bedeutet, dass sie sämtliche Methoden und Variablen dieser erbt. Diese Klasse repräsentiert den SuperUser, mit dem Zimmerangebote erstellt bzw. bearbeitet werden können. Der Hotelier wird im Usermanagement benötigt, um Rechte für Analyst und Customer einzuschränken.

2.2.5 Klasse 5

SaveUsers:
Diese Klasse dient der Speicherung, sowie dem Auslesen der registrierten und angelegten Personen. Sie beinhaltet die dazu benötigten In- und Outputstreams. Auf diese wird mittels der public Methoden „readFile“, auslesen der Arraylist in der alle UserObjekte gespeichert sind , „saveFile“ zur Speicherung der ObjektArraylist in das File , „getUserList“ zum Aufrufen der ausgelesenen ArrayList und „getUserbyUsername“ um ein bestimmtes Object aus der Liste zu suchen.

2.2.6 Klasse 6

Booking:
Hierbei wird das Objekt einer Buchung erzeugt. Diese enthält die privaten Variablen „bookingnumber“, „bookingstart“(ab wann eine Buchung gültig ist), „bookingend“(wann die Buchung endet), „payment“ (Art der Bezahlung, mit Kreditkarte oder in Bar)und „bnop“(booking number of persons). Zu diesen Variablen hinzu, kommen die public get- und set- Methoden, um auf die Variablen zugreifen zu können bzw. diese zu ändern.

2.2.7 Klasse 7

SaveBooking:
Diese Klasse wird dazu verwendet, dass erfolgte Buchungen in ein File gespeichert werden und wieder ausgelesen werden können. Auf diese wird mittels der public Methoden „readFile“, auslesen der Arraylist in der alle BuchungsObjekte gespeichert sind , „saveFile“ zur Speicherung der ObjektArraylist in das File , „getBookingList“ zum Aufrufen der ausgelesenen ArrayList und „getBookingbyBookingnumber“ um ein bestimmtes Object aus der Liste zu suchen.

2.2.8 Klasse 8

Room:
Diese Klasse repräsentiert das anzulegende Objekt Raum. Sie besteht sowohl aus den private Variablen „roomnumber“, „nop“(number of persons), „equipment“(besondere zusätzliche Ausstattung), „price“, „priceoffer“(vom Analysten vorgeschlagener Preis), “rating”(vom Kunden erstellte Bewertung der Zimmer) und “status”(ist Zimmer frei oder belegt), als auch den benötigten public get- und set- Methoden.

2.2.9 Klasse 9

SaveRooms:
Diese Klasse wird dazu verwendet, um erzeugte Objekte vom Typ Room in ein File zu speichern und diese wieder auszulesen. Auf diese wird mittels der public Methoden „readFile“ aus der Arraylist zugegriffen, in der alle RoomObjekte gespeichert sind , „saveFile“ zur Speicherung der ObjektArraylist in das File , „getRoomList“ zum Aufrufen der ausgelesenen ArrayList und „getRoombyRoomnumber“ um ein bestimmtes Objekt aus der Liste zu suchen.

2.2.10 Klasse 10

Hotelmanagement:
Stellt die Verwaltungsklasse unseres Projektes dar. Von dieser Klasse ausgehend wird auf alle anderen zugegriffen. Zusätzlich sind die ihr zugewiesenen Methoden nur für die spezifischen Subklassen von Person ausführbar. Für den Costumer : „register“, „login“, „searchroom“, „Book“, „CancelBooking“, „RateBooking“. Dem Hotelier stehen die public Methoden: „ShowAllRooms“, „ShowRating“, „NewOffer“ und „Edit“ zur Verfügung. Schlussendlich hat der Analyst die Möglichkeit auf: „ShowAllRooms“, „ShowStatistic“ und „MakePriceOffer“ zuzugreifen. Diese Klasse wird benötigt um alle notwendigen Methoden zu verwalten.

2.2 Wichtige Designentscheidungen

Die Klasse Person wurde gewählt, um eine einfache Erweiterbarkeit des Hotelreservierungssystems zu ermöglichen. In dieser Klasse wurden die wichtigsten Attribute und Methoden definiert, welche in jeder Subklasse benötigt werden.

Der Hotelier wird im Hotelmanagement wichtig, da dort mittels „instanceof“ unterschieden wird, welche Methoden und Aktionen von den unterschiedlichen Usern ausgeführt werden dürfen. Ansonsten hat der Hotelier keine zusätzlichen Funktionen.

Um eine Vereinfachung und Übersichtlichkeit der Datenspeicherung zu ermöglichen, wurden 3 unterschiedliche Klassen gewählt, welche in 3 unterschiedliche Dateien serialisieren.

Das Hotelmanagement agiert als zentrale Verbindung zwischen den verschiedenen Klassen des Systems. Über das Hotelmanagement wird auf alle wichtigen Methoden zugegriffen und weiters werden Berechtigungen geprüft.

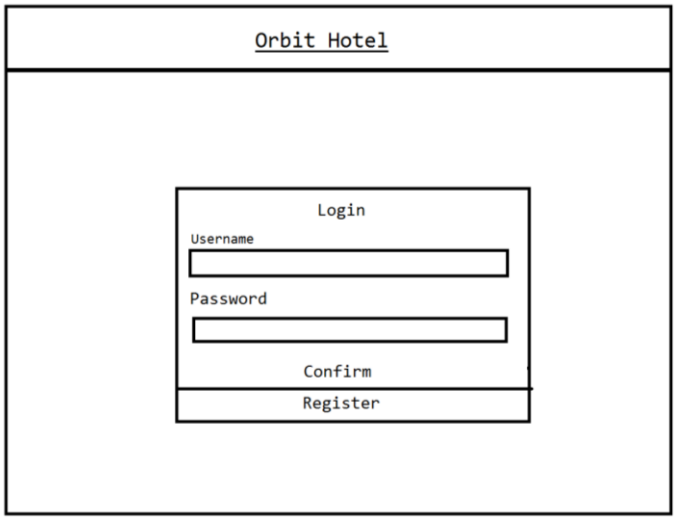
2.3 Klassendiagramm

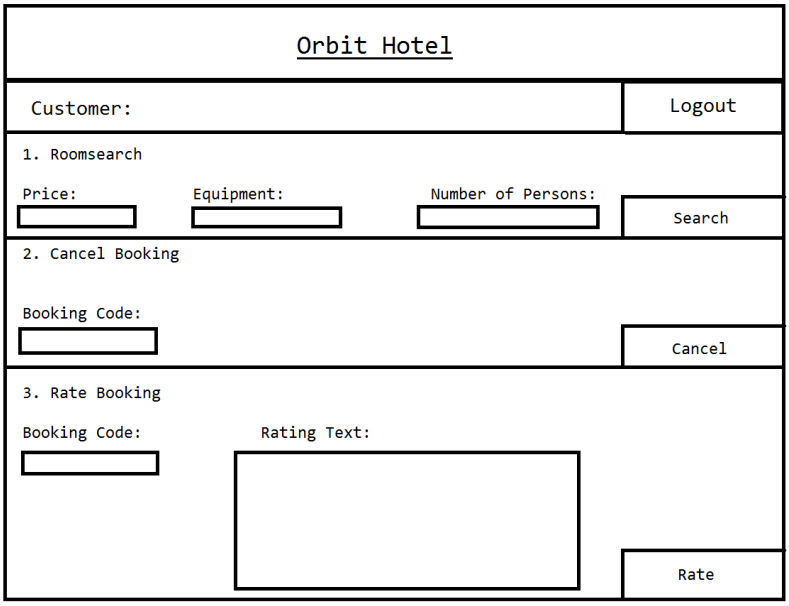
2.4 Paketstruktur

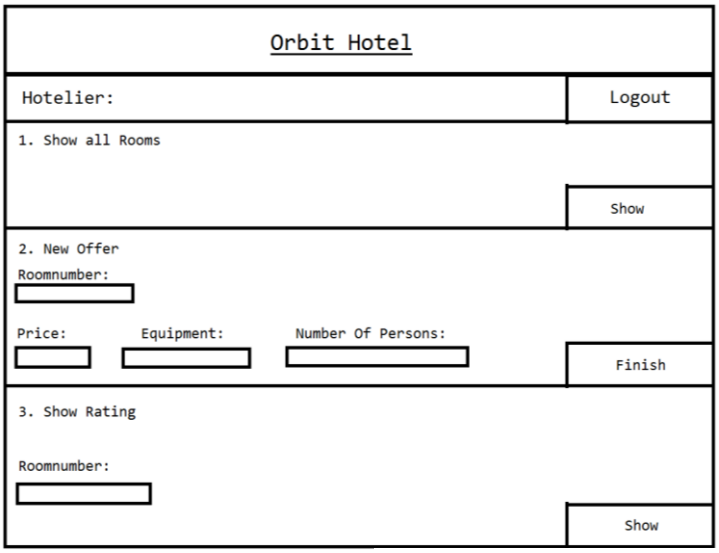
Um die Übersicht über unser Projekt zu behalten haben wir uns dazu entschlossen es in vier verschiedene Pakete zu unterteilen. Diese Pakete beinhalten jeweils verschiedene Arten von Klassen. Die Klassen eines Paketes interagieren untereinander und zueinander. In unserem Oberpaket „hotel2013.hm“ legen wir die Klasse Hotelmanagement an, da in dieser Klasse die wichtigsten Operationen durchgeführt werden, welche alle auf die Klassen der Unterpakete zugreifen. Unser erstes Unterpaket „hotel2013.hm.dao“ beinhaltet die Klassen zur Speicherung bzw. zum Auslesen der Daten „SaveRooms“, „SaveUsers“ und „SaveBookings“. In dem Paket „hotel2013.hm.data“ sind folgende Klassen enthalten: „Room“ und „Booking“ , welche zu verwaltende Objekte des Hotels darstellen und welche zur Statisktikerstellung notwendig sind. Das letzte Subpaket enthält die User, deren Oberklasse die Klasse „Person“ repräsentiert. Die erbenden Klassen lauten „Customer“, „Hotelier“ und „Analyst“.

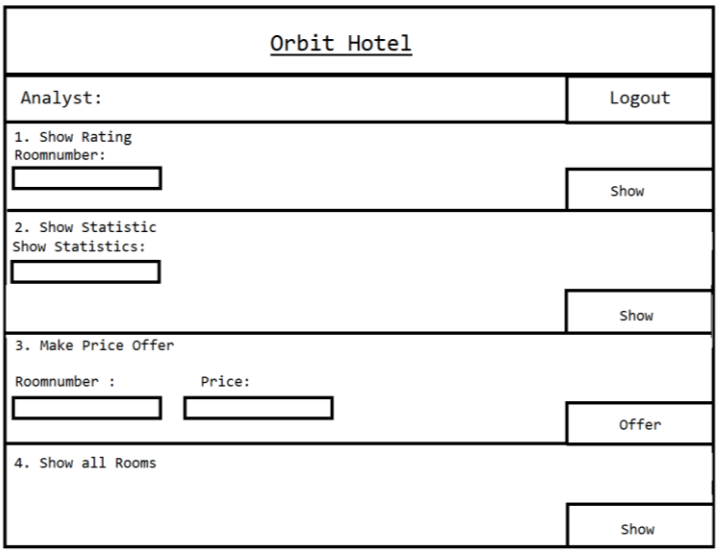
User Interface Beschreibung

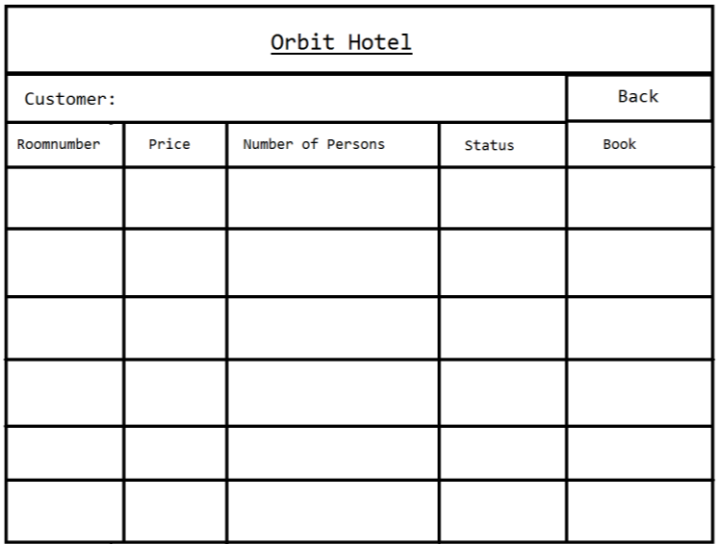
2.5 GUI Screenshot / User Interface Skizzen

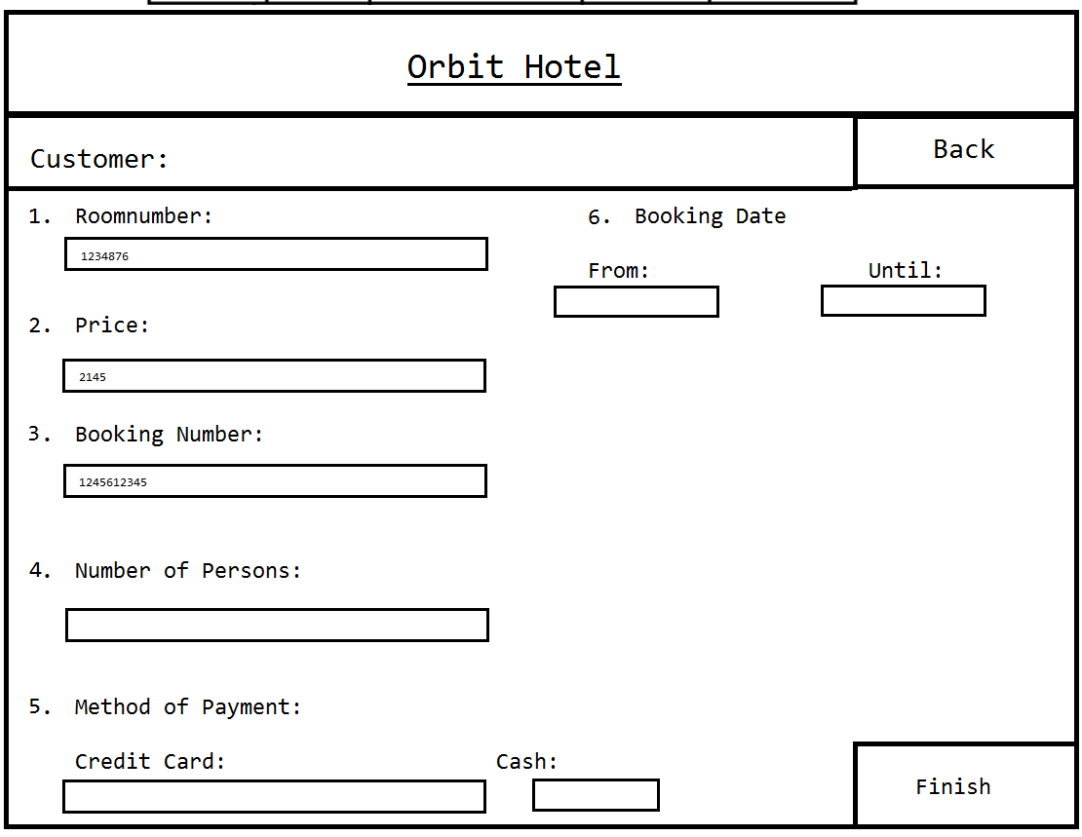


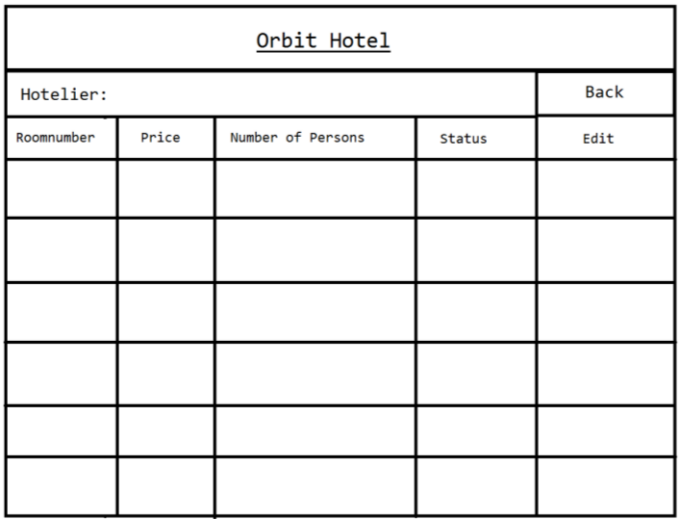


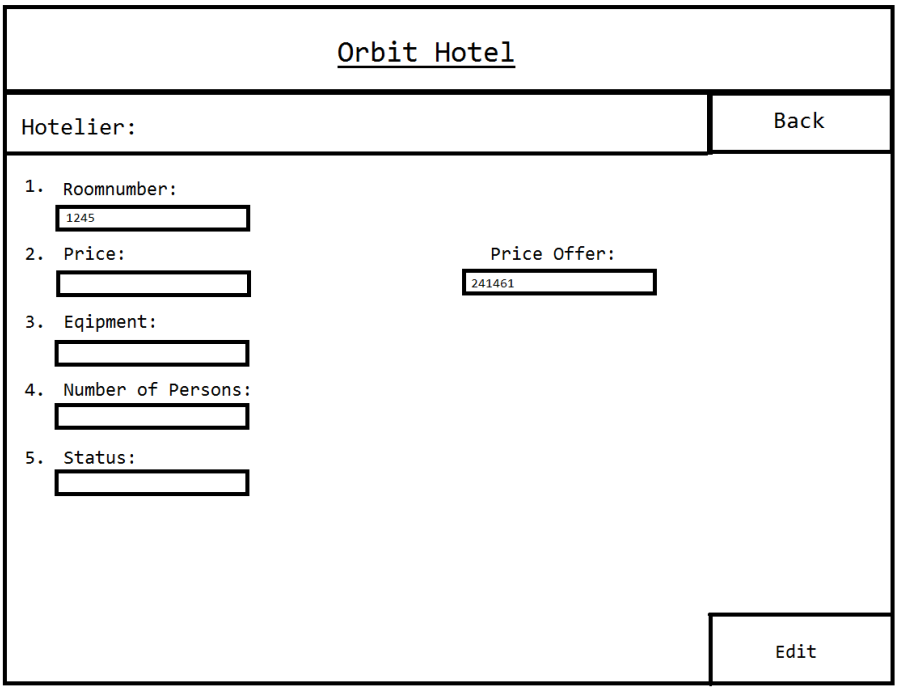

Orbit Hotel
Hotelier:
Logout
1. Show all Rooms
Show
2. New Offer
Roomnumber:
Price:
Equipment:
Number Of Persons:
Finish
3. Show Rating
Roomnumber:
Show

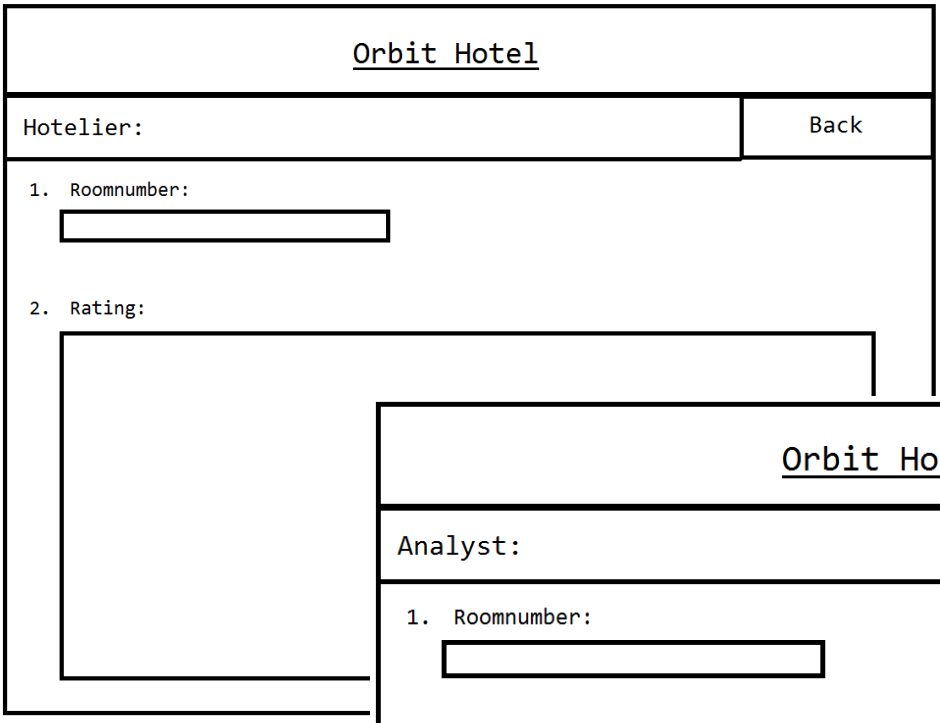

Orbit Hotel
Analyst:
Logout
1. Show Rating
Roomnumber:
Show
2. Show Statistic
Show Statistics:
Show
3. Make Price Offer
Roomnumber :
Price:
Offer
4. Show all Rooms
Show

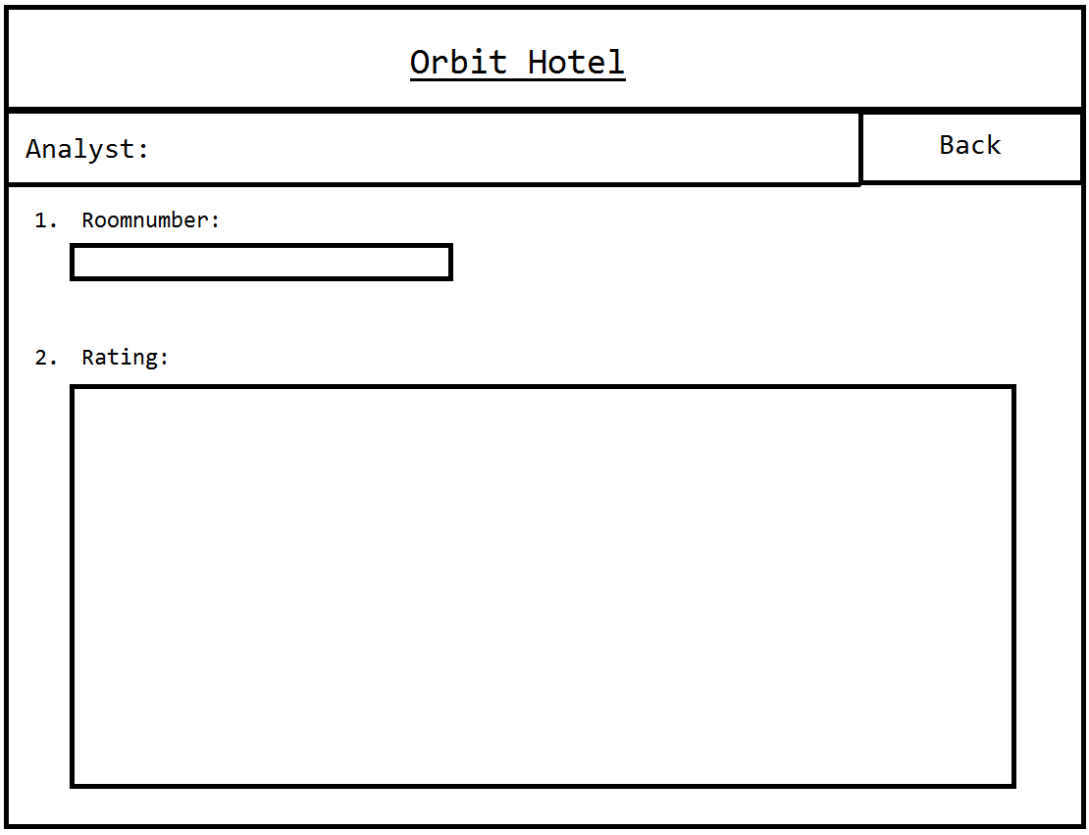

Orbit Hotel
Customer:
Back
Roomnumber
Price
Number of Persons
Status
Book


Orbit Hotel
Customer:
Back
1. Roomnumber:
1234876
2. Price:
2145
3. Booking Number:
1245612345
4. Number of Persons:
5. Method of Payment:
Credit Card:
Cash:
6. Booking Date
From:
Until:
Finish


Orbit Hotel
Hotelier:
Back
Roomnumber
Price
Number of Persons
Status
Edit


Orbit Hotel
Hotelier:
Back
1. Roomnumber:
1245
2. Price:
Price Offer:
241461
3. Eqipment:
4. Number of Persons:
5. Status:
Edit


Orbit Hotel
Hotelier:
Back
1. Roomnumber:
2. Rating:


Orbit Hotel
Analyst:
Back
1. Roomnumber:
2. Rating:

Orbit Hotel

Analyst: | Back

Roomnumber	Price	Number of Persons	Status

Orbit Hotel

Analyst: | Back

1. Statistic:

	Jan	Feb	Mar	Apr	May	June	July	Aug	Sep	Oct	Nov	Dec
Bookings												
Cancelations												
Customer Number												

Average Duration of Stay:

Orbit Hotel

Analyst: | Back

1. Statistic:

Roomnumber	Occupancy
Total:	

Orbit Hotel

Register: | Back

Please Enter your register Data:

1. Username:

2. Password:

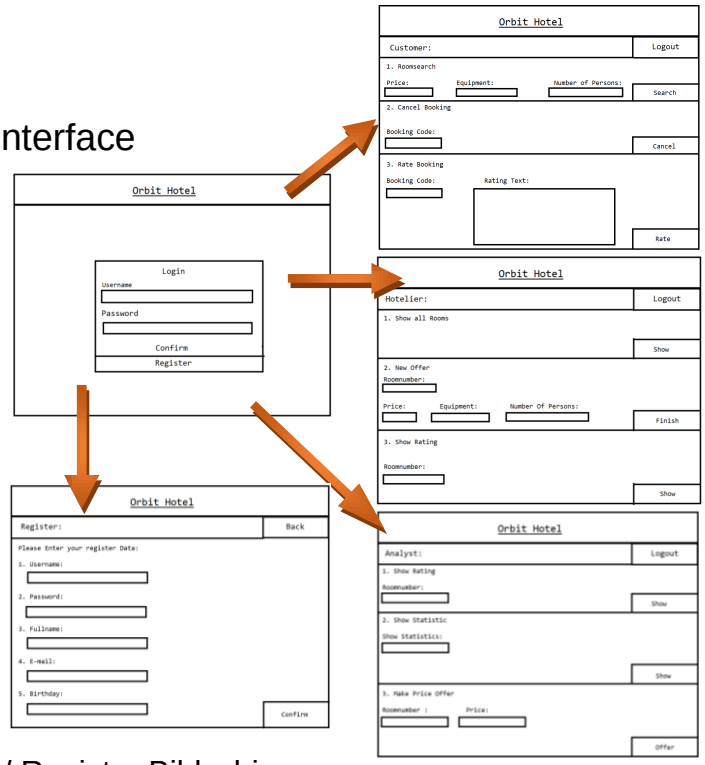
3. Fullname:

4. E-mail:

5. Birthday:

Confirm

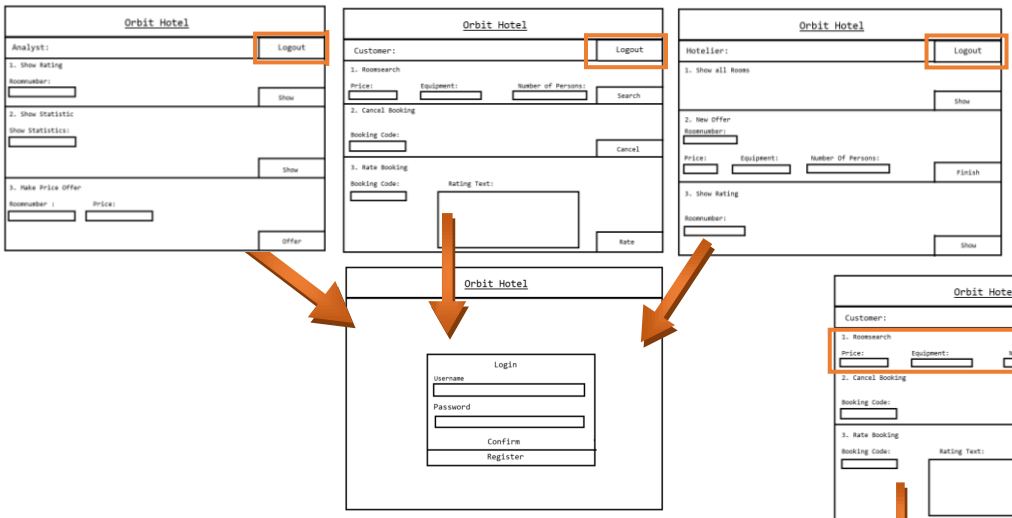
2.6 Beschreibung des GUI / User Interface



Login / Registrierung:

Zu Beginn jedes Aufrufs der Orbit Hotel Startseite steht jedem Nutzer ein Login bzw. eine Registrierung bevor. Dabei muss der User seinen Username und Password eingeben und anschließend auf „Confirm“ drücken.
Ist der User nicht registriert, kann er sich mittels „Register“ registrieren.

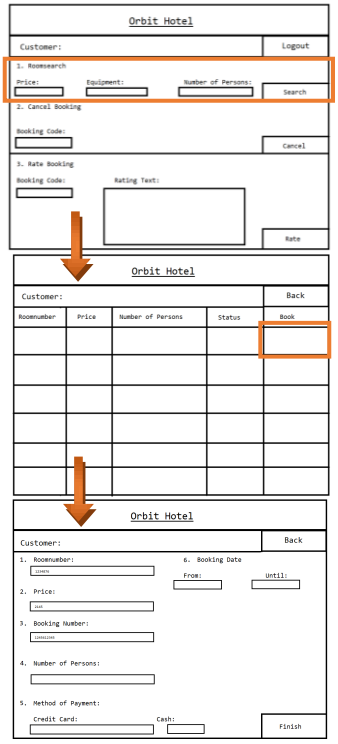
Der User kann sich im jeweiligen Teil seines GUI neben seinen gegebenen Aktionen ausloggen.
Mittels drücken des „Logout“ Buttons kommt man wieder zurück auf den Login / Register Bildschirm.



Customer:

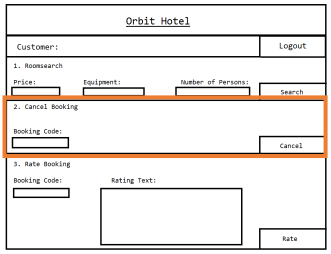
Um ein bestimmtes Zimmer zu suchen, kann der Kunde die Suchkriterien „Price“, „Equipment“ und „Number of Persons“ eingeben und anschließend auf „Search“ drücken.
Anschließend werden den Kriterien entsprechend verfügbare Zimmer angezeigt.

Klickt der Customer anschließend auf das „Book“ Feld, wird das Buchungsfenster angezeigt, bei dem der Kunde noch die Bezahl Methode „Method of Payment“ und die Daten zur Reservierung „From“ und „Until“ eingeben kann.

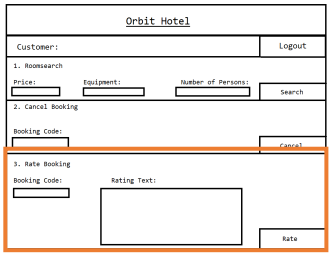


Mit Drücken des „Finish“ Buttons bestätigt er die Reservierung.

Um eine bereits erfolgte Buchung zu stornieren kann der Customer die erhaltene Buchungsnummer eingeben um die Stornierung durchzuführen. Dazu muss zuletzt der „Cancel“ Button gedrückt werden.

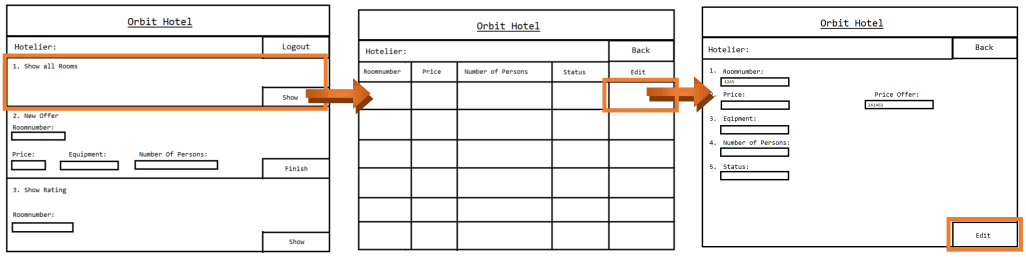


Weiters kann der Customer eine Zimmerbewertung abgeben indem er seine Buchungsnummer in das Feld „Booking Code“ eingibt, eine Bewertung in das dafür vorgesehene Feld eingibt und auf den Button „Rate“ klickt.

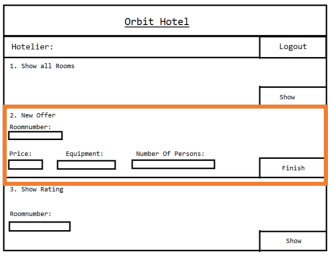


Hotelier:

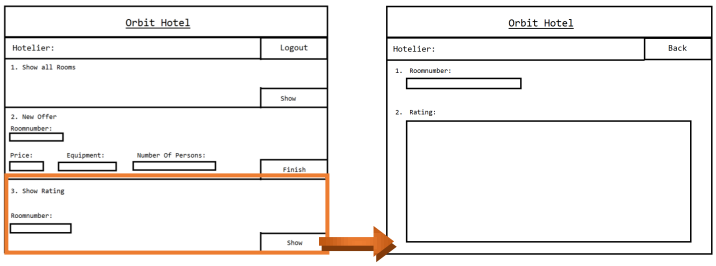
Um als Hotelier alle Zimmer anzuzeigen, kann man den Button „Show“ drücken und bekommt eine Tabelle mit allen Zimmern. Um anschließend ein Zimmer zu editieren muss der Button „Edit“ drücken. Dann werden die Zimmerinformationen angezeigt, welche durch den Hotelier geändert werden können. Die Eingabe wird wiederum mit „Edit“ bestätigt.



Um ein neues Zimmer freizugeben, muss der Hotelier eine Raumnummer unter „Roomnumber“, einen Preis (Price), die Ausstattung (Equipment) und die Anzahl der maximalen Personenbelegung pro Zimmer (Number of Persons) eingeben. Darauf drückt er auf „Finish“ um die Eingabe zu bestätigen bzw. um ein neues Zimmer zu erstellen.

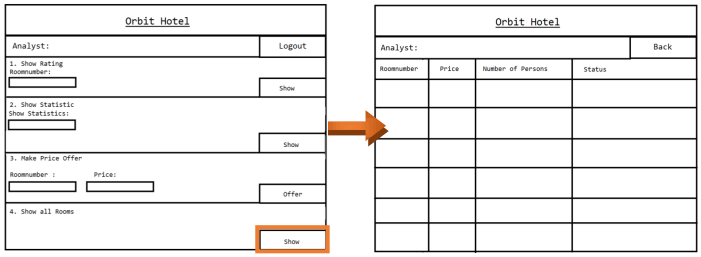


Zuletzt kann der Hotelier noch das Rating für ein gewisses Zimmer anzeigen lassen indem er die Zimmernummer im Feld „Roomnumber“ eingibt und anschließend den Button „Show“ drückt.

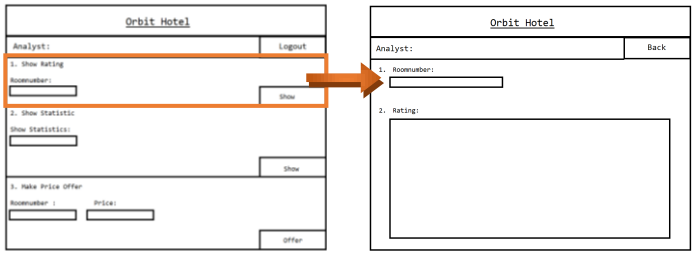


Analyst:

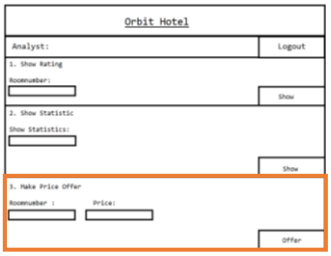
Der Analyst hat wie der Hotelier die Möglichkeit sich alle Zimmer anzeigen zu lassen indem er auf „Show“ drückt.



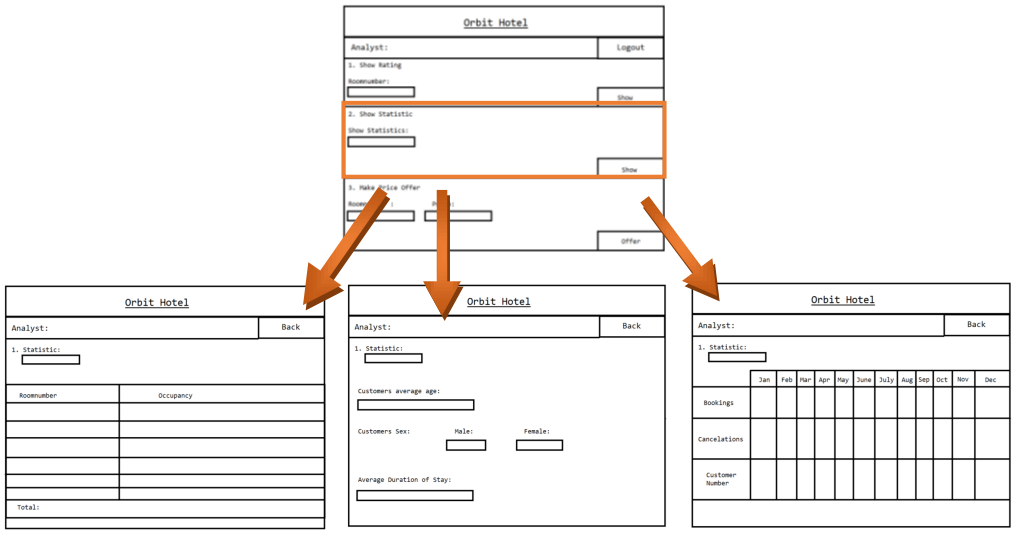
Um ein bestimmtes Rating anzusehen, muss der Analyst die Raumnummer (Roomnumber) des zu betrachtenden Zimmers eingeben.



Der Analyst kann Preisvorschläge für ein bestimmtes Zimmer machen, indem er die Raumnummer (Roomnumber) und den vorgeschlagenen Preis (Price) in das zugehörige Fenster schreibt und mittels Klicken des „Offer“ Buttons die Eingabe bestätigt.



Zuletzt hat er noch die Möglichkeit 3 verschiedene Statistiken abzurufen. Dies geschieht dadurch, dass der Analyst eine der 3 vorgegeben Statistiken im Optionsfeld „Show Statistics“ aufruft und den „Show“ Button drückt. Je nachdem welche Statistik gewählt wurde, werden 3 verschiedene Fenster aufgerufen.



3 Persistente Datenspeicherung

Zur Speicherung und Verwaltung nahmen wir uns eines ähnlichen Konzeptes, wie bei der Abgabe1 an. In Frage kamen also eine Standardserialisierung oder die Speicherung in ein „.txt“ File. Letztendlich entschieden wir uns dann für die Standardserialisierung. Um diese durchzuführen und sie so übersichtlich wie möglich zu gestalten, teilten wir sie in 3 Files auf. So dienen uns diese zur Wahrung der kontextuellen Konsistenz. Unsere jeweiligen Klassenobjekte legen wir in einer ArrayList an, in der sie auch bearbeitet werden können. Diese Objektlisten werden dann von den jeweiligen Speicherklassen(SaveUser,SaveRooms und SaveBookings) in ihre zugehörigen Files(„Users.ser“,“Rooms.ser“ und „Bookings.ser“) geschrieben. Bei jedem Speichervorgang werden die alten Daten komplett durch die Neuen ersetzt. Die Speicherklassen dienen ebenfalls dem Auslesen zuvor gespeicherter Datensätze aus den Files und übertragen diese Objekte dann wieder in eine ArrayList.