



Hotel Management System

myHotel

Abschlussprojekt - Ausbildung zur geprüften
Java Softwareentwicklerin

Dokumentation



Datum: 

Inhaltsverzeichnis

1. Projektvorstellung	4
1.1 Vorwort	4
1.2. Projektidee	4
1.3 Umfang des Projekts	5
2. Visualisierung	5
2.1 Erste Designphase	5
2.2 Zweite Designphase	6
2.2.1 Dashboard	7
3. Finales Design	8
3.1 Vorgehensweise und Überblick	8
3.2 Libraries	8
3.2.1 Implementierung von MaterialFX	8
3.2.2 Implementierung von Faker	8
3.2.3 Implementierung von BCrypt	9
3.2.4 Implementierung von ControlsFX	9
4. Diagramme	9
4.1 Klassendiagramm	10
4.2 Datenbankdiagramm	11
4.3 Anwendungsfalldiagramm	12
5. Methoden	13
5.1 Login	13
5.2 Change Window	15
5.2 Change Window As New Stage	15
6. Front Desk	16
6.1 Kernfunktion für den Empfang	16
6.2 Wichtigste Funktionen	16
6.2.1 Reservierung erstellen	16

6.2.1 Reservierung bearbeiten	17
7. Guest Profiles	18
7.1 Kernfunktion für die Gästekartei	18
7.2 Wichtigste Funktionen	18
7.2.1 Gastprofil anlegen	18
7.2.2 Gastprofil bearbeiten	18
8. Requirements	19
8.1 Funktionale Anforderungen	19
8.2 Nicht-Funktionale Anforderungen	19
8.3 Systemanforderungen	20
9. How-To Dokumentation	20
9.1 Systemanleitung	20
9.1.1 Programm ausführen	20
9.1.2 Login	20
9.1.3 Dashboard	20
9.1.4 Rezeption	20
9.1.6 Reservierung bearbeiten	21
9.1.7 Gästekartei	21
9.1.8 Gastprofil anlegen	21
9.1.9 Gastprofil bearbeiten	21
9.1.10 Passwort ändern	22
9.1.11 Logout	22
9.2 FakeDataInserter	22
9.2.1 Importieren der Datenbank	22

1. Projektvorstellung

1.1 Vorwort

Dieses Dokument soll einen Überblick über das Hotel Management System “myHotel” verschaffen. Es beinhaltet alle Meilensteine, welche bei der Entwicklung des endgültigen Managementsystems hinterlegt wurden. Das Software-System ist hilfreich bei der Verwaltung eines Hotelbetriebs.

Bei der Softwareentwicklung wurde das agile Vorgehensmodell verwendet und entwickelt wurde es mit Hilfe der objektorientierten Programmierung in Java. Während des Projekts wurde die Umgebung Visual Studio Code mit dem Building-Tool Maven angewendet. Für das finale Produkt wurde über die IDE Eclipse die endgültige, ausführbare JAR-Datei gebaut und hierbei auf Maven verzichtet.

In dieser Dokumentation wird auf einige Methoden, die als besonders wichtig erachtet werden, genauer eingegangen, die restlichen Methoden werden nur grob oder soweit nötig beschrieben.

1.2. Projektidee

[REDACTED]

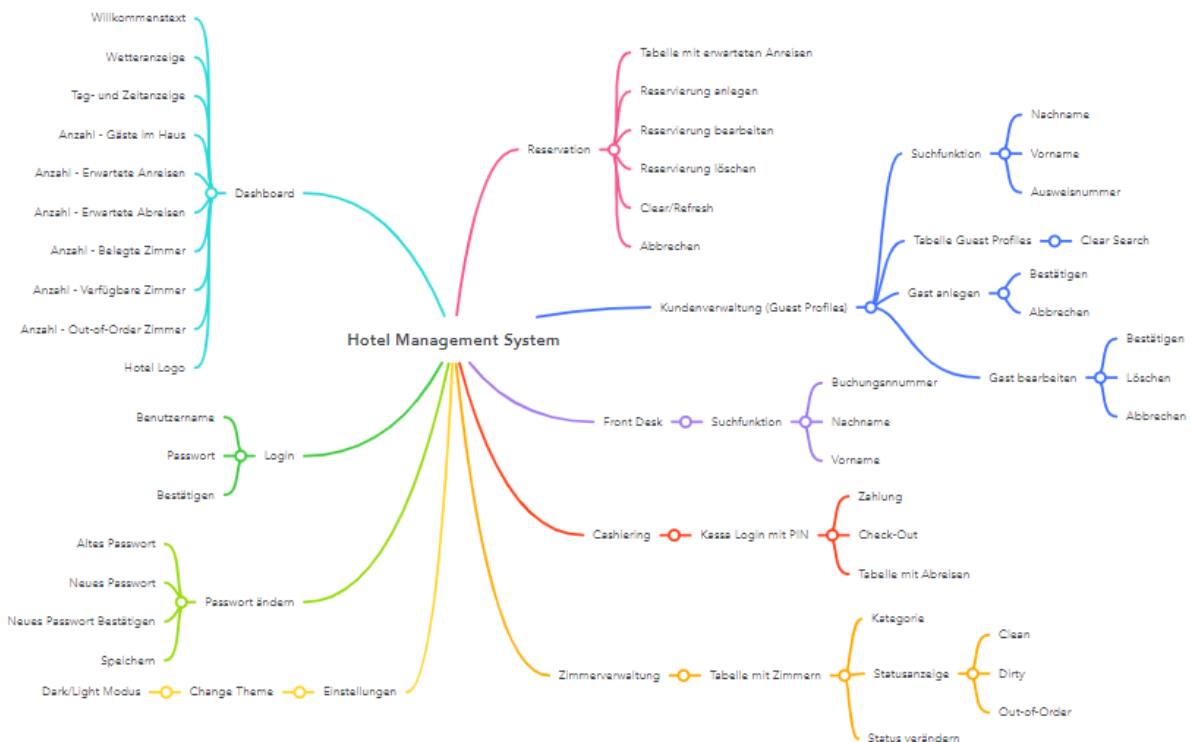
1.3 Umfang des Projekts

Alle Services rund um die Reservierung werden vom Management System verwaltet. Die Gästedaten werden von der Gästeverwaltung verwaltet. Es gibt zwei Endbenutzer: Rezeptionist und Administrator. In künftigen Versionen des Programms ist angedacht, dass sich die beiden Benutzergruppen auch rechtetechnisch unterscheiden. So sollen dann Administratoren beispielsweise auch Konten für Rezeptionisten über eine graphische Benutzerfläche anlegen können. Wie auch später im Anwendungsfalldiagramm ersichtlich, ist die Anlage von Gästen telefonisch, per Mail oder vor Ort im Hotelbetrieb zu tätigen.

2. Visualisierung

2.1 Erste Designphase

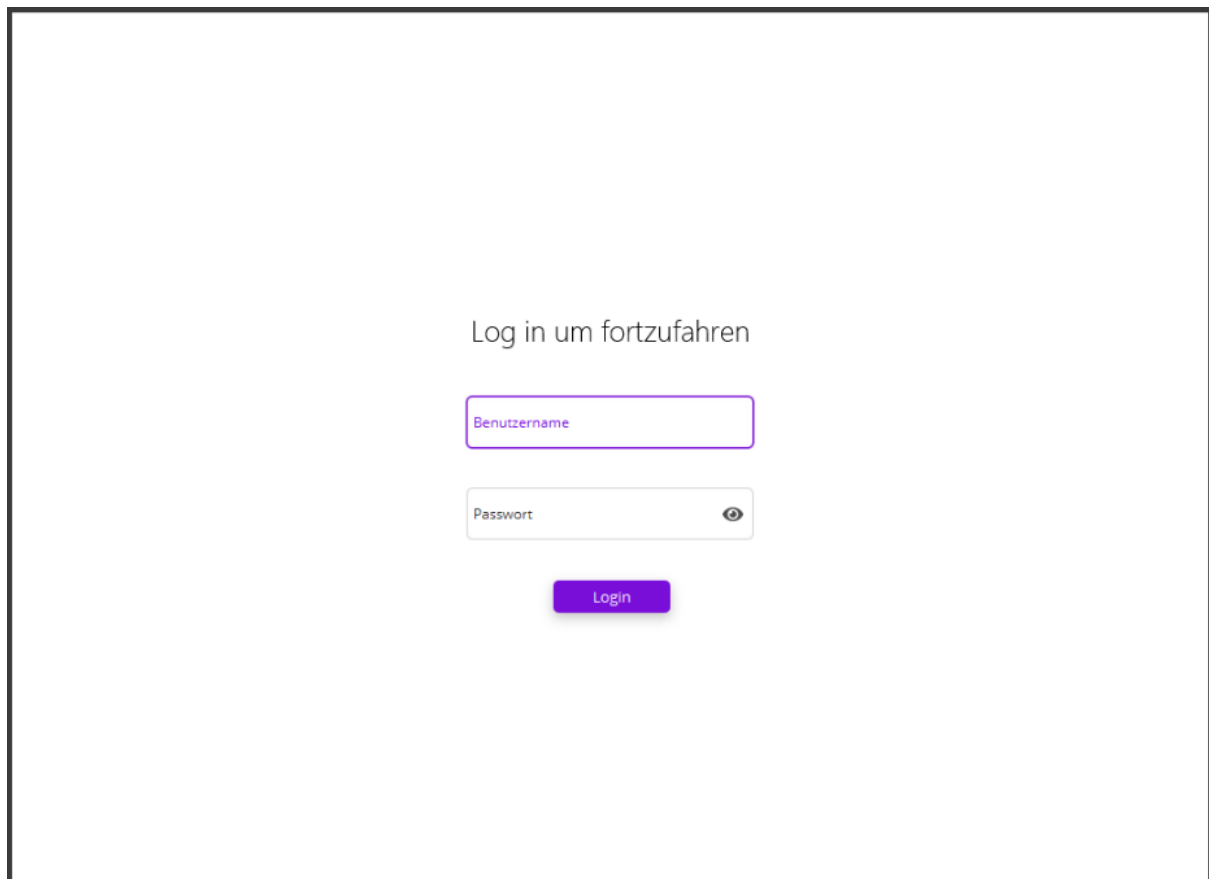
Das erste Design wurde mittels einer MindMap erstellt. Hier war es wichtig herauszufinden, was das Programm können muss und wie es anschließend implementiert werden soll. Für die MindMap wurde das Online-Tool *MindMeister* verwendet. Sie soll einen Überblick über die Vorstellung der Funktionalitäten geben.



2.2 Zweite Designphase

Für das zweite Design wurden mittels Online-Tool *Moqups* Mockup-Folien erstellt, um dem Frontend eine tatsächliche Form zu verleihen. Hier war das Ziel, ein benutzerfreundliches und intuitives User-Interface zu kreieren. Farblich habe ich mich für ein modernes Bright-Purple entschieden.

Hier das Design des Logins:



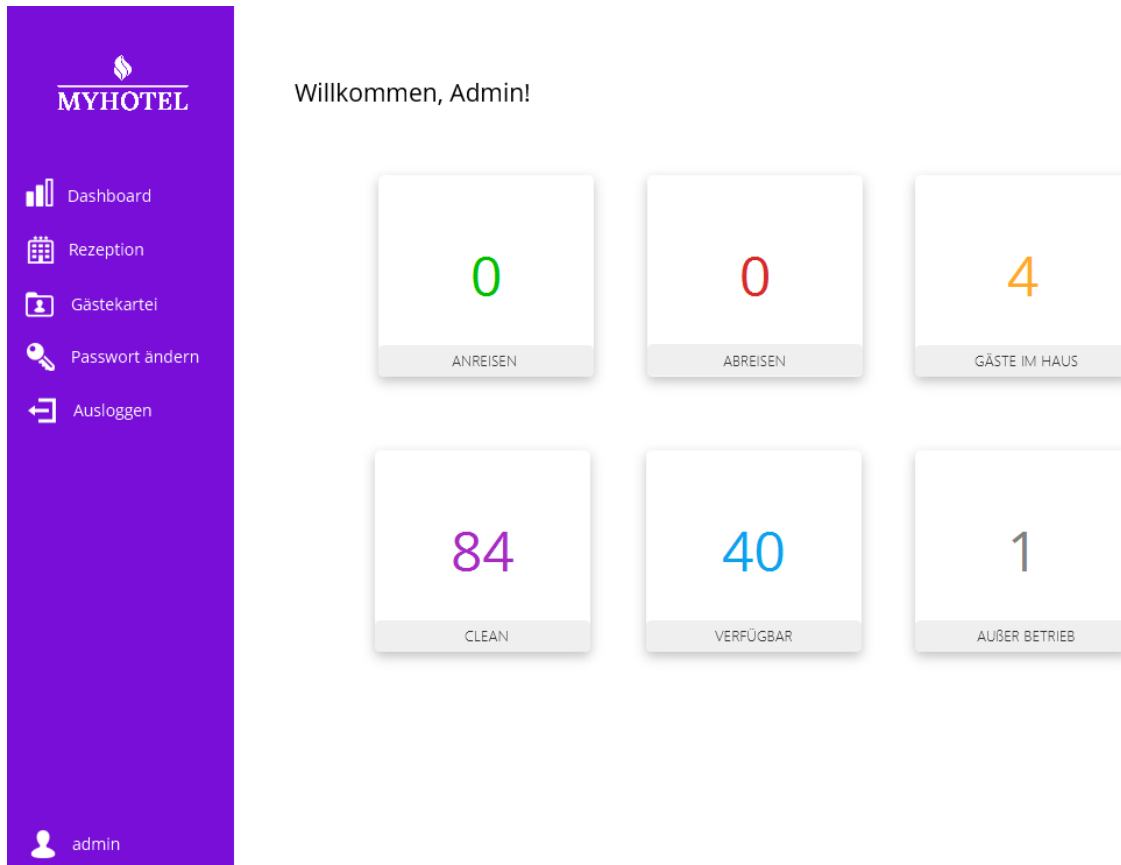
A mockup of a login form centered on a white background. The form consists of the following elements:

- The text "Log in um fortzufahren" (Log in to continue) in a dark gray font.
- A text input field with a light purple border and the placeholder text "Benutzername" (Username) in a light purple font.
- A password input field with a light gray border, the placeholder text "Passwort" (Password) in a light gray font, and a small eye icon on the right side to toggle visibility.
- A bright purple button with the text "Login" in white font.

2.2.1 Dashboard

Das Dashboard ist unser Einstieg in die gesamte Funktionalität des Hotel Management Systems. Mittels Live-Buttons werden alle relevanten Tagesinformationen auf einen Blick gegeben, wie z.B. die Anzahl der An- und Abreisen, verfügbare Zimmer, usw.

Hier das Design des Dashboards:



Für die Sidebar, welche sich über das gesamte Projekt erstreckt, wurde die Funktion `setButtonPictures()` erstellt. Hinter den Icon-Buttons befinden sich Funktionen wie beispielsweise `openFrontDeskWindow()` oder `openGuestProfilesWindow()`. Diese Funktionen arbeiten mit der `changewindow()` oder ggf. mit der `changewindowAsNewStage()` Methode, welche im Abschnitt "Methoden" genauer definiert werden. Müssten noch beschrieben werden. Diese Icon-Buttons ermöglichen es alle Bereiche des Hotel Management Systems per Mausklick zu erreichen. Ebenfalls wurde eine dynamische Sidebar-Animation mittels `setupSidebarAnimation()` erarbeitet. Diese funktioniert mit CSS-Color-Hex-Codes, welche der Maus-Bewegung entsprechend angepasst werden.

3. Finales Design

3.1 Vorgehensweise und Überblick

Für das finale Design wurden mehrere Libraries genutzt, um einerseits dem gesamten User-Interface einen modernen Look zu verleihen und andererseits, um Test-Daten zu erstellen und eine erhöhte Sicherheit zu gewährleisten. Für das Frontend-Design wurde während des gesamten Projekts JavaFX in Kombination mit dem SceneBuilder verwendet.

3.2 Libraries

Für die Implementierung einer benutzerfreundlichen Oberfläche, sowie einem sicheren und umfangreichen Backend wurden einige dem heutigen Standard entsprechende und beliebte Maven-Libraries im Projekt benutzt. Im Folgenden wird auf diese genauer eingegangen.

3.2.1 Implementierung von MaterialFX

MaterialFX wurde benutzt, um der Benutzeroberfläche einen modernen Look zu verleihen. Nodes, Controls sowie Notifications haben dadurch eine schönere Erscheinung mit teils integrierter Animation. Es wurden anstelle von üblichen *Buttons* sogenannte *MFXButtons* verwendet, welche sich am Material-Design-Standard von Google orientieren.

Des Weiteren wurden Tabellen im Material Design (*MFXTableView*) verwendet.

Über das gesamte Projekt hinweg wurde – wo möglich – die MaterialFX-Library verwendet, um dem mittlerweile etwas überholten nativen JavaFX-Design etwas Moderne einzuhauchen.

3.2.2 Implementierung von Faker

Um vorab Daten in die Datenbank einzulesen, wurde die Klasse *FakeDataInserter* erstellt, die mit der Faker-Library arbeitet. Diese ermöglicht es uns, Datensätze automatisiert zu erstellen und diese in die Datenbank einzufügen, um das System anschließend mit diesen zu testen. Verwendet wurden hierbei Objekte der Klasse *PreparedStatement*, um eine höhere Sicherheit gegen SQL-Injections zu gewährleisten. Anschließend wurden die dem ER-Diagramm entsprechend benötigten Datensätze in die Datenbank eingefügt.

3.2.3 Implementierung von BCrypt

Um bestmögliche Sicherheit zu gewährleisten, wurde BCrypt verwendet, um das Benutzerpasswort für den Login im Code sowohl als auch in der Datenbank zu verschlüsseln - dies war eine sehr wichtige Sicherheitsanforderung. Bei der Verwendung wurde außerdem darauf geachtet die verschlüsselten Passwörter zusätzlich mit einem zehn-stelligen Salt zu versehen, damit die in der Datenbank gespeicherten Passwörter umso sicherer sind. Gegen einen höher bezifferten Salt-Wert wurde aus Performance-Gründen entschieden.

3.2.4 Implementierung von ControlsFX

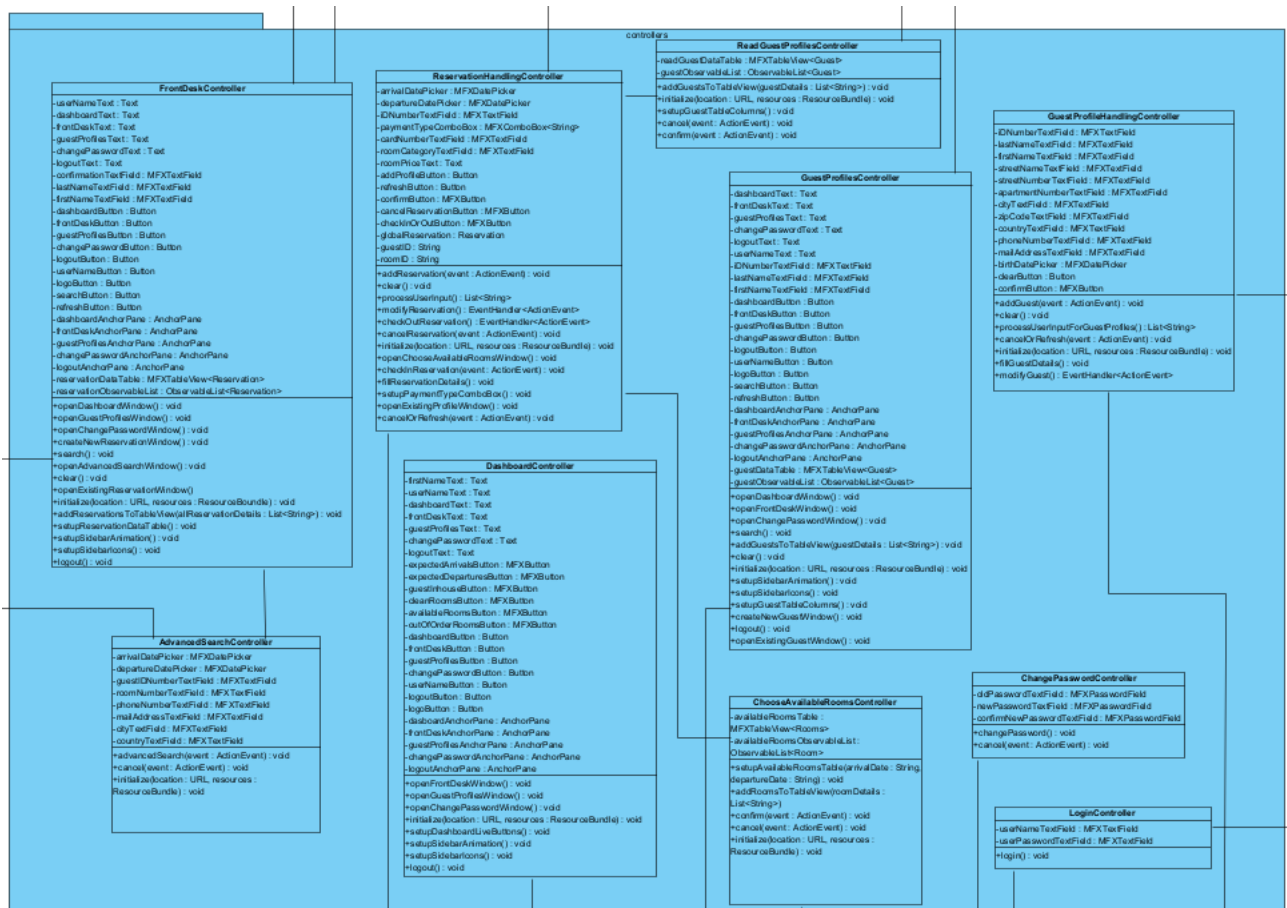
Um eine erhöht-intuitive Benutzeroberfläche zu implementieren war mir wichtig, dass die benutzende Person bei Fehlern ihrerseits (z.B. durch das Nicht-Eingeben von essenziellen Daten) auf elegante Art und Weise darauf hingewiesen wird. Hier hat sich mir die Option angeboten die von JavaFX nativ angebotene *Alert*-Klasse zu verwenden, wobei es mir nicht gefiel, dass die benutzende Person bei jedem kleinen Fehler mit einem Popup konfrontiert wird, welches sie aktiv wegklicken muss. Stattdessen habe ich mir eine kleine Popup-Notification à la Android und iOS gewünscht, welche kurzzeitig aufpoppt und anschließend wieder von selbst verschwindet. Die ControlsFX-Library hat sich hier perfekt angeboten, da sie genau diesen Zweck erfüllt ohne viel extra abzuverlangen.

4. Diagramme

Für das gesamte Software-Projekt wurden Diagramme mittels der grafischen Modellierungssprache UML (Unified Modelling Language) erstellt. Sie sollen einen gegliederten Überblick über die verschiedenen Klassen, die Tabellen mit unterschiedlichsten Daten und die ausgearbeiteten Anwendungsfälle geben. Jedes der folgenden Diagramme ist in den Dokumentationsunterlagen hoch-auflösend zu finden. Im Folgenden werden die Diagramme aufgrund von Platzmangel nur als Übersicht angezeigt.

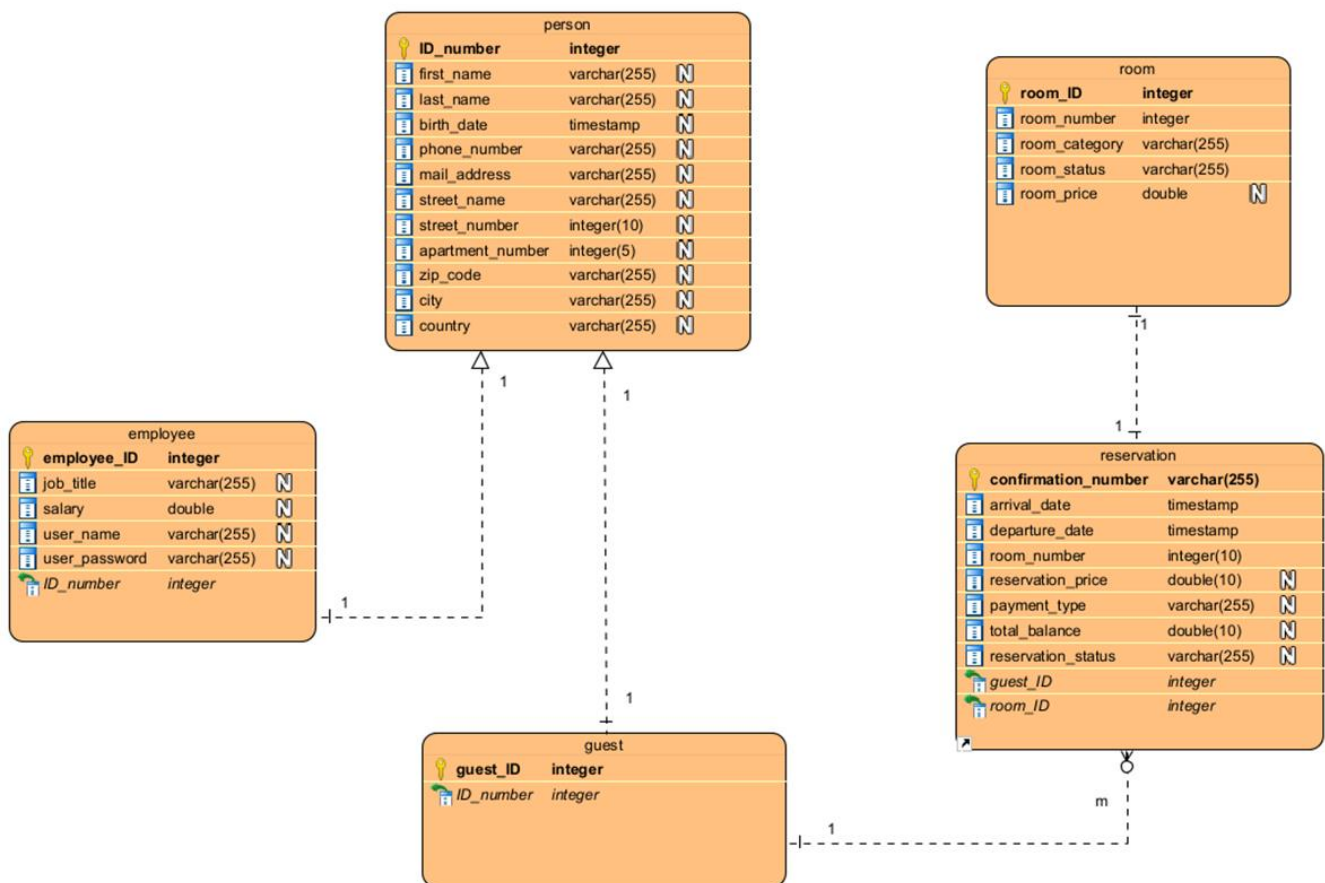
4.1 Klassendiagramm

Das Klassendiagramm (*Class-Diagram*) wurde in der Package-Formation gruppiert dargestellt, um ein klareres Bild zu verschaffen. Hier ein kleiner Ausschnitt (Controllers-Package).



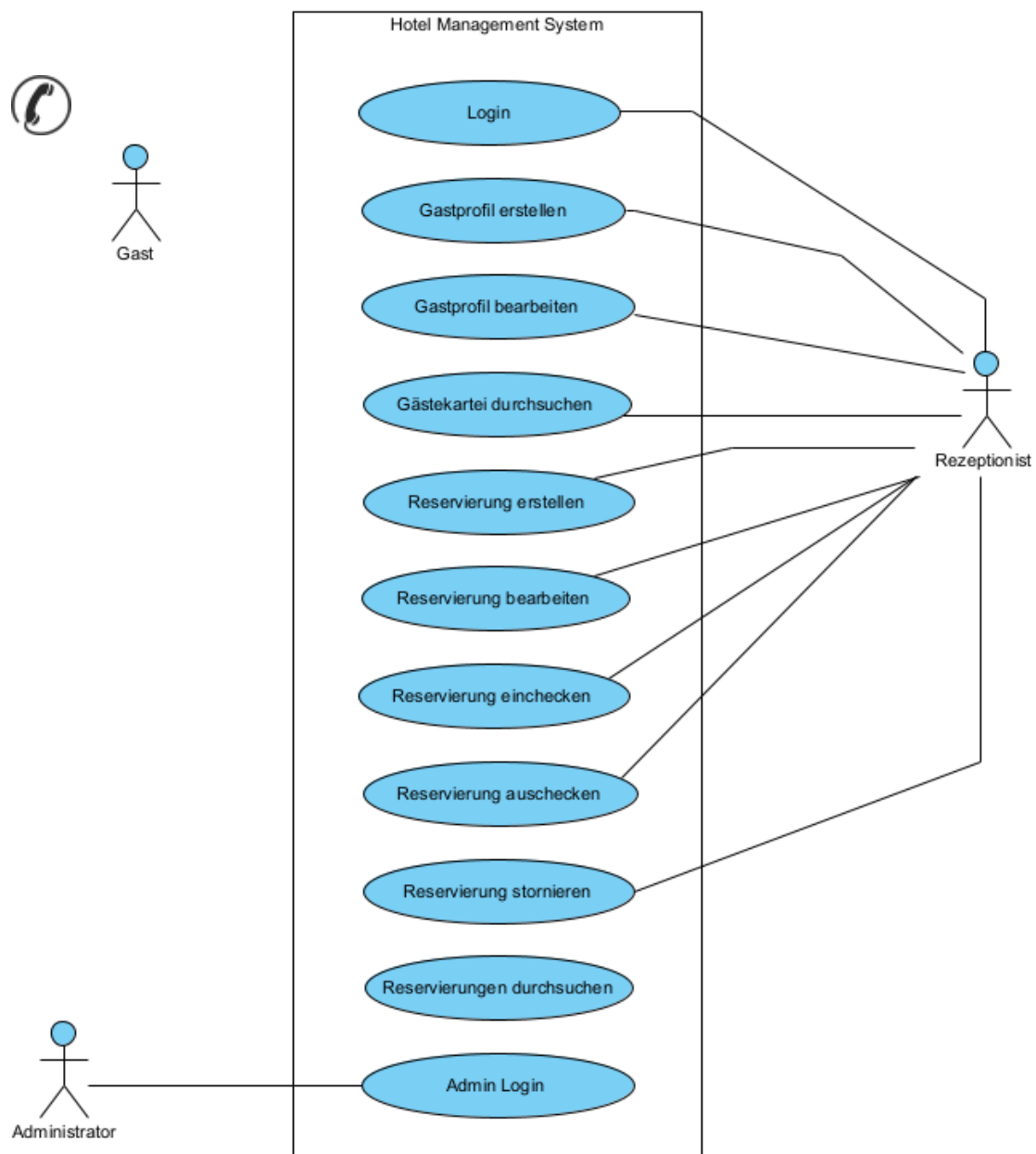
4.2 Datenbankdiagramm

Das Datenbankdiagramm (*Entity-Relationship-Diagramm*) soll einen Überblick über die verschiedenen Tabellen und die genutzten Daten verschaffen. Für die Datenbank wurde das relationale Datenbanksystem “MySQL” verwendet. Es ist für verschiedene Betriebssysteme u.a. Windows, MacOS und Linux verfügbar und anwendbar. Das Hotel Management System wurde basierend auf Windows erstellt. Alle SQL bezogenen Methoden im Softwarecode wurden in eigene Handler-Klassen ausgelagert, um eine klare Trennung der Datenbankanwendungen für einen besseren Überblick zu schaffen.



4.3 Anwendungsfalldiagramm

Dieser Abschnitt enthält die Anwendungsfälle (*Use-Cases*) für das Hotel Management System. Ein Anwendungsfall veranschaulicht alle möglichen Szenarien, die eintreten können, wenn der Benutzer versucht, mit Hilfe des Systems ein bestimmtes Ziel zu erreichen, z.B. ein Gastprofil zu erstellen oder zu bearbeiten.



5. Methoden

Wie schon im Vorwort beschrieben wird auf einige Methoden von hoher Wichtigkeit detaillierter eingegangen, um deren Komplexität zu vereinfachen.

5.1 Login

Aus der Klasse **BCrypt** wird die Methode `checkpw()` aufgerufen.

Diese verschlüsselt den Input, um ihn anschließend in der Datenbank abzugleichen.

```
1.     public static boolean checkPasswordIsCorrect(String inputUserName, String
inputUserPassword)
2.         throws SQLException, UserNameOrPasswordWrongException {
3.         String sqlGetPasswordStatement = "SELECT USER_PASSWORD FROM EMPLOYEE WHERE
USER_NAME =?;";
4.
5.         PreparedStatement myPreparedStatement =
connection.prepareStatement(sqlGetPasswordStatement);
6.         myPreparedStatement.setString(1, inputUserName);
7.         ResultSet resultSet = myPreparedStatement.executeQuery();
8.
9.         while (resultSet.next()) {
10.            if (BCrypt.checkpw(inputUserPassword,
resultSet.getString("USER_PASSWORD"))) {
11.                return true;
12.            } else
13.                throw new UserNameOrPasswordWrongException("Username or password
wrong.");
14.        }
15.        return false;
16.    }
17.
```

In der Methode `login()` werden der eingegebene Benutzername und das Passwort mittels `getText()` geholt, um es anschließend im `LoginSQLHandler` auf Richtigkeit zu überprüfen. In diesem wird zuallererst die Methode `employeeLogin()` mit dem Return-Typ **boolean** aufgerufen, diese dann in sich die beiden Funktionen, `checkUserNameInDatabase()` und `checkPasswordIsCorrect()` aufruft. Returned diese Funktion **true** war die Anmeldung erfolgreich und der Zugang wird gewährt. Ist der Datenabgleich **nicht** erfolgreich, werden Exceptions mittels eines großen **try/catch**-Blocks (Code-Snippet nicht vollständig sichtbar) abgefangen. Hierfür wurden zusätzliche custom exceptions **UserNameNotFoundException** und **UserNameOrPasswordWrongException** erstellt.

```
1. @FXML
2.     public void login() {
3.
4.         String inputUserName = userNameTextField.getText().trim();
5.         String inputUserPassword = userPasswordTextField.getText();
6.         boolean loginSuccess = false;
7.         boolean exceptionThrown = false;
8.
9.         if (inputUserName.isBlank() || inputUserPassword.isBlank()) {
10.             Notifications.create()
11.                 .title("Achtung")
12.                 .text("Felder dürfen nicht leer sein!")
13.                 .position(Pos.CENTER)
14.                 .showWarning();
15.
16.             return;
17.         }
18.
19.         try {
20.             if (DBConnectionHandler.getConnection() == null) {
21.                 DBConnectionHandler.connectToServer();
22.             }
23.             loginSuccess = LoginSQLHandler.employeeLogin(inputUserName,
inputUserPassword);
24.         } catch (IOException | SQLException | UserNameNotFoundException |
UserNameOrPasswordWrongException ex) {
25.             if (ex instanceof UserNameNotFoundException) {
26.                 Alert alert = new Alert(AlertType.WARNING,
27.                     "Username " + inputUserName + " nicht in der Datenbank
gefunden.", ButtonType.OK);
28.                 alert.setTitle("myHotel");
29.                 alert.setHeaderText("Datenbank-Fehler");
30.                 alert.show();
31.                 exceptionThrown = true;
32.             }
33.         }
```

5.2 Change Window

Die Methode `changeWindow()` befindet sich in der *General-Methods-Klasse* und ermöglicht einen vereinfachten *Window-Change* indem das `nextWindow` als Parameter übergeben wird. Diese ist eine der meistgenutzten Methoden im gesamten Hotel Management System.

```
1.  public static void changeWindow(String nextWindow) {
2.      Parent root = null;
3.      try {
4.          FXMLLoader loader = new FXMLLoader();
5.          URL xmlUrl = GeneralMethods.class.getResource("/com/bednara/windows/" +
nextWindow + "Window.fxml");
6.          loader.setLocation(xmlUrl);
7.          root = loader.load();
8.      } catch (IOException e) {
9.          e.printStackTrace();
10.     }
11.     Main.getScene().setRoot(root);
12. }
```

5.2 Change Window As New Stage

Die Methode `changeWindowAsNewStage()` befindet sich ebenfalls in der *General-Methods-Klasse* und ermöglicht es ein Window in einer neuen Stage zu öffnen.

Diese Funktion wurde als Generic-Methode implementiert um die Übergabe von Objekten verschiedener Datentypen als optionalem Parameter zu ermöglichen. Diese Methode findet sich beispielsweise im *FrontDeskController* beim Öffnen oder Anlegen einer Reservierung.

```
1.  public static <T> Stage openWindowAsNewStage(String nextWindow, String
windowTitle, T... object) {
2.      Stage newStage = new Stage();
3.      newStage.setUserData(object);
4.      setCurrentStage(newStage);
5.      try {
6.          Parent root = (Parent) FXMLLoader
7.              .load(GeneralMethods.class.getResource("/com/bednara/windows/" +
nextWindow + "Window.fxml"));
8.          Scene scene = new Scene(root);
9.          newStage.setTitle(windowTitle);
10.         newStage.setScene(scene);
11.         newStage.setResizable(false);
12.         newStage.initModality(Modality.APPLICATION_MODAL);
13.         newStage.getIcons().add(new Image(GeneralMethods.class.
getResource("/com/bednara/images/icon.png")
.toExternalForm()));
14.         newStage.show();
15.     } catch (Exception e) {
16.         e.printStackTrace();
17.     }
18.     return newStage;
19. }
```

6. Front Desk

6.1 Kernfunktion für den Empfang

Die Hauptfunktionalität für den Front Desk Bereich ist es Kundendaten zu verwalten, Reservierungen zu erstellen und die Gäste im Hotel Management System reibungslos ein- und auszuchecken. Hier ist eine übersichtliche Tabelle mit allen Reservierungen zu sehen, welche anhand einer Suchfunktion und einer erweiterten Suche beliebig gefiltert werden können.

6.2 Wichtigste Funktionen

Hierfür wurden neben dem *FrontDeskController* ein eigener *ReservationHandlingController* erstellt in dem sich u. a. die Funktionen `addReservation()` und `modifyReservation()` befinden. Für die SQL bezogenen Funktionen wurden wie immer korrespondierende *SQLHandler* gebaut.

6.2.1 Reservierung erstellen

Eine Reservierung kann manuell oder mit einem bereits vorhandenen Gastprofil erstellt werden. Für die manuelle Anlage wird über `addReservation()` zuallererst die Methode `processUserInputforReservation()` aufgerufen. Diese wurde so erstellt, dass sie abhängig vom Parameter für das Anlegen sowie das Bearbeiten einer Reservierung verwendet werden kann.

Durch ein *Add-Profile-Symbol* kann ein eigenes **ReadGuestProfilesWindow** geöffnet werden aus dem die benutzende Person ein bestehendes Profil wählen kann. Durch das Drücken des *Übernehmen-Buttons* werden die Eingabefelder automatisch vorbefüllt. Somit müssen nur noch das An- und Abreisedatum gewählt werden. Nach der Eingabe korrekter Daten wird der User mittels *Alert-Notification* gefragt, ob die Reservierung bereit für die Anlage ist. Falls ja, wird die Methode `insertReservation()` aufgerufen, welche sich im *ReservationHandlingSQLHandler* befindet. Sollte die Eingabe leer oder fehlerhaft sein, wird der User mittels *Popup-Notification* darauf hingewiesen.

Hier die addReservation() Methode:

```
1. @FXML
2.     public void addReservation(ActionEvent event) {
3.         List<String> userInput = processUserInputForReservations();
4.         if (!userInput.isEmpty()) {
5.             Alert alert = new Alert(AlertType.CONFIRMATION,
6.                 "Sind Sie sicher, dass Sie die Reservierung anlegen möchten?",
7.                 ButtonType.YES, ButtonType.NO);
8.             alert.showAndWait();
9.
10.            if (alert.getResult().toString().contains("YES")) {
11.                ReservationHandlingSQLHandler.insertReservation(userInput); // Hier
12.                wird die Reservierung erstellt
13.                Stage stage = (Stage) ((Node)
14.                    event.getSource()).getScene().getWindow();
15.                event.consume();
16.                stage.hide();
17.            }
18.        } else {
19.            Notifications.create()
20.                .title("Achtung")
21.                .text("Eingabefelder dürfen nicht leer sein!")
22.                .position(Pos.CENTER)
23.                .showWarning();
24.        }
25.    }
```

6.2.1 Reservierung bearbeiten

Die modifyReservation() Methode ermöglicht es bereits vorhandene Reservierungen zu bearbeiten. Diese ruft sich durch das Öffnen der Reservierung über den *Öffnen-Button* auf. Nach dem Öffnen werden die Eingabefelder automatisch mit den Reservierungsdaten befüllt. Diese Daten können nun nach Belieben angepasst werden.

Anschließend werden diese über die Methode processUserInput() verarbeitet und über die sich im ReservationHandlingSQLHandler befindende Methode updateReservation() in der Datenbank mittels Update-Statement erneuert.

7. Guest Profiles

7.1 Kernfunktion für die Gästekartei

Im Bereich der Gästekartei befinden sich Funktionen, um Gastprofile zu verwalten.

Diese ermöglichen es, ein Profil zu erstellen und dieses zu bearbeiten.

Hier sieht man ähnlich wie im *FrontDeskWindow*, eine übersichtliche Auflistung mit allen Gastdaten, welche anhand einer Filterfunktion innerhalb der Tabelle durchsucht werden können.

7.2 Wichtigste Funktionen

7.2.1 Gastprofil anlegen

Ein Gastprofil kann über die Funktion `addGuest()` erstellt werden.

Hier wird die Methode `processUserInputForGuests()` Methode aufgerufen.

Diese wurde wie die `processUserInputForReservations()` Methode für die Reservierung so erstellt, dass sie abhängig vom Parameter für das Anlegen sowie das Bearbeiten von Gastprofile verwendet werden kann. Nach korrekter Eingabe kann das Profil über den *Anlegen-Button*, mittels `insertGuest()` Funktion in die Datenbank hinzugefügt werden. Davor wird mittels Alert-Notification sichergestellt, dass das Profil angelegt werden möchte. Sollte die Eingabe leer oder fehlerhaft sein, wird die benutzende Person durch eine kleine Popup Notification darauf aufmerksam gemacht.

7.2.2 Gastprofil bearbeiten

Über die `modifyGuest()` Methode können bestehende Profile in der Gästekartei angepasst werden. Diese kann nach dem Öffnen eines Gastprofils über den Bearbeiten-Button aufgerufen werden. Nach dem Öffnen werden die Eingabefelder automatisch mit Gast-Daten befüllt. Diese Daten können nun nach Belieben verändert werden.

Anschließend werden sie über die Methode `processUserInputForGuests()` verarbeitet und über die sich im *GuestProfileHandlingSQLHandler* befindende Methode `updateGuest()` in der Datenbank mittels Update-Statement erneuert.

8. Requirements

Die Anforderungen an das Hotel Management System werden durch die Software-Anforderungen (*Requirements*) festgelegt. Diese Anforderungen geben vollständige Informationen darüber, was vom System verlangt ist. Mit Hilfe dieser wurde das Hotel Management System entworfen, erstellt und anschließend getestet.

Im Folgenden werden die Requirements grob beschrieben, eine ausführlichere Version derer befindet sich in den Dokumentationsunterlagen.

8.1 Funktionale Anforderungen

- User Login
- User Logout
- Dashboard Implementierung
- Sidebar Implementierung
- Gästekartei Implementierung
- Gastprofil erstellen
- Gastprofil bearbeiten
- Gästekartei durchsuchen
- Front Desk Implementierung
- Reservierung erstellen
- Reservierung bearbeiten
- Reservierung einchecken
- Reservierung auschecken
- Reservierung stornieren
- Suchfunktion
- Erweiterte Suche
- Passwort ändern

8.2 Nicht-Funktionale Anforderungen

- Dashboard Live-Buttons

8.3 Systemanforderungen

- Datenbankverbindung mit MySQL

9. How-To Dokumentation

9.1 Systemanleitung

9.1.1 Programm ausführen

Das Programm wird nach einem Doppelklick auf die ausführbare JAR-Datei auf Ihrem Computer ausgeführt.

9.1.2 Login

Nach dem Öffnen der Oberfläche werden Sie gebeten Ihren Benutzernamen sowie Ihr Passwort einzugeben und die Eingabe anschließend mit dem *Login-Button* zu bestätigen.

9.1.3 Dashboard

Nach erfolgreichem Login sehen Sie das Dashboard des Hotel Management Systems. Aus diesem Bereich ist es Ihnen möglich, alle Programmbereiche mittels Mausklick zu erreichen. Dabei hilft Ihnen die Sidebar welche sich im gesamten Programm auf der linken Seite befindet. Die Live-Buttons zeigen Ihnen tagesrelevante Daten welche sich im Hintergrund automatisch Updaten.

9.1.4 Rezeption

Im Bereich der Rezeption finden Sie eine Tabelle mit allen Reservierungen. Ihnen ist es möglich über das *Lupe-Symbol* nach einer bestimmten Reservierung zu suchen. Falls Sie eine detailliertere Suche benötigen, steht Ihnen links unten eine erweiterte Suche zur Verfügung. Die Buttons rechts unten ermöglichen Ihnen das Anlegen und das Öffnen zur späteren Bearbeitung.

9.1.5 Reservierung erstellen

Wenn Sie eine neue Reservierung anlegen möchten, klicken Sie bitte rechts unten auf den *Neu-Button*. Nun öffnet sich eine Reservierungsmaske, die es Ihnen ermöglicht Reservierungsdaten wie z.B. An- und Abreisedatum zu wählen.

Über das *Add-Profiles-Symbol* können Sie nach einem bestehenden Gastprofil suchen. Die Zimmerkategorie und die Zimmernummer können erst nach Auswahl des An- und Abreisedatums gewählt werden. Die Auswahl passt sich dem ausgewählten Zeitraum an. Die verfügbaren Zimmer können Sie sich über das *Add-Room-Symbol* anzeigen lassen und über den *Übernehmen-Button* bestätigen. Der Zimmerpreis passt sich automatisch der ausgewählten Kategorie an. Jetzt müssen Sie nur noch auf den Anlegen-Button klicken, um die Reservierung dem Hotel Management System hinzuzufügen. Sie können diesen Vorgang jederzeit über den *Abbrechen-Button* abbrechen.

9.1.6 Reservierung bearbeiten

Nach der Auswahl einer Reservierung, klicken Sie bitte auf den *Öffnen-Button* rechts unten. Sobald Sie dies getan haben, öffnet sich eine vorbefüllte Reservierungsmaske. In dieser ist es Ihnen möglich Reservierungsdaten nach Belieben zu editieren. Nach Änderung der Daten müssen Sie nur noch den *Bearbeiten-Button* als Bestätigung klicken. Sie können diesen Vorgang jederzeit über den *Abbrechen-Button* abbrechen. Das Hotel Management System erkennt automatisch den aktuellen Reservierungsstatus. Wenn die Reservierung für den heutigen Tag erwartet ist und Sie noch nicht eing_checked wurde, erscheint rechts unten ein *Storno-Button* und ein *Check-In-Button*. Ist die Reservierung aber bereits eing_checked erscheint stattdessen ein *Check-Out-Button*.

9.1.7 Gästekartei

Im Bereich der Gästekartei ist es Ihnen möglich über das Lupe-Symbol nach einem bestimmten Profil zu suchen. Falls Sie detaillierter suchen möchten, ist dies über das *Filter-Symbol* innerhalb der Tabelle möglich. Die Buttons rechts unten ermöglichen Ihnen das Bearbeiten und Anlegen eines Profils.

9.1.8 Gastprofil anlegen

Nach dem Klicken des *Hinzufügen-Buttons* öffnet sich die Maske um ein Gastprofil anzulegen. Bitte füllen Sie die Eingabefelder aus und klicken Sie anschließend auf den *Anlegen-Button*. Sie können diesen Vorgang jederzeit über den *Abbrechen-Button* abbrechen.

9.1.9 Gastprofil bearbeiten

Nach der Auswahl eines Profils, können Sie über den *Öffnen-Button* rechts unten, ein Gastprofil öffnen. Sobald Sie dies getan haben, öffnet sich eine vorbefüllte Profilmaske.

In dieser ist es Ihnen möglich Gastdaten über den *Bearbeiten-Button* zu editieren. Jetzt müssen Sie nur auf den *Bearbeiten-Button* klicken, um das Gastprofil der Gästekartei hinzuzufügen. Sie können diesen Vorgang jederzeit über den *Abbrechen-Button* abbrechen.

9.1.10 Passwort ändern

Über das *Schlüssel-Symbol* haben Sie die Möglichkeit Ihr Passwort zu ändern. Dafür müssen Sie Ihr altes Passwort eingeben, Ihr gewünschtes (neues) Passwort und dieses mittels Eingabe (*Neues Passwort bestätigen*) bestätigen.

Jetzt müssen Sie nur den *Bestätigen-Button* klicken und Ihr Passwort wird aktualisiert. Sie können diesen Vorgang jederzeit über den *Abbrechen-Button* abbrechen.

9.1.11 Logout

Aus dem Hotel Management System können Sie sich über das *Logout-Symbol* abmelden. Nach dem Klicken dieses Buttons werden Sie gefragt, ob Sie sich tatsächlich ausloggen möchten. Wenn Sie mit *Ja* bestätigen, kommen Sie auf den Login-Screen zurück. Sie können diesen Vorgang jederzeit über das Klicken von *Nein* abbrechen.

9.2 FakeDataInserter

In der Dokumentation wurde bereits die *FakeDataInserter-Klasse* erwähnt, welche dank der genutzten Faker-Library automatisch generierte Datensätze in die Datenbank hinzufügt. Diese wird hier in einfachen Schritten genauer erklärt, um es dem Anwender zu ermöglichen das Hotel Management System mit Fake-Daten zu befüllen und zu testen.

9.2.1 Importieren der Datenbank

Importieren Sie die Datenbank, welche diesem GitLab-Repository hinzugefügt wurde.

9.2.3 Datenbank befüllen

Bitte kommentieren Sie die `fillDatabase()` Methode und den dazugehörigen Aufruf innerhalb der *Main-Klasse* ein und starten Sie diese anschließend. Dies befüllt die Datenbank mit automatisch generierten Datensätzen und ermöglicht es Ihnen dadurch das Hotel Management System zu testen. Eine Manuelle Eingabe von Gastdaten ist über den Bereich der Gästekartei möglich.