Projektbericht

Immobilienverwaltung



Erstellt von: Markus Badzura  
Ungarnstraße 62  
13349 Berlin

Ausbildungsberuf: Fachinformatiker  
(Anwendungsentwicklung)

Prüfungskommission: FIAN00

Ausbildungsstätte: Comhard GmbH  
Möllendorffstr. 52  
10367 Berlin

Auftraggeber/Praktikumsbetrieb Übungsfirma  
Möllendorffstr. 52  
10367 Berlin

**Inhaltsverzeichnis**

[1 Projektauftrag 5](#_Toc485489513)

[1.1 Beschreibung der Ausgangslage 5](#_Toc485489514)

[1.2 Ableitung des Projektauftrages 5](#_Toc485489515)

[2 Projektplanung 6](#_Toc485489516)

[2.1 Aufgaben- und Zeitplanung 6](#_Toc485489517)

[2.2 Personal-, Sachmittel- und Kostenplanung 6](#_Toc485489518)

[2.2.1 Personal 7](#_Toc485489519)

[2.2.2 Sachmittel 7](#_Toc485489520)

[2.2.3 Kostenplanung 7](#_Toc485489521)

[2.3 Plan der Maßnahmen zur Qualitätssicherung 8](#_Toc485489522)

[3 Projektdurchführung 9](#_Toc485489523)

[3.1 Ist-Analyse 9](#_Toc485489524)

[3.1.1 Ermittelte Schwachstellen 9](#_Toc485489525)

[3.1.2 Datenanalyse 9](#_Toc485489526)

[3.1.3 Datenfluss 10](#_Toc485489527)

[3.2 Entwicklung Sollkonzept 10](#_Toc485489528)

[3.2.1 Nutzwertanalyse 10](#_Toc485489529)

[3.2.2 Fachentwurf Immobilienverwaltung 11](#_Toc485489530)

[3.2.3 Grobentwurf Immobilienverwaltung 12](#_Toc485489531)

[3.3 Entwicklung des Konzepts mit Prototyp 14](#_Toc485489532)

[3.3.1 Implementierung der Datenstruktur 14](#_Toc485489533)

[3.3.2 Entwicklung des Prototypen 14](#_Toc485489534)

[3.4 Abnahme und Übergabe an den Kunden bzw. AG 16](#_Toc485489535)

[4 Auswertung der Projektarbeit 17](#_Toc485489536)

[4.1 Darstellung der Projektergebnisse 17](#_Toc485489537)

[4.2 Bewertung des Projektablaufes 17](#_Toc485489538)

[5 Anlagen, Verzeichnisse 18](#_Toc485489539)

[5.1 Anlage: Projektauftrag 18](#_Toc485489540)

[5.2 Anlage: Auszug aus dem Pflichtenheft 20](#_Toc485489541)

[5.3 Anlage: Leistungsangebot 21](#_Toc485489542)

[5.4 Anlage: Datenanalyse 22](#_Toc485489543)

[5.5 Anlage: Nutzwertanalyse 23](#_Toc485489544)

[5.6 Anlage Datenbankmodellierung 24](#_Toc485489545)

[5.7 Anlage SQL-Statement Datenbankmodell 25](#_Toc485489546)

[5.7.1 Eingefügte Datensätze 26](#_Toc485489547)

[5.8 Anlage config-Datei 27](#_Toc485489548)

[5.8.1 Inhalt der config.txt 27](#_Toc485489549)

[5.8.2 Anlage Auszug aus immo\_config.java 27](#_Toc485489550)

[5.9 Anlage Auszug Quelltext Neuanlage Wohnung mit Anlegen eines neuen Mieters 28](#_Toc485489551)

[5.9.1 Erstellen der Benutzeroberfläche 28](#_Toc485489552)

[5.9.2 Methoden zum Befüllen der Auswahlfelder Mieter, Küchen, Heizung, Bad 30](#_Toc485489553)

[5.9.3 Methode Abfrage Datenbank zum Befüllen Auswahlfeld Mieter 31](#_Toc485489554)

[5.9.4 Methode zum Auslesen der Datenbanktabellen Kuechen, Bad, Heizung 31](#_Toc485489555)

[5.9.5 Methoden zum Öffnen der Datenbankverbindung, Durchführung der Select-Abfragen und Schließen der Datenbankverbindung 32](#_Toc485489556)

[5.9.6 Methode zum Befüllen der Wohnungsübersicht bei Neueingabe einer Wohnung 33](#_Toc485489557)

[5.9.7 Methode zum Auslesen der vorhanden Wohnungen einer Immobilien-ID aus der Datenbank 33](#_Toc485489558)

[5.9.8 Immo\_wohnungen.java Klasse zur Objekterstellung von Wohnungen 34](#_Toc485489559)

[5.9.9 Auszug ItemListener, wenn Auswahl „Neue Auswahl hinzufügen“ beim Auswahlfeld Mieter 37](#_Toc485489560)

[5.9.10 Auszug ActionListener zum Speichern und Resetten der Eingaben einer neuen Wohnung 37](#_Toc485489561)

[5.9.11 Eingabedialogfenster für neuen Mieter 37](#_Toc485489562)

[5.10 Anlage Oberflächenentwürfe 38](#_Toc485489563)

[5.10.1 Menüführung 38](#_Toc485489564)

[5.10.2 Entwurf Startseite 39](#_Toc485489565)

[5.10.3 Entwurf Hilfeseite 39](#_Toc485489566)

[5.10.4 Entwurf Neuaufnahme Immobilien 40](#_Toc485489567)

[5.11 Anlage Testberichte 41](#_Toc485489568)

[5.11.1 Funktionstest Tastaturkürzel 41](#_Toc485489569)

[5.11.2 Funktionstest Menüpunkte 41](#_Toc485489570)

**Abbildungsverzeichnis**

[Abbildung 1 - Projektstrukturplan 6](#_Toc485105977)

[Abbildung 2: Datenflussdiagramm Hausverwaltung "Private Home" 10](#_Toc485105978)

[Abbildung 3: Datenbankmodellierung Immobilienverwaltung 22](#_Toc485105979)

[Abbildung 4: Entwurf Startseite 27](#_Toc485105980)

[Abbildung 5: Entwurf Hilfeseite 27](#_Toc485105981)

**Tabellenverzeichnis**

[Tabelle 1 - Maßnahmen zur Qualitätssicherung 8](#_Toc485105982)

[Tabelle 2 - Projektauftrag 17](#_Toc485105983)

[Tabelle 3: Datenanalyse Hausverwaltung "Private Home" 20](#_Toc485105984)

[Tabelle 4: Präferenzdreieck 21](#_Toc485105985)

[Tabelle 5: Nutzwertanalyse 21](#_Toc485105986)

# Projektauftrag

## Beschreibung der Ausgangslage

Die Hausverwaltung „Private Home“ verwaltet Immobilien mit Eigentumswohnungen. Die Verwaltung besteht aktuell aus 1 Geschäftsführer und 1 Mitarbeiterin. Alle relevanten Daten werden in einer Excel-Mappe mit verschiedenen Tabellen verwaltet, welche sich auf dem PC der Mitarbeiterin befindet.

Aktuell verwaltet die Hausverwaltung 2 Immobilien, plant allerdings den Bestand der zu verwaltenden Immobilien zu erhöhen. Ebenso vorgesehen für das kommende Jahr ist die Einstellungen eines weiteren Mitarbeiters in der Verwaltung.

Die einzelnen PCs innerhalb der Verwaltung sind nicht miteinander verbunden, allerdings verfügt die Hausverwaltung über einen WLAN-Router.

## Ableitung des Projektauftrages

Durch die geplante Vergrößerung der zu verwaltenden Immobilien wird die Hausverwaltung „Private Home“ mit der Verwaltung der Daten innerhalb verschiedener Excel-Mappen schnell an einen Punkt gelangen, an welchen keine Übersichtlichkeit mehr gewährleistet ist. Außerdem erhöhen sich mit jeder Excel-Mappe die Fehlerquellen beim Eintragen in die Tabellen.

Bei der Projektumsetzung geht es um das Erstellen eines Konzeptes mit Prototyp, wie der bisherige Verwaltungsvorgang mit einer Datenbankbasierenden Anwendung umgesetzt werden kann. Dieses Konzept dient als Diskussionsgrundlage und der Bewertung der Realisierung. Hierbei sollen verschiedene Lösungswege mit den dadurch resultierenden Kosten ermittelt werden.

Durch eine Datenbankbasierende Anwendung werden folgende Vorteile erzielt.

* Zeitersparnis (Bei Eingabe und Selektierung von Daten)
* Reduzierung möglicher Fehlerquellen
* Schnelleres Einarbeiten neuer Mitarbeiter durch intuitive Bedienung

Somit werden die Mitarbeiter bei der Verwaltung deutlich entlastet.

Die Projektdurchführung erfolgt als Einzelprojekt in den Räumen der Comhard GmbH.

# Projektplanung

## Aufgaben- und Zeitplanung

Die Realisierung des Konzeptes Immobilienverwaltung erfolgt in insgesamt 96 Unterrichtseinheiten. In Vorbereitung des Konzeptes ist eine ausführliche IST-Analyse durchzuführen. Im Rahmen des Soll-Konzeptes sind die Anforderungen an das Konzept und den zu entwickelnden Prototyp zu spezifizieren. Anschließend sind das Konzept zu erarbeiten und der Prototyp zu codieren.

Die Vorgehensweise entspricht somit weitestgehend dem Wasserfallmodell. Nachfolgender Projektstrukturplan gibt eine Übersicht über die einzelnen Projektphasen und deren zeitliche Dauer.

Abbildung - Projektstrukturplan

Folgende Termine sind dabei einzuhalten:

Vorlage Projektbericht bis Punkt 1: 06.06.2017, 15:00 Uhr

Vorlage Projektbericht bis Punkt 2: 07.06.2017, 15:00 Uhr

Vorlage Projektbericht bis Punkt 3.2: 09.06.2017, 11:00 Uhr

Vorlage Projektbericht: 19.06.2017, 15:00 Uhr

Präsentation des Projektes: 21.06.2017

## Personal-, Sachmittel- und Kostenplanung

Während des Projektes werden Personal und Sachmittel gebunden. Die entstehenden Kosten können auf Grund fehlender Informationen über Gemeinkosten nicht als tatsächlicher Wert ermittelt werden. Eine Kostenplanung erfolgt an Hand eines fiktiven Angebotes.

### Personal

Die Durchführung des Projektes wird als Einzelprojekt durchgeführt. Zusätzlich wird in unregelmäßigen Abständen Frau Lorenz bei Konsolidierungsgespräche gebunden.

Weiterhin sind keine personellen Ressourcen eingeplant.

### Sachmittel

Die Projektdurchführung findet im Unterrichtsraum 417 statt. Dieser beinhaltet Tower-PCs zur Umsetzung der Immobilienverwaltung. Um das Konzept mit Prototyp zu entwickeln besteht außerdem Zugang zu einem SQL-Server innerhalb des Netzwerkes der Comhard Übungsfirma GmbH.

Für die Umsetzung des Projektes wird nachfolgende Software verwendet, die in der Comhard Übungsfirma GmbH bereits vorhanden ist, oder als kostenfreier Download seitens der Hersteller zur Verfügung gestellt wird.

* Betriebssystem Windows 7
* Microsoft Office Paket 2010 (Insbesondere Word 2010 und Excel 2010)
* NetBeans 8.1(Entwicklungsumgebung Java)
* Java JDK 1.8
* Dia (Zur Erstellung der notwendigen Diagramme)
* Foxit Reader (Zur Erstellung von PDF-Dateien)
* MySQL-Server (im Comhard-Netz vorhanden)
* EasySetup (kostenloses Programm zum Erstellen einer Setup-Datei)

### Kostenplanung

Für die Realisierung des Projektes können direkte und indirekte Kosten anfallen. Direkte Kosten sind alle Kosten, welche direkt mit dem Projekt in Zusammenhang stehen. Zur Erstellung der Konzeption sind sämtliche Hard- und Softwarekomponenten bereits vorhanden. Hierdurch fallen keine Anschaffungskosten an.

Die Planung und Durchführung wird durch mich als Praktikant selbstständig realisiert. Als Praktikant im Rahmen der beruflichen Weiterbildung erhalte ich keine Zuwendungen in Form von Lohn oder Prämien, so dass diesem Projekt keine direkten Lohnkosten zugeordnet werden.

Insgesamt fallen also keine dem Projekt direkt zurechenbare Kosten an. Indirekte Kosten sind die Kosten für den Arbeitsplatz (Stromkosten, Leasingkosten, Abschreibung auf Technik, Telefon, Miete, Versicherungen, Heizkosten usw.). Über den Betriebsabrechnungsbogen werden diese Kosten auf alle Kostenstellen verteilt. Der Gemeinkostensatz der Kostenstelle der Abteilung, in welcher das Projekt durchgeführt wird, wurde mir nicht bekannt gegeben.

Angenommen es würde sich um ein externes Projekt handeln, könnte dem Kunden folgende Leistung in Rechnung gestellt werden. Für die Leistungsstunden sollen laut Konsolidierungsgespräch vom 07.06.2017, 65,00 Euro (netto) pro Stunde in Rechnung gestellt werden. Daraus lässt sich ein Leistungsangebot im Umfang von 5569,20 € brutto ableiten. (siehe Anlage: Leistungsangebot).

## Plan der Maßnahmen zur Qualitätssicherung

Als stetige Maßnahme zur Qualitätssicherung wird die Rücksprache zum Auftraggeber angestrebt. Bestandteil dieser Rücksprachen sind insbesondere die Oberflächenentwürfe und Funktionalitäten, um auf etwaige Kundenwünsche auf Bedarf und nach Machbarkeit einzugehen.

Das in der ISO 9126 und DIN 66272 genannte Qualitätsmodell zur Softwarequalität erfordert weitere Maßnahmen, welche in nachfolgender Tabelle dargestellt werden.

| Qualitätsmerkmal | Maßnahme |
| --- | --- |
| Funktionalität | * Umsetzung der im Pflichtenheft (siehe Anlage: Auszug aus dem Pflichtenheft) festgelegten, spezifizierten Eigenschaften. |
| Zuverlässigkeit | * Unit-Tests zum Testen der Methoden und Funktionen * Black- und White-Box - Tests |
| Benutzbarkeit | * Intuitive Menüführung * Implementierung eines Hilfesystems * Ergonomische Oberflächengestaltung |
| Effizienz | * Relationales Datenbankmodell in 3ter Normalform * Gewährleistung referenzielle Integrität |
| Änderbarkeit Wartbarkeit | * Dokumentation des Quelltexts * Erstellen JavaDoc * Objektorientierte Programmierung * Einhaltung der Allgemeinen Konventionen der Programmierung |
| Übertragbarkeit | * Nutzung einer Plattformunabhängigen Programmiersprache (Java) * Dynamische Fensteranpassung zur Nutzung mit verschiedenen Bildschirmdiagonalen |

Tabelle - Maßnahmen zur Qualitätssicherung

# Projektdurchführung

## Ist-Analyse

Die Hausverwaltung „Private Home“ verwaltet Immobilien in Excel-Tabellen. Die Eingabe der erforderlichen Daten erfolgt an einen PC durch die Mitarbeiterin. Die PCs sind nicht miteinander vernetzt. Zur Durchführung der Ist-Analyse standen die Mittel der Befragung und Beobachtung, sowie die übergebene Excel-Arbeitsmappe einer Immobilie, zur Verfügung.

### Ermittelte Schwachstellen

Die durchgeführte Ist-Analyse anhand der durchgeführten Befragungen, Beobachtungen und Analyse der zur Verfügung gestellten Unterlagen offenbart einige Schwachstellen. Diese sind in nachfolgender Aufzählung aufgeführt.

* Keine ausreichende Dokumentation innerhalb der Arbeitsmappen.
* Daten werden mehrfach in die Tabellen eingetragen.
* Datenstrukturen wiederholen sich in verschiedenen Excel-Mappen.
* Mangelnde Übersichtlichkeit durch Verteilung auf mehrere Tabellen.
* Mehrfacherfassung von Wohnungen und Immobilien wird nicht verhindert.
* Betriebskostenabrechnung gestaltet sich als umständlich, da die notwendige Trennung zwischen Kaltmiete und Nebenkosten bei Zahlung nicht erfasst wird.
* Mietänderungen während einer Betriebsabrechnungsperiode werden nicht berücksichtigt.
* Verteilung der Kosten nicht automatisiert (Rechenfehler möglich).
* Zugriff auf einen PC beschränkt.
* Es kann nur 1 Mitarbeiter mit den Daten arbeiten.
* Datenverlust möglich, da kein Sicherungskonzept erkennbar
* Fehlende Mietzahlungen sind nicht sofort ersichtlich.
* Ein- und Ausgaben werden nicht direkt den Verursachern zugeordnet.
* Fehlende Daten im Vergleich zur Projektanforderung (z. B. Adressangaben Mieter, Personenangaben)

### Datenanalyse

Die Verarbeitung der Daten der Hausverwaltung „Private Home“ erfolgt bisher pro Immobilie in einer Excel-Arbeitsmappe. Es werden 2 Tabellen verwendet.

* Tabelle Immobilie
  + Immobilien, Eigentümer und Mieterinformationen
* Tabelle Ein- und Ausgaben
  + Sämtliche Ein- und Ausgaben, welche die Immobilie betreffen

Auf Basis der von der Hausverwaltung „Private Home“ zur Verfügung gestellten Excel-Arbeitsmappe wurde eine Datenanalyse durchgeführt. Die Datenanalyse ist im Anhang unter Punkt 5.4 beigefügt.

### Datenfluss

Die Datenspeicherung erfolgt in einer Excel-Arbeitsmappe mit den Tabellen Immobilie und Ein- und Ausgaben. In der Tabelle Immobilie wird der Eigentümer, die Wohnungen und die Mieter verwaltet. Die Mietzahlungen werden in der Tabelle Ein- und Ausgaben eingetragen, ebenso die Kosten, welche die Immobilie betreffen. Aus diesen Informationen heraus ergibt sich folgender Datenfluss.

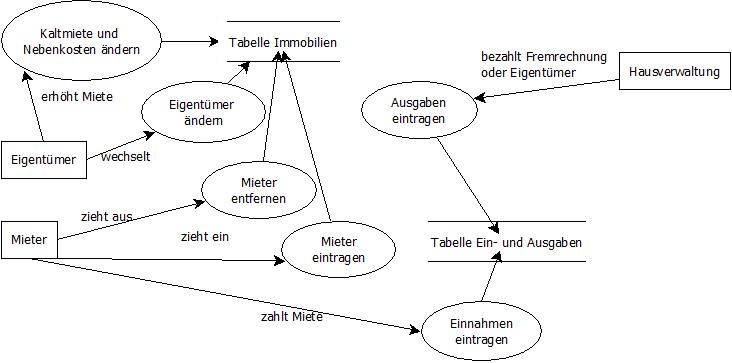


Abbildung : Datenflussdiagramm Hausverwaltung "Private Home"

## Entwicklung Sollkonzept

Das Sollkonzept gliedert sich grob in zwei Bereiche. Einmal in den Bereich Fachentwurf, welcher wiedergibt, was das zu entwickelnde Konzept leisten muss und in den Bereich Grobentwurf, welcher wiedergibt, wie es verwirklicht werden kann.

Um festzulegen, für welchen Prototyp das Sollkonzept erstellt wird, wurde zu Beginn eine Nutzwertanalyse durchgeführt, um die verschiedenen Lösungsansätze einer Immobilienverwaltung zu bewerten und zu gewichten.

Als Basis für die Entwicklung des Sollkonzeptes ist das Lastenheft, die Schwachstellen der Ist-Analyse und die Datenanalyse herangezogen worden.

### Nutzwertanalyse

Es stehen verschiedene Lösungsmöglichkeiten zur Verfügung. Als einzelne Desktop-Anwendung besteht die Möglichkeit einer Embedded Database, z. B. H2. Da allerdings bereits eine Planung für einen weiteren Mitarbeiter besteht, sollte die Möglichkeit bestehen, dass auch alle Mitarbeiter und alle PCs auf die Datenbank zugreifen können. Hierfür stehen die Möglichkeiten einer Access-Datenbank oder einer MySQL-Datenbank zur Verfügung. Die PCs der Hausverwaltung sind untereinander nicht verbunden. Sollte dies weiterhin gewünscht sein, so empfiehlt sich eine Cloud-Lösung.

Für diese Lösungsmöglichkeiten wurde eine Nutzwertanalyse (siehe Anlage: Nutzwertanalyse) durchgeführt. Die Nutzwertanalyse hat ergeben, dass eine Konzeption auf Basis einer MySQL-Datenbank erfolgt. Die Nutzung einer ServerCloud ist hierbei möglich. Die monatlichen Kosten hierbei betragen zwischen 10 bis 20 Euro. Durch Root-Zugriff ist die Installation des ansonsten kostenfreien MySQL-Servers möglich. Falls sich die Entscheidungsträger der Hausverwaltung entscheiden ihre Rechner untereinander zu vernetzen, so besteht die Möglichkeit, einen MySQL-Server auch innerhalb des Firmennetzwerkes zu installieren.

### Fachentwurf Immobilienverwaltung

**Funktionale Anforderungen:**

* Löschung einer Immobilie.
* Neuanlage einer Immobilie mit den dazugehörigen Wohneinheiten und Ausstattungsmerkmalen und des Eigentümers
* Ändern eines Eigentümers
* Anlegen und Ändern von Mietern
* Ändern der Ausstattungsmerkmale einer Wohnung
* Bearbeiten des Kostenkontos einer Immobilie
* Übersicht aller Immobilien
* Übersicht aller Mieter einer Immobilie
* Verteilung der verteilungsfähigen Kosten auf die einzelnen Mieter an Hand der Größe einer Wohnung
* Verteilung der verteilungsfähigen Kosten auf die einzelnen Mieter an Hand der Anzahl der Wohnungen innerhalb der Immobilie.
* Datensicherung automatisch und manuell.
* Standortunabhängigkeit des mySQL-Servers garantieren

**Qualitative Anforderungen:**

* Nachweis der Funktionalität durch Unit-Tests, Blackbox- und Whitebox-Tests.
* Optimierung des Speicherverbrauchs durch Datenspeicherung in Datenbank (Datenbankentwurf in 3ter Normalform)
* Referenzielle Integrität beachten
* Einfache und intuitive Benutzerführung
* Ausführliche Dokumentation
* Erstellen eines einfachen Benutzerhandbuches für den Prototypen
* Einhaltung der allgemeinen Konventionen für Quellcode
* Einhaltung Grundsätze der Software-Ergonomie

**Systembezogene Anforderungen:**

* Desktop-Anwendung
* Plattformunabhängiger Einsatz
* Unabhängigkeit vom Standort des mySQL-Servers
* Zugriff auf SQL-Server über WLAN-Router

**Projektbezogene Anforderungen:**

* Einzelprojekt durch Praktikanten Markus Badzura
* Zeitlicher Rahmen 72 Stunden
* Erstellung eines ausführlichen Entwurfs mit Prototyp, keine produktionsreife Anwendung

### Grobentwurf Immobilienverwaltung

***Entwicklungskonzept***

Die Desktop-Anwendung wird mit der objektorientierten Programmiersprache Java, mit der derzeit aktuellen JDK 1.8, entwickelt. Als Entwicklungsumgebung wird die bereitstehende Entwicklungsumgebung NetBeans 8.1 verwendet.

Alle notwendigen Daten zur Datenbankverbindung werden in einer Konfigurationsdatei dem Quellcode ausgelagert, damit die Hausverwaltung nach einem evtl. Wechsel des MySQL-Servers ohne großen Aufwand auf den neuen MySQL-Server zugreifen kann.

Die Datenbankmodellierung wird auf dem zur Verfügung stehenden MySQL-Server innerhalb des Comhard-Netzwerkes erfolgen. Auf Grund fehlender Datenbankrechte trägt die Datenbank den Namen mbadzuradb.

***Softwareentwurf***

Für die Immobilienverwaltung für die Hausverwaltung „Private Home“ werden die Grundsätze der objektorientierten Programmierung umgesetzt, d. h. Datenkapselung, Polymorphismus und Vererbung. Durch die Datenkapselung kann auf die Daten nur gezielt zugegriffen werden. Polymorphismus und Vererbung finden hauptsächlich bei der Eventbehandlung Anwendung.

Die Grundlage für den Softwareentwurf bildet die Datenbankmodellierung (siehe ). Diese Modellierung ist so ausgelegt, dass z. B. bestehende Ausstattungsmerkmale von der Hausverwaltung eigenständig erweitert werden kann. Dies wird durch die dritte Normalform erreicht. Weiterhin ist jederzeit eine Erweiterung der Datenbankstruktur möglich, welche ohne erheblichen Mehraufwand in die Anwendung aufgenommen werden kann.

Die Menüführung ist intuitiv gegliedert (siehe Anlage Entwurf Menüführung). Um der Tatsache gerecht zu werden, dass Mitarbeiter nicht alle den gleichen Umgangsstand mit solcher Software haben, können die geforderten funktionalen Bestandteile entweder über das Menü, oder aber auch über Tastaturkürzel aufgerufen werden. Um den Inhalt der einzelnen Menüpunkte auch ohne öffnen erkennen zu können, werden Tooltip-Texte den Menüpunkten hinzugefügt.

Eingabefenster werden so gestaltet, dass Sie den ergonomischen Anforderungen entsprechen. Bei Neuanlage von Immobilien wird darauf geachtet, dass im gleichen Anwendungsfenster auch alle notwendigen weiteren Eingaben (Beispielsweise Eigentümer, Wohnungen) angestoßen werden. Die Auswahl von Ausstattungsmerkmalen wird mit Comboboxen ermöglicht. Dadurch wird erreicht, dass einheitliche Informationen in der Datenbank abgespeichert werden. (siehe Anlage Entwurf Immobilienseite)

Bei einer Neuanlage einer Immobilie kann der Eigentümer per Combobox ausgewählt werden. Sollte es ein Eigentümer sein, welcher noch nicht in der Datenbank eingespeichert ist, so bieten die Comboboxen auch die Möglichkeit, die Neueingabe eines Eigentümers vorzunehmen, ohne dass die bisherigen Eintragungen verloren gehen.

Das Hilfesystem wird über ein bereits bestehendes Template eingebunden. Für das Hilfesystem muss jdom2.jar der Bibliothek hinzugefügt werden, da die Hilfedatei auf XML basiert.

Trotz ausführlichen Tests und Planungen kann es immer vorkommen, dass unerwartete Fehler auftreten. Zu diesem Zwecke wird eine Logdatei implementiert, welche bei auftretenden Fehlern die Fehlersuche erleichtert und somit beschleunigt. Diese Logdatei wird ebenfalls durch das bestehende Template MBTemplates.jar mit eingebunden.

Für das Anlegen der Datenbank wird für den Datenbankadministrator eine SQL-Datei zur Verfügung gestellt, damit erreicht wird, dass die Datenbankstruktur auch so entspricht, wie es das Anwendungsprogramm anschließend erfordert.

***Qualitätskonzept***

Die Projektdokumentation erfolgt als Projektbericht nach dem „Leitfaden zur IHK-Abschlussprüfung, Fachinformatiker/in für Anwendungsentwicklung“ in der Fassung vom 26.05.2016. Die Dokumentation des Quelltextes wird durch Kommentare innerhalb des Quelltextes vorgenommen. Außerdem werden JavaDoc-Kommentare erstellt, so dass eine ausführliche Dokumentation auch über ein erstelltes JavaDoc möglich wird.

Die Funktionalität einzelner Methoden wird mit Hilfe von Unit-Tests erfolgen. Um die Funktionalitäten im Umgang der Datenbank zu testen, wird eine Testdatenbank mit vorgegebenen Datensätzen erstellt. Alle angewandten Testmethoden und deren Resultate werden dokumentiert.

Die EN ISO 9241 und DIN EN ISO 14915 werden als Maßstab für die Oberflächengestaltung und Programmführung verwendet, um das Qualitätsmerkmal Softwareergonomie zu verwirklichen.

## Entwicklung des Konzepts mit Prototyp

### Implementierung der Datenstruktur

Auf Basis der im Sollkonzept erstellten Datenmodellierung (siehe 5.6-Anlage Datenbankmodellierung) wurde die Implementierung der Datenbankstruktur auf einem MySQL-Server vorgenommen. Auf Grund mangelnder administrativer Rechten konnte keine eigene Datenbank erstellt werden, so dass die Tabellen innerhalb der zugänglichen Datenbank mbadzuradb erstellt wurden. Der Zugriff auf die Datenbank erfolgt hierbei über phpMyAdmin. Die Tabellen wurden mit SQL-Statements (siehe -Anlage SQL-Statement Datenbankmodell) angelegt, welche durch den Autor Markus Badzura erstellt wurden. Wichtig bei der Erstellung der Datenbank war das beachten der Fremdschlüssel, damit auch alle Tabellen nacheinander mit den SQL-Statements angelegt werden konnten.

Parallel hierzu wurde gleiche Datenstruktur auf einen externen MySQL-Server eingespielt. Bereitgestellt wurde dieser MySQL-Server für den Projektzeitraum vom Autor selber. Die erforderliche Datenbank wurde dort angelegt und mit den SQL-Statements die Datenbanktabellen erstell.

Diese Vorgehensweise habe ich gewählt, um später auch die zu erstellende Konfigurationsdatei testen zu können. Einige Datensätze wurden gleich bei der Erstellung der Datenbank ebenfalls per SQL-Statement (siehe 5.7.1-Eingefügte Datensätze) eingefügt.

Die erstellte Datenbank entspricht somit der in der Entwurfsphase definierten Struktur und erfüllt die nötigen Anforderungen.

### Entwicklung des Prototypen

Um die Entwicklung des Prototypen möglichste komfortabel durchführen zu können, ist eine geeignete Entwicklungsumgebung notwendig. Der Autor hat hierfür NetBeans IDE 8.1 verwendet.

Im ersten Schritt wurde die Menüführung und die dadurch benötigten Action- und KeyListener implementiert. Die Listener werden benötigt, um auf die Menüpunktauswahl und auf die implementierten Tastaturkürzel und Funktionstastenbelegungen zu reagieren. Die Hilfefunktion und die Logdatei, welche in diesem Prototypen verwendet werden, sind aus vorangegangenen Projekten des Autors. Diese Bestandteile hat er in einer JAR-Datei als Template mit eingebunden. Dadurch sind diese benutzerfreundlichen Bestandteilen bereits in die Anwendung schnell und unkompliziert integriert worden.

Als nächsten Iterationsschritt wurde die Konfigurationsdatei (Siehe 5.8.1-Inhalt der config.txt) erstellt, welche die notwendigen Verbindungsinformationen zur MySQL-Datenbank enthält. Diese Datei wird vom Prototyp ausgelesen und die darin enthaltenen Informationen der Datenbankklasse zum Herstellen der Datenbankverbindung zur Verfügung gestellt. Entscheidend sind die Variablennamen innerhalb der config.txt. Die config.txt wird komplett ausgelesen. In einer Schleifenfunktion werden dann die Werte für die benötigten Datenbankvariablen ausgelesen. Durch den zeilenweisen Vergleich wird sichergestellt, dass beim Eintragen erzeugte Leerzeilen keine Auswirkung auf die Funktionalität des Auslesens haben. (Siehe 5.8.2-Anlage Auszug aus immo\_config.java)

Um auf die MySQL-Datenbank zugreifen zu können, wird der Datenbank-Connector für MySQL benötigt. Der Connector (mysql-connector-java-5.1.41-bin.jar) wurde der Library des Projektes hinzugefügt. Der Datenbankzugriff erfolgt über ein Datenbankobjekt. Dieses Datenbankobjekt erhält die aus der config.txt ausgelesenen Informationen zum Aufbau der Datenbankverbindung. Sämtliche Datenbankzugriffe erfolgen nur in der Klasse Immo\_db.java. Auf Grund der zu erwartenden geringen Zugriffe bestand die Möglichkeit, die Connection zu Beginn der Laufzeit der Anwendung zu starten und erst beim Schließen der Anwendung wieder zu schließen. Da es allerdings zukünftig durchaus sein kann, dass eine Vielzahl von Zugriffen auf die Datenbank erfolgen wird, hat sich der Autor entschlossen, für jeden Datenbankzugriff die Connection aufzubauen und nach Ausführung des SQL-Statements diese wieder zu Beenden. Dies spart Ressourcen auf dem SQL-Server ein.

Bei der Implementierung der Benutzeroberfläche für die Neuanlage von Immobilien wurde vom Autor eine Ungenauigkeit in der Datenbankmodellierung festgestellt. Die Immobilien sind nur durch die Straße gekennzeichnet. Mittels des SQL-Statements

ALTER TABLE immobilien ADD (plz CHAR(5)), ADD (ort CHAR(100));

wurde die Tabellenstruktur der Tabelle immobilien um die Attribute plz und ort erweitert. Sämtliche Auswahlfelder (wie Wohnlage, Eigentümer, etc) werden neben den vorhandenen Einträgen, ausgelesen aus der Datenbank, mit der Zusatzauswahl „Neue Auswahl hinzufügen“ ergänzt. Hiermit erreicht der Autor, dass die Hausverwaltung „Private Home“ bestimmte Merkmale eigenständig hinzufügen kann und somit die Auswahlmöglichkeiten auf Ihre Bedürfnisse anpassen kann. Auswahlmerkmale entfernen ist im Prototyp nicht vorgesehen und zählt auch nicht zum Funktionsumfang. Da das Auslesen der Ausstattungsmerkmale stets nach dem gleichen Algorithmus verläuft, wurde die Funktion dahingehend angepasst, dass bei Aufruf der Tabellename als Parameter übergeben wird. Das SQL-Statement wird anhand dieses Parameters angepasst.

Als erster großer Schritt in der Prototypentwicklung ist die Aufnahme neuer Immobilien anzusehen. Diese benötigt neben den config-Werten schon mehrfach eine Datenbankabfrage zum Befüllen der Auswahlfelder. Weiterhin musste gewährleistet, dass die Daten in der richtigen Reihenfolge in die Datenbank gespeichert werden, um keine Fremdschlüsselfehler, welche dazu führen würden, dass der Datensatz nicht gespeichert werden kann, auftreten. Um dies zu erreichen, wird die Möglichkeit geschaffen, bei Neueingabe einer Immobilie auch einen neuen Eigentümer mit anzulegen. Wenn eine Immobilie neu angelegt wurde, wird automatisch die Ansicht für die Eingabe neuer Wohnungen ermöglicht, ohne dass hierfür zusätzliche Aktionen seitens des Benutzers notwendig wären. Dies wurde erreicht, in dem der inaktive Tab der Eingabeformularansicht mit dem Speichern einer neuen Immobilie aktiviert wird und in den Vordergrund gestellt wird.

Damit bei der Eingabe von neuen Wohnungen gleichzeitig erkennbar ist, welche Wohnungen bereits erfasst sind, erfolgt die Anzeige über ein JTextPane mit einer HTML-Tabelle. Beim Aufruf der Tab-Seite für neue Wohnungen wird aus der Datenbank als erstes ausgelesen, ob es bereits Wohnungen mit diese Wohnungs-ID gibt. Diese Vorgehensweise wurde vom Autor gewählt, um diesen Tab auch autark nutzen zu können. Dem Tab wird nach Speicherung der neuen Immobilie die Immobiliennummer übergeben. Sollte später eine bereits bestehende Immobilie eine zusätzliche Wohnung erhalten, so wird diesem Tab dann ´von dieser Immobilie die Immobiliennummer übergeben. Die Ausstattungsmerkmale einer neuen Wohnung werden per ComboBox ausgewählt. Diese werden durch ein Array initialisiert. Das Array ist ein Ergebnis einer Datenbankabfrage. Dem Array selber erhält als ersten Eintrag den Text „Bitte wählen Sie ….“. Diese ermöglicht, dass in Feldern, in welchen keine Auswahl notwendig ist, wie z. B. bei Titel, der Index 0 abgefragt werden kann. Ohne diesen Eintrag bestünde das Problem, dass trotzdem eine Auswahl getroffen wird, nämlich der erste Eintrag aus der Datenbank. Als letzter Wert wird dem Array der Text „Neue Auswahl hinzufügen“ übergeben, damit der Benutzer die Möglichkeit hat, neue Einträge hinzuzufügen.

Um diese 2 zusätzlichen Einträge in dem Array eintragen zu können, erfolgt als erstes eine Datenbankabfrage, wie viele Einträge in der betreffenden Tabelle vorhanden sind. Anschließend wird das Array initialisiert mit dem Ergebnis der Datenbankabfrage + 2. Nachdem der Index 0 mit „Bitte wählen Sie ….“ befüllt wurde, werden die Datenbankinhalte eingelesen und anschließend der Wert „Neue Auswahl hinzufügen“ hinzugefügt. Dieser Vorgang wiederholt sich bei allen Auswahlfeldern. Wenn ein neuer Eintrag in einem Auswahlfeld erfolgt ist, werden die Inhalte des Auswahlfeldes neu initialisiert. (Siehe

Wenn nach Eingabe einer neuen Wohnung die Wohnung gespeichert wird, wurde im Quelltext eine Aktualisierung des JTextPane veranlasst.

Als nächsten Iterationsschritt wurde im Prototyp Kostenkonten erstellt. Ziel der Kostenkonten ist es, eingehende Zahlungen und Ausgaben der Immobilie zu verwalten. Außerdem wird anhand der Kostenkonten die Betriebskostenabrechnung erstellt. Hierfür wurden 3 Kostenarten für Rechnungen erstellt. Abrechenbare Kosten auf Wohnungsgröße, abrechenbare Kosten auf Wohnungsanzahl und nicht abrechenbare Kosten.

## Abnahme und Übergabe an den Kunden bzw. AG

Dem Auftraggeber wurde in einer Präsentation die Konzeption und der bis dahin vorgefertigte Prototyp vorgestellt. Alle vom Auftraggeber gestellten Fragen wurden in einer anschließenden Diskussionsrunde erörtert und erklärt. Auch wenn der Prototyp noch nicht alle Funktionalitäten erfüllt, so sind sich alle einig, dass der erstellte Prototyp das Konzept ist, welches die Hausverwaltung auch umgesetzt haben möchte.

Die vom Auftraggeber gewünschten Änderungen werden in der Weiterentwicklung des Prototyps zu einer endgültigen Anwendung noch berücksichtigt. Die Präsentation, Diskussionspunkte und Änderungswünsche wurden in einem Protokoll festgehalten.

# Auswertung der Projektarbeit

## Darstellung der Projektergebnisse

## Bewertung des Projektablaufes

Bei einer rückwirkenden Betrachtung des Projektes kann festgehalten werden, dass die für die Konzeption gestellten Anforderungen in der Planung erfüllt wurden und der Prototyp über die Hauptfunktionalitäten verfügt. Der Zeitplan konnte zu großen Teilen eingehalten werden. Abweichungen von der Zeitplanung konnten sich untereinander kompensieren.

Im Zuge des Projektes konnte der Autor wertvolle Erfahrungen bezüglich der Planung und Durchführung von Projekten sammeln. Dabei wurde besonders deutlich, von welcher großen Bedeutung eine umfangreiche Ist-Analyse mit anschließendem Soll Konzept ist. Im Projektverlauf konnten die Kenntnisse der Datenbankmodellierung und die objektorientierte Programmierung vergrößert werden.

Auch der Umgang mit bisher nicht angewandten Techniken, wie das Einrichten eines MySQL-Servers auf einem virtuellen Root war für den Autor sehr lehrreich und wird in Zukunft weiterhin vertieft.

# Anlagen, Verzeichnisse

## Anlage: Projektauftrag

|  |  |
| --- | --- |
| Projektname | Immobilienverwaltung |
| Projektleiter | Markus Badzura |
| Projektanlass | Die Verwaltung beabsichtigt den Bestand der zu verwaltenden Immobilien zu erhöhen. Um das zu leisten soll eine Datenbank-Anwendung individuell für diese Verwaltung entwickelt werden. Es soll in Vorbereitung der Entwicklung ein Entwurf mit Prototyp erarbeitet werden, um die Kosten der Entwicklung sicher zu planen. Zurzeit besteht die Verwaltung aus einer Mitarbeiterin und dem Geschäftsführer. Beide haben nicht miteinander verbundene PCs. Es soll im nächsten Jahr ein weiterer Mitarbeiter eingestellt werden.  Die Mitarbeiterin führt für jede Immobilie eine Excel-Mappe mit Tabellen über die Eigentümer, die Mieter und die einzelnen Wohnungen in der Immobilie. Außerdem führt Sie eine einfache Einnahmen- und Ausgabenrechnung über alle Mieteinnahmen, Hausgelder und Instandhaltungskosten. |
| Projektziel | **sachlich**  Es soll ein ausführlicher Entwurf (mit Prototyp) der Anwendung erstellt werden. Der Entwurf soll Basis einer zukünftigen Diskussion und Bewertung der Realisierung sein. Es sollen Lösungswege und auch dabei entstehende Kosten aufgezeigt werden.  **Kosten**  Da es sich um einen Entwurf handelt, welcher noch nicht zu einem Auftrag führen muss, müssen die Kosten minimal gehalten werden. Da Projektdurchführung während des Praktikums durchgeführt wird, fallen keine offenen Personalkosten an.  **zeitlich**  Beginn: 06.06.2017 Ende: 21.06.2017 |
| Projektressourcen | An Ressourcen steht die EDV-Anlage im Unterrichtsraum 417 zur Verfügung. Es besteht Zugriff auf einen MySQL Datenbankserver. Zur Datenanalyse wird die Excel-Mappe der Mitarbeiterin zur Verfügung gestellt. Zur Klärung aller Fragen steht Frau Lorenz über Konsulationstermine zur Verfügung |
| Auftragsbedingungen | Immobilien, Eigentümer und Mieter sind mit vollen Adressangaben und Personenangaben zu führen.  Eine Immobile ist außerdem mit folgenden Angaben zu beschreiben:   * Fertigstellungszeit * Anzahl der Wohnungen * Größe der einzelnen Wohnung * Ausstattungsmerkmale der Wohnung (analog dem Mietspiegel)   Es ist eine Betriebskostenabrechnung für die Immobilie vorzusehen. Dafür muss für jede Kostenart der Immobilie ein Kostenkonto zu führen sein. |
| Hauptaufgaben | * Projektvorbereitung (Analysierung und Planung) * Projektdurchführung (Entwurf, Codierung, Test) * Dokumentation und Abschluss |
| Teilaufgaben | 4 UE Analyse des Projektanlasses und der Aufgabenstellung  4 UE Planung Projektablauf, Ressourcen, Qualitätssicherung  4 UE Erarbeitung der Ist-Analyse  4 UE Anfertigung des Soll-Konzeptes  8 UE Entwurf der Datenbankstrukturen  8 UE Entwurf der Abläufe der Anwendung  8 UE Entwurf der Benutzerschnittstelle  8 UE Codierung und Test der Benutzerschnittstelle  8 UE Codierung und Test weiterer Methoden  4 UE Anfertigung von Gesprächsprotokollen  8 UE Zusammenstellung des Konzeptes  16 UE Abschließende Beschreibungen zum Projektbericht  8 UE Vorbereitung der Präsentation  4 UE Präsentation der Ergebnisse |
| Termine und Meilensteine | 06.06.2017 – 15:00 Uhr Vorlage Projektbericht bis Punkt 1  07.06.2017 – 15:00 Uhr Vorlage Projektbericht bis Punkt 2  09.06.2017 – 11:00 Uhr Vorlage Projektbericht bis Punkt 3.2  19.06.2017 – 15:00 Uhr Vorlage Projektbericht als pdf-Dokument  21.06.2017 Präsentation des Projektes |

Tabelle - Projektauftrag

## Anlage: Auszug aus dem Pflichtenheft

**2. Aufgaben- und Zielstellung**

**2.1 Projektziele**

In der Immobilienverwaltung sollen Informationen über Immobilien, deren Eigentümer und Mieter, sowie eine Betriebskostenabrechnung möglich sein.

*Ziele der Entwicklung sind:*

* Bearbeiten von Änderungen von Eigentümer und Mieter einer Immobilie.
* Änderungen eines Ausstattungsmerkmales
* Bearbeiten der Betriebskostenabrechnung, insbesondere das zuweisen der Kostenart auf ein Kostenkonto.
* Auslösung der Betriebskostenabrechnung für die Mieter und als Gesamtübersicht für den Eigentümer.
* Erfassung von Mieteingängen.
* Auskünfte über den aktuellen Mieterbestand.
* Auskünfte über die aktuelle Kosten

Die Rechner innerhalb der Verwaltung sind nicht vernetzt. Da ein WLAN-Router zur Verfügung steht soll der Zugriff auf eine externe, über das Internet verfügbare, mySQL-Datenbank erfolgen.

**2.2 Funktionsumfänge**

Folgende Funktionsumfänge hat die Immobilienverwaltung zu leisten:

* Löschung einer Immobilie.
* Neuanlage einer Immobilie mit den dazugehörigen Wohneinheiten und Ausstattungsmerkmalen und des Eigentümers
* Ändern eines Eigentümers
* Anlegen und Ändern von Mietern
* Ändern der Ausstattungsmerkmale einer Wohnung
* Bearbeiten des Kostenkontos einer Immobilie
* Übersicht aller Immobilien
* Übersicht aller Mieter einer Immobilie
* Verteilung der verteilungsfähigen Kosten auf die einzelnen Mieter an Hand der Größe einer Wohnung
* Verteilung der verteilungsfähigen Kosten auf die einzelnen Mieter an Hand der Anzahl der Wohnungen innerhalb der Immobilie.
* Datensicherung automatisch und manuell.
* Standortunabhängigkeit des mySQL-Servers garantieren

## Anlage: Leistungsangebot

Comhard Übungsfirma GmbH

Möllendorfstraße 52

10367 Berlin

Berlin, 06.05.2017

Hausverwaltung

„Private Home“

Möllendorfstraße 52

10367 Berlin

Leistungsangebot

Sehr geehrte Damen und Herren,

für die Erstellung eines ausführlichen Entwurfes mit Prototyp für eine Datenbankbasierenden Anwendung für Ihre Immobilienverwaltung, können wir Ihnen folgendes Leistungsangebot unterbreiten.

Die in diesem Angebot angegebenen Preise gelten bis 20.05.2017.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Menge | Einheit | Einzelsatz |  |  |
| 72 | Leistungsstunden | 65,00 € |  | 4680,00 € |
|  |  |  | gesamt netto | 4680,00 € |
|  |  |  | zzgl.19 % USt | 889,20 € |
|  |  |  | gesamt brutto | 5569,20 € |

Die Zahlungsbedingungen entnehmen Sie bitte unseren aktuellen AGBs.

Mit freundlichen Grüßen

## Anlage: Datenanalyse

***Datenanalyse Excel-Arbeitsmappe Hausverwaltung „Private Home“***

Auf Grund der zur Verfügung gestellten Arbeitsmappe wurden folgende Daten ermittelt.

|  |  |  |
| --- | --- | --- |
| **Daten** | **Datentyp** | **Bemerkung** |
| Immobilienbezeichnung | String (100) | Straßenname und Nummer der Immobilie |
| Wohnlage | String(50) | Wohnlage gemäß Wohnlagenkarte Berlin |
| Eigentümer Name | String (100) | Name, Vorname und Anschrift des Eigentümers |
| Eigentümer Straße | String (100) |
| Eigentümer Wohnort | String (100) |
| Wohnungsnummer | String (20) | Wohnungsnummer innerhalb der Immobilie |
| qm | Double (2 NKSt) | Wohnfläche |
| Bad | String (100) | Ausstattungsmerkmale |
| Zimmer, Küche | String (100) | Übernommen aus Mietvertrag |
| Mieter | String (100) | Name und Vorname des Mieters |
| Anzahl der Personen | Integer |  |
| Kaltmiete | Double (2 NKSt) | Übernommen aus Mietvertrag |
| Nebenkosten | Double (2 NKSt) |
| Miete gesamt | Double (2 NKSt) | Ermittelt aus Kaltmiete + Nebenkosten |
| Datum | Date | Datum des Zahlungsein- oder ausganges |
| Bezeichnung | String(100) | Art der Zahlung oder Ausgabe |
| Einnahme | Double (2 NKSt) | Betrag der Einnahme |
| Ausgabe | Double (2 NKSt) | Betrag der Ausgabe |
| Aktueller Stand Kostenkonto | Double (2 NKSt) | Ermittelt aus Einnahmen und Ausgaben |

Tabelle : Datenanalyse Hausverwaltung "Private Home"

## Anlage: Nutzwertanalyse

Für die Nutzwertanalyse sind folgende Kriterien bewertet worden:

* Anschaffungskosten
* Folgekosten
* Zugriffsmöglichkeiten
* Strukturänderung Hard- und Software

Als Bewertungsgrundlage sind folgende Kriterien betrachtet worden.

1. Anschaffungskosten sollen so gering wie möglich sein
2. Folgekosten sind ebenfalls gering zu halten
3. Da ein zusätzlicher Mitarbeiter geplant ist, soll von mehreren PCs erfolgen können.
4. Zusätzliche Software und Hardware soll vermieden werden
5. Die Wartung der Datenbank und der Software

Mit Hilfe der Bewertungskriterien werden in einem Präferenzdreieck die Gewichtungsfaktoren ermittelt.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kriterium | a | b | c | d | e | Summe | Prozent |
| a |  | 2 | 2 | 1 | 2 | 7 | 35 % |
| b | 0 |  | 0 | 1 | 1 | 2 | 10 % |
| c | 0 | 2 |  | 2 | 1 | 5 | 25 % |
| d | 1 | 1 | 0 |  | 2 | 4 | 20 % |
| e | 0 | 1 | 1 | 0 |  | 2 | 10 % |
|  | Summen: | | | | | 20 | 100 % |

Tabelle : Präferenzdreieck

Die einzelnen Kriterien wurden anschließend bewertet, so dass folgende Nutzwertanalyse erstellt wurde.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kriterium | Gewich-tung | H2  (bew[[1]](#footnote-1)) | Netzwerk  (bew) | MySql  (bew> | H2  (gew[[2]](#footnote-2)) | Netzwerk  (gew) | MySql  (gew) |
| Anschaffungskosten | 35 % | 10 | 5 | 10 | 3,5 | 1,75 | 3.5 |
| Folgekosten | 10 % | 10 | 10 | 5 | 1 | 1 | 0,5 |
| Zugriff | 25 % | 3 | 7 | 9 | 0,75 | 1,75 | 2,25 |
| Zusatz HW/SW | 20 % | 10 | 5 | 10 | 2 | 1 | 2 |
| Wartung | 10 % | 3 | 5 | 7 | 0,3 | 0,5 | 0,7 |
| Summe |  |  |  |  | 7,55 | 6 | 8,95 |

Tabelle : Nutzwertanalyse

Auf Grund der Nutzwertanalyse wurde entschieden, dass im weiteren Projektverlauf ein Konzept basierend auf einem MySQL-Server favorisiert wird.

## Anlage Datenbankmodellierung

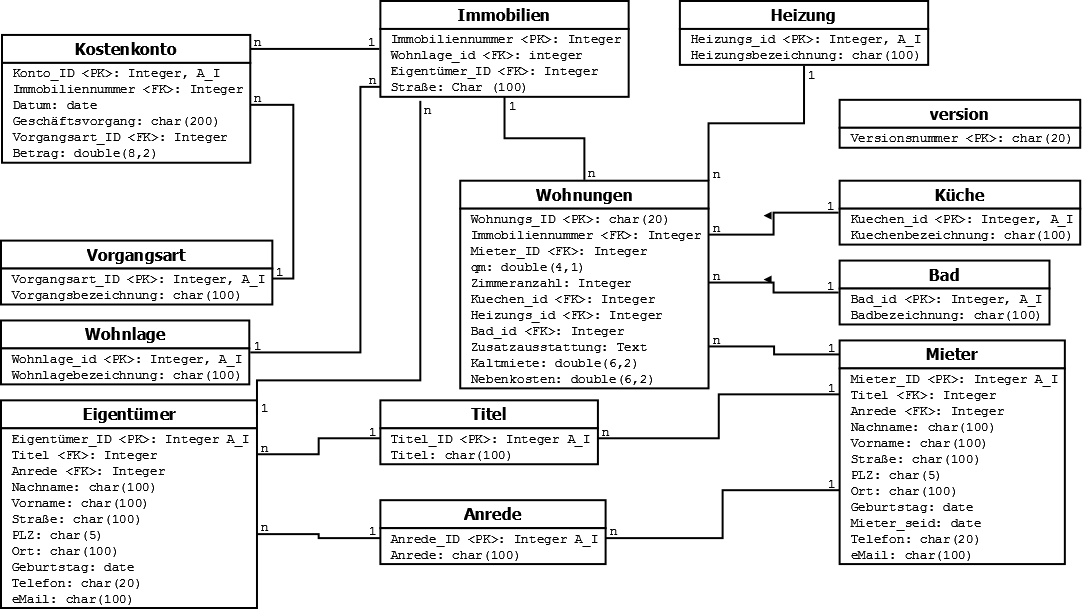


Abbildung : Datenbankmodellierung Immobilienverwaltung

## Anlage SQL-Statement Datenbankmodell

# Tabelle Vorgangsart, Wohnlage, Heizung, Küche und Bad anlegen

CREATE TABLE vorgangsart(

Vorgangsart\_ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

Vorgangsbezeichnung CHAR(100));

CREATE TABLE wohnlage(

Wohnlage\_ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

Wohnlagebezeichnung CHAR(100));

CREATE TABLE heizung(

heizungs\_ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

heizungbezeichnung CHAR(100));

CREATE TABLE kuechen(

kuechen\_ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

kuechenbezeichnung CHAR(100));

CREATE TABLE bad(

bad\_ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

badbezeichnung CHAR(100));

#Tabelle Version anlegen

CREATE TABLE version(

versionsnummer CHAR(20) PRIMARY KEY);

#Tabelle Titel und Anrede anlegen

CREATE TABLE titel(

titel\_ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

titelbezeichnung CHAR(100));

CREATE TABLE anrede(

anrede\_ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

anredebezeichnung CHAR(100));

#Tabelle Eigentümer anlegen

CREATE TABLE eigentuemer(

Eigentuemer\_ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

titel INTEGER,

anrede INTEGER,

nachname CHAR(100),

vorname CHAR(100),

strasse CHAR(100),

plz CHAR(5),

ort CHAR(100),

geburtstag DATE,

telefon CHAR(20),

email CHAR(100),

FOREIGN KEY (titel) REFERENCES titel(titel\_id),

FOREIGN KEY (anrede) REFERENCES anrede(anrede\_id));

#Tabelle Mieter anlegen

CREATE TABLE mieter(

mieter\_ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

titel INTEGER,

anrede INTEGER,

nachname CHAR(100),

vorname CHAR(100),

strasse CHAR(100),

plz CHAR(5),

ort CHAR(100),

geburtstag DATE,

mieter\_seid DATE,

telefon CHAR(20),

email CHAR(100),

FOREIGN KEY (titel) REFERENCES titel(titel\_id),

FOREIGN KEY (anrede) REFERENCES anrede(anrede\_id));

#Tabelle Immobilien anlegen

CREATE TABLE immobilien(

immobiliennummer INTEGER PRIMARY KEY AUTO\_INCREMENT,

Wohnlage\_ID INTEGER,

Eigentuemer\_ID INTEGER,

Strasse CHAR(100),

plz CHAR(5),

ort CHAR(100),

FOREIGN KEY (wohnlage\_ID) REFERENCES wohnlage(wohnlage\_id),

FOREIGN KEY (eigentuemer\_ID) REFERENCES eigentuemer(eigentuemer\_id));

#Tabelle Kostenkonto anlegen

CREATE TABLE kostenkonto(

Konto\_ID INTEGER PRIMARY KEY AUTO\_INCREMENT,

immobiliennummer INTEGER,

Datum DATE,

Geschaeftsvorgang CHAR(200),

Vorgangs\_ID INTEGER,

Betrag DOUBLE(8,2),

FOREIGN KEY (immobiliennummer) REFERENCES immobilien(immobiliennummer));

#Tabelle Wohnungen anlegen

CREATE TABLE wohnungen(

wohnungs\_ID CHAR(20) PRIMARY KEY,

immobiliennummer INTEGER,

mieter\_ID INTEGER,

qm DOUBLE(4,1),

zimmeranzahl INTEGER,

kuechen\_ID INTEGER,

heizungs\_ID INTEGER,

bad\_ID INTEGER,

zusatzausstattung TEXT,

kaltmiete DOUBLE(6,2),

nebenkosten DOUBLE(6,2),

FOREIGN KEY (immobiliennummer) REFERENCES immobilien(immobiliennummer),

FOREIGN KEY (mieter\_id) REFERENCES mieter(mieter\_id),

FOREIGN KEY (kuechen\_id) REFERENCES kuechen(kuechen\_id),

FOREIGN KEY (heizungs\_id) REFERENCES heizung(heizungs\_id),

FOREIGN KEY (bad\_id) REFERENCES bad(bad\_id));

### Eingefügte Datensätze

# Tabelle Vorgangsart befüllen

INSERT INTO vorgangsart (vorgangsbezeichnung) VALUES

('Kaltmiete'),('Nebenkosten'),('Nebenkosten Umlage qm'),('Nebenkosten Umlage Wohnungsanzahl'),('Nicht umlagefähige Kosten'),('Auszahlung an Eigentümer');

# Tabelle Wohnlage befüllen

INSERT INTO wohnlage (wohnlagebezeichnung) VALUES

('einfach'),('mittel'),('gut');

#Tabelle Titel befüllen

INSERT INTO titel (titelbezeichnung) VALUES

('Dr. agr.'),('Dr. biol. anim.'),('Dr. biol. hom.'),('Dr. nat. med.'),('Dr. phil.');

#Tabelle Anrede befüllen

INSERT INTO anrede (anredebezeichnung) VALUES

('Frau'),('Herr'),('Herr Landrat'),('Frau Regierungsrätin'),('Herr Minister');

# Tabelle version befüllen

INSERT INTO version (versionsnummer) VALUES

('1.0');

# Tabelle Heizung befüllen

INSERT INTO heizung (heizungbezeichnung) VALUES

('Zentralheizung'),('Kaminofen');

# Tabelle Küche befüllen

INSERT INTO kuechen (kuechenbezeichnung) VALUES

('Küche'),('Küchenecke'),('Einbauküche');

# Tabelle Bad befüllen

INSERT INTO bad (badbezeichnung) VALUES

('WC'),('Wanne, WC'),('Dusche, WC'),('Dusche, Wanne, WC');

## Anlage config-Datei

Die Konfiguration der Datenbankverbindung benötigt die Informationen aus der config.txt. Die Informationen werden mit der Klasse immo\_config.java ausgelesen.

### Inhalt der config.txt

# hostname

# IP oder Adresse zum MySQL-Server

hostname: <<hier Adresse eintragen>>

# MySQL-Port (Standardport: 3306)

port: <<hier Port eintragen>>

# MySQL-Datenbank-Username

username: <<Hier Username eintragen>>

# MySQL-Datenbank-Benutzerpasswort

password: <<Hier Passwort eintragen>>

# Datenbankname

dbname: <<Hier Datenbanknamen eintragen>>

### Anlage Auszug aus immo\_config.java

public void readConfig()

{

try

{

br = new BufferedReader(new InputStreamReader(

new FileInputStream(speicherort),StandardCharsets.UTF\_8));

while ((inhalt=br.readLine())!= null)

{

if (!"".equals(inhalt))

{

try

{

if("hostname:".equals(inhalt.substring(0,9).toLowerCase()))

HOSTNAME = (inhalt.substring(10).trim());

if("port:".equals(inhalt.substring(0,5).toLowerCase()))

PORT = (inhalt.substring(6).trim());

if("username:".equals(inhalt.substring(0,9).toLowerCase()))

USER = (inhalt.substring(10).trim());

if("password:".equals(inhalt.substring(0,9).toLowerCase()))

PASSWORD = (inhalt.substring(10).trim());

if("dbname:".equals(inhalt.substring(0,7).toLowerCase()))

DBNAME = (inhalt.substring(8).trim());

}

catch(StringIndexOutOfBoundsException

| NullPointerException e)

{

el.schreibe(e.toString(), "immo\_config");

}

}

}

}

catch(IOException e)

{

el.schreibe(e.toString(), "immo\_config");

JOptionPane.showMessageDialog(null, "Die Datei config.txt kann im

Verzeichnis dieser Anwendung nicht gefunden werden.\n\nBitte

kopieren Sie die Datei config.txt in Verzeichnis, in welchem sich auch

diese Anwendung befindet, und starten Sie die Anwendung

neu.","Programm beenden", JOptionPane.OK\_OPTION);

System.exit(0);

}

}

## Anlage Auszug Quelltext Neuanlage Wohnung mit Anlegen eines neuen Mieters

Die Neuanlage einer Immobilie erfolgt nach ähnlicher Syntax wie der folgende Quelltextauszug für die Neuanlage einer Wohnung und dem Anlegen eines neuen Mieters. Die Auszüge des Quelltextes wurden chronologisch angeordnet.

### Erstellen der Benutzeroberfläche

/\*\*

\* Panel für Tab Neue Wohnung

\* @author Markus Badzura

\* @since 1.0.001

\*/

private void setTabNewWohnung()

{

int x1 = 10;

int x2 = SCREENSIZE.width/4;

int x3 = SCREENSIZE.width/2;

int breite = x2-20;

int hoehe = 25;

jp\_wohnungen = new JPanel();

jp\_wohnungen.setSize(this.getSize());

jp\_wohnungen.setLayout(null);

jl\_whgImmoNummer = new JLabel("Immobiliennummer: "+Integer.toString(aktuelleImmo));

jl\_whgImmoNummer.setFont(jl\_whgImmoNummer.getFont().deriveFont(14f));

jl\_whgImmoNummer.setBounds(x1,10,breite,hoehe);

jl\_whgImmoAnschrift = new JLabel(database.getImmoAnschrift(aktuelleImmo));

jl\_whgImmoAnschrift.setFont(jl\_whgImmoAnschrift.getFont().deriveFont(14f));

jl\_whgImmoAnschrift.setBounds(x2,10,breite,hoehe);

jl\_whgId = new JLabel("Wohnungskennzeichnung: ",JLabel.RIGHT);

jl\_whgId.setFont(jl\_whgId.getFont().deriveFont(14f));

jl\_whgId.setBounds(x1,50,breite,hoehe);

jl\_whgMieter = new JLabel("Mieter: ",JLabel.RIGHT);

jl\_whgMieter.setFont(jl\_whgMieter.getFont().deriveFont(14f));

jl\_whgMieter.setBounds(x1,80,breite,hoehe);

jl\_whgQm = new JLabel("Quadratmeter: ",JLabel.RIGHT);

jl\_whgQm.setFont(jl\_whgQm.getFont().deriveFont(14f));

jl\_whgQm.setBounds(x1,110,breite,hoehe);

jl\_whgZimmer = new JLabel("Zimmer: ",JLabel.RIGHT);

jl\_whgZimmer.setFont(jl\_whgZimmer.getFont().deriveFont(14f));

jl\_whgZimmer.setBounds(x1,140,breite,hoehe);

jl\_whgKuechenid = new JLabel("Küche: ",JLabel.RIGHT);

jl\_whgKuechenid.setFont(jl\_whgKuechenid.getFont().deriveFont(14f));

jl\_whgKuechenid.setBounds(x1,170,breite,hoehe);

jl\_whgHeizungid = new JLabel("Heizung: ",JLabel.RIGHT);

jl\_whgHeizungid.setFont(jl\_whgHeizungid.getFont().deriveFont(14f));

jl\_whgHeizungid.setBounds(x1,200,breite,hoehe);

jl\_whgBadid = new JLabel("Bad: ",JLabel.RIGHT);

jl\_whgBadid.setFont(jl\_whgBadid.getFont().deriveFont(14f));

jl\_whgBadid.setBounds(x1,230,breite,hoehe);

jl\_whgZusatz = new JLabel("Zusatzausstattung: ",JLabel.RIGHT);

jl\_whgZusatz.setFont(jl\_whgZusatz.getFont().deriveFont(14f));

jl\_whgZusatz.setBounds(x1,260,breite,hoehe);

jl\_whgKaltmiete = new JLabel("Kaltmiete: ",JLabel.RIGHT);

jl\_whgKaltmiete.setFont(jl\_whgKaltmiete.getFont().deriveFont(14f));

jl\_whgKaltmiete.setBounds(x1,290,breite,hoehe);

jl\_whgNebenkosten = new JLabel("Nebenkosten: ",JLabel.RIGHT);

jl\_whgNebenkosten.setFont(jl\_whgNebenkosten.getFont().deriveFont(14f));

jl\_whgNebenkosten.setBounds(x1,320,breite,hoehe);

jb\_whgSave = new JButton("Speichern");

jb\_whgSave.setFont(jb\_whgSave.getFont().deriveFont(14f));

jb\_whgSave.addActionListener(this);

jb\_whgSave.setBounds(x1,370,breite,hoehe);

jtf\_whgId = new JTextField();

jtf\_whgId.setFont(jtf\_whgId.getFont().deriveFont(14f));

jtf\_whgId.setBounds(x2,50,breite,hoehe);

jcb\_whgMieter = new JComboBox();

fillMieter();

jcb\_whgMieter.setFont(jcb\_whgMieter.getFont().deriveFont(14f));

jcb\_whgMieter.setBounds(x2,80,breite,hoehe);

jcb\_whgMieter.addItemListener(this);

jtf\_whgQm = new JTextField();

jtf\_whgQm.setFont(jtf\_whgQm.getFont().deriveFont(14f));

jtf\_whgQm.setBounds(x2,110,breite,hoehe);

jtf\_whgZimmer = new JTextField();

jtf\_whgZimmer.setFont(jtf\_whgZimmer.getFont().deriveFont(14f));

jtf\_whgZimmer.setBounds(x2,140,breite,hoehe);

jcb\_whgKueche = new JComboBox();

fillKueche();

jcb\_whgKueche.setFont(jcb\_whgKueche.getFont().deriveFont(14f));

jcb\_whgKueche.setBounds(x2,170,breite,hoehe);

jcb\_whgHeizung = new JComboBox();

fillHeizung();

jcb\_whgHeizung.setFont(jcb\_whgHeizung.getFont().deriveFont(14f));

jcb\_whgHeizung.setBounds(x2,200,breite,hoehe);

jcb\_whgBad = new JComboBox();

fillBad();

jcb\_whgBad.setFont(jcb\_whgBad.getFont().deriveFont(14f));

jcb\_whgBad.setBounds(x2,230,breite,hoehe);

jtf\_whgZusatz = new JTextField();

jtf\_whgZusatz.setFont(jtf\_whgZusatz.getFont().deriveFont(14f));

jtf\_whgZusatz.setBounds(x2,260,breite,hoehe);

jtf\_whgKaltmiete = new JTextField();

jtf\_whgKaltmiete.setFont(jtf\_whgKaltmiete.getFont().deriveFont(14f));

jtf\_whgKaltmiete.setBounds(x2,290,breite,hoehe);

jtf\_whgNebenkosten = new JTextField();

jtf\_whgNebenkosten.setFont(jtf\_whgNebenkosten.getFont().deriveFont(14f));

jtf\_whgNebenkosten.setBounds(x2,320,breite,hoehe);

jb\_whgClear = new JButton ("zurücksetzen");

jb\_whgClear.setFont(jb\_whgClear.getFont().deriveFont(14f));

jb\_whgClear.addActionListener(this);

jb\_whgClear.setBounds(x2,370,breite,hoehe);

jtp\_whgUebersicht = new JTextPane();

jtp\_whgUebersicht.setEditorKit(eKit);

jtp\_whgUebersicht.setBounds(x3+20,10,x3-40,SCREENSIZE.height);

fillWhgUebersicht();

jp\_wohnungen.add(jtp\_whgUebersicht);

jp\_wohnungen.add(jb\_whgClear);

jp\_wohnungen.add(jtf\_whgNebenkosten);

jp\_wohnungen.add(jtf\_whgKaltmiete);

jp\_wohnungen.add(jtf\_whgZusatz);

jp\_wohnungen.add(jcb\_whgBad);

jp\_wohnungen.add(jcb\_whgHeizung);

jp\_wohnungen.add(jcb\_whgKueche);

jp\_wohnungen.add(jtf\_whgZimmer);

jp\_wohnungen.add(jtf\_whgQm);

jp\_wohnungen.add(jcb\_whgMieter);

jp\_wohnungen.add(jtf\_whgId);

jp\_wohnungen.add(jb\_whgSave);

jp\_wohnungen.add(jl\_whgNebenkosten);

jp\_wohnungen.add(jl\_whgKaltmiete);

jp\_wohnungen.add(jl\_whgZusatz);

jp\_wohnungen.add(jl\_whgBadid);

jp\_wohnungen.add(jl\_whgHeizungid);

jp\_wohnungen.add(jl\_whgKuechenid);

jp\_wohnungen.add(jl\_whgZimmer);

jp\_wohnungen.add(jl\_whgQm);

jp\_wohnungen.add(jl\_whgMieter);

jp\_wohnungen.add(jl\_whgId);

jp\_wohnungen.add(jl\_whgImmoAnschrift);

jp\_wohnungen.add(jl\_whgImmoNummer);

}

### Methoden zum Befüllen der Auswahlfelder Mieter, Küchen, Heizung, Bad

/\*\*

\* Befüllen des Auswahlfeldes Mieter

\* @author Markus Badzura

\* @since 1.0.001

\*/

private void fillMieter()

{

String[] mieter = database.getMieter();

for (int i = 0; i<mieter.length;i++)

{

jcb\_whgMieter.addItem(mieter[i]);

}

jcb\_whgMieter.addItemListener(this);

jcb\_whgMieter.repaint();

}

/\*\*

\* Befüllen des Auswahlfeldes Küche

\* @author Markus Badzura

\* @since 1.0.001

\*/

private void fillKueche()

{

String[] kueche = database.getBezeichnungen("kuechen");

for (int i = 0; i<kueche.length;i++)

{

jcb\_whgKueche.addItem(kueche[i]);

}

}

/\*\*

\* Befüllen des Auswahlfeldes Heizung

\* @author Markus Badzura

\* @since 1.0.001

\*/

private void fillHeizung()

{

String[] heizung = database.getBezeichnungen("heizung");

for (int i = 0; i<heizung.length;i++)

{

jcb\_whgHeizung.addItem(heizung[i]);

}

}

/\*\*

\* Befüllen des Auswahlfeldes Bad

\* @author Markus Badzura

\* @since 1.0.001

\*/

private void fillBad()

{

String[] bad = database.getBezeichnungen("bad");

for (int i = 0; i<bad.length;i++)

{

jcb\_whgBad.addItem(bad[i]);

}

}

### Methode Abfrage Datenbank zum Befüllen Auswahlfeld Mieter

/\*\*

\* Vorhandene Mieter auslesen für Auswahlfeld

\* @return mieter String[] mit vorhanden Mieter mit Vor- und Nachnamen

\* @author Markus Badzura

\* @since 1.0.001

\*/

public String[] getMieter()

{

int anzahl = 0;

String query = "SELECT Count(\*) FROM mieter";

openDB();

ResultSet resultSet = selectDB(query);

try

{

while (resultSet.next())

{

anzahl = resultSet.getInt("Count(\*)");

}

}

catch(SQLException se)

{

}

String[] mieter = new String[(anzahl)+2];

query = "SELECT CONCAT (nachname,' ',vorname) AS bez FROM mieter";

mieter[0] = "Bitte wählen...";

resultSet = selectDB(query);

try

{

int zaehler = 1;

while (resultSet.next())

{

mieter[zaehler] = resultSet.getString("bez");

zaehler++;

}

mieter[zaehler] = "Neue Auswahl hinzufügen";

}

catch(SQLException se)

{

}

closeDB();

return mieter;

}

### Methode zum Auslesen der Datenbanktabellen Kuechen, Bad, Heizung

/\*\*

\* Auslesen der bezeichnungen

\* @return bezeichnungen String[]

\* @param tabelle String mit Tabellenname (wohnlage, heizung, küchen, bad, anrede, titel)

\* @author Markus Badzura

\* @since 1.0.001

\*/

public String[] getBezeichnungen(String tabelle)

{

int anzahl = 0;

String query = "SELECT Count(\*) FROM "+tabelle;

openDB();

resultSet = selectDB(query);

try

{

while (resultSet.next())

{

anzahl = resultSet.getInt("Count(\*)");

}

}

catch(SQLException se)

{

}

String[] bezeichnungen = new String[(anzahl)+2];

bezeichnungen[0] = "Bitte wählen ...";

query = "SELECT "+tabelle+"bezeichnung AS bez FROM "+tabelle;

resultSet = selectDB(query);

try

{

int zaehler = 1;

while (resultSet.next())

{

bezeichnungen[zaehler] = resultSet.getString("bez");

zaehler++;

}

bezeichnungen[zaehler] = "Neue Auswahl hinzufügen";

}

catch(SQLException se)

{

}

closeDB();

return bezeichnungen;

### Methoden zum Öffnen der Datenbankverbindung, Durchführung der Select-Abfragen und Schließen der Datenbankverbindung

private void openDB()

{

try

{

String url = "jdbc:mysql://"+hostname+":"+port+"/"+dbname;

connection = DriverManager.getConnection(url, user, password);

}

catch(SQLException se)

{

JOptionPane.showMessageDialog(null,"Datenbankverbindung kann nicht hergestellt werden",

"Datenbankfehler",JOptionPane.OK\_OPTION);

}

}

/\*\*

\* Datenbankverbindung schließen

\* @author Markus Badzura

\* @since 1.0.001

\*/

private void closeDB()

{

try

{

statement.close();

resultSet.close();

connection.close();

}

catch(SQLException e)

{

JOptionPane.showMessageDialog(null,"Datenbankverbindung konnte nicht beendet werden",

"Datenbankfehler",JOptionPane.OK\_OPTION);

}

}

/\*\*

\* Select-Anweisungen ausführen

\* @param query Select-Anweisung

\* @author Markus Badzura

\* @since 1.0.001

\*/

private ResultSet selectDB(String query)

{

try

{

statement = connection.createStatement();

resultSet = statement.executeQuery(query);

}

catch(SQLException e)

{

}

return resultSet;

}

### Methode zum Befüllen der Wohnungsübersicht bei Neueingabe einer Wohnung

/\*\*

\* Erstinitialisierung Übersicht aktuelle Immobilie bei Neuer Wohnung

\* @author Markus Badzura

\* @since 1.0.001

\*/

private void fillWhgUebersicht()

{

al\_wohnungen = database.getWohnungsUebersicht(aktuelleImmo);

Immo\_wohnungen iwTemp = new Immo\_wohnungen();

String whgUebersicht =

"<html>"

+ "<table>"

+ "<thead>"

+ "<tr>"

+ "<th>Wohnungsnummer</th>"

+ "<th>Mieter</th>"

+ "<th>qm</th>"

+ "<th>Zimmer</th>"

+ "<th>Kaltmiete</th>"

+ "<th>Nebenkosten</th>"

+ "</tr>"

+ "<tbody>";

for(int i = 0; i<al\_wohnungen.size();i++)

{

iwTemp = (Immo\_wohnungen) al\_wohnungen.get(i);

whgUebersicht = whgUebersicht +

"<tr><td>"+iwTemp.getWohnungsid()+"</td>"

+ "<td>"+iwTemp.getMieter()+"</td>"

+ "<td>"+iwTemp.getQm()+"</td>"

+ "<td>"+iwTemp.getZimmer() +"</td>"

+ "<td>"+iwTemp.getKaltmiete()+"</td>"

+ "<td>"+iwTemp.getNebenkosten()+"</td></tr>";

}

whgUebersicht = whgUebersicht + "</tbody></table></html>";

jtp\_whgUebersicht.setText(whgUebersicht);

}

### Methode zum Auslesen der vorhanden Wohnungen einer Immobilien-ID aus der Datenbank

/\*\*

\* Ermitteln der eingetragenen Wohnungen einer Immobilie

\* Für jede Wohnung wird ein Wohnungsobjekt erstellt und in

\* einer ArrayList abgelegt.

\* @param aktuelleImmobilie Integer ID der Immobilie

\* @return wohnungen ArrayList mit Wohnungsobjekten

\*/

public ArrayList getWohnungsUebersicht(int aktuelleImmobilie)

{

String wohnungsid, name, kuechenbezeichnung, heizungbezeichnung,

badbezeichnung, zusatzausstattung;

double qm,zimmeranzahl, kaltmiete, nebenkosten;

ArrayList wohnungen = new ArrayList();

String query = "SELECT wohnungs\_id, Concat(nachname,' ',vorname) "

+ "as name, qm, zimmeranzahl, kuechenbezeichnung, "

+ "heizungbezeichnung,badbezeichnung, zusatzausstattung,"

+ " kaltmiete, nebenkosten "

+ "FROM wohnungen LEFT JOIN mieter ON wohnungen.mieter\_ID "

+ "= mieter.mieter\_ID "

+ "LEFT JOIN bad ON bad.bad\_id = wohnungen.bad\_id "

+ "LEFT JOIN kuechen on kuechen.kuechen\_id "

+ "= wohnungen.kuechen\_id "

+ "LEFT JOIN heizung on heizung.heizungs\_id = "

+ "wohnungen.heizungs\_id "

+ "WHERE immobiliennummer = "+aktuelleImmobilie;

openDB();

resultSet = selectDB(query);

try

{

while (resultSet.next())

{

wohnungsid = resultSet.getString("wohnungs\_id");

name = resultSet.getString("name");

qm = resultSet.getDouble("qm");

zimmeranzahl = resultSet.getDouble("zimmeranzahl");

kuechenbezeichnung = resultSet.getString("kuechenbezeichnung");

heizungbezeichnung = resultSet.getString("heizungbezeichnung");

badbezeichnung = resultSet.getString("badbezeichnung");

zusatzausstattung = resultSet.getString("zusatzausstattung");

kaltmiete = resultSet.getDouble("kaltmiete");

nebenkosten = resultSet.getDouble("nebenkosten");

Immo\_wohnungen temp = new Immo\_wohnungen(wohnungsid, name, qm,

zimmeranzahl, kuechenbezeichnung, heizungbezeichnung,

badbezeichnung, zusatzausstattung, kaltmiete, nebenkosten);

wohnungen.add(temp);

}

}

catch(SQLException se)

{

}

closeDB();

return wohnungen;

}

### Immo\_wohnungen.java Klasse zur Objekterstellung von Wohnungen

package Immobilienverwaltung;

/\*\*

\* PSA-Projekt Immobilienverwaltung

\* Objektklasse Wohnungen

\* @author Markus Badzura

\* @version 1.0.001

\*/

public class Immo\_wohnungen

{

private String wohnungsid;

private String mieter;

private double qm;

private double zimmer;

private String kueche;

private String heizung;

private String bad;

private String zusatz;

private double kaltmiete;

private double nebenkosten;

/\*\*

\* Standardkonstruktur zur Einzelobjektabfrage

\* @author Markus Badzura

\* @since 1.0.001

\*/

Immo\_wohnungen(){}

/\*\*

\* Konstrukur für das Erstellen eines Wohnungsobjektes

\* @param wohnungsid String Wohnungskennung

\* @param mieter String Nachname und Vorname des Mieters

\* @param qm Double Quadratmeteranzahl der Wohnung

\* @param zimmer Double Zimmeranzahl der Wohnung

\* @param kueche String Küchenbezeichnung der Wohnung

\* @param heizung String Heizungsbezeichnung der Wohnung

\* @param bad String Badbezeichnung der Wohnung

\* @param zusatz String Zusatzausstattung der Wohnung

\* @param kaltmiete Double Kaltmiete der Wohnung

\* @param nebenkosten Double Nebenkosten der Wohnung

\* @author Markus Badzura

\* @since 1.0.001

\*/

Immo\_wohnungen(String wohnungsid, String mieter, double qm, double zimmer,

String kueche, String heizung, String bad, String zusatz,

double kaltmiete, double nebenkosten)

{

this.wohnungsid = wohnungsid;

this.mieter = mieter;

this.qm = qm;

this.zimmer = zimmer;

this.kueche = kueche;

this.heizung = heizung;

this.bad = bad;

this.zusatz = zusatz;

this.kaltmiete = kaltmiete;

this.nebenkosten = nebenkosten;

}

/\*\*

\* Übergabe der Wohnungskennzeichnung

\* @return wohnungsid String

\* @author Markus Badzura

\* @since 1.0.001

\*/

public String getWohnungsid()

{

return wohnungsid;

}

/\*\*

\* Übergabe Nach- und Vorname des Mieters

\* @return mieter String

\* @author Markus Badzura

\* @since 1.0.001

\*/

public String getMieter()

{

return mieter;

}

/\*\*

\* Übergabe der Quadramtmeterzahl

\* @return qm Double

\* @author Markus Badzura

\* @since 1.0.001

\*/

public double getQm()

{

return qm;

}

/\*\*

\* Übergabe der Zimmeranzahl

\* @return zimmer Double

\* @author Markus Badzura

\* @since 1.0.001

\*/

public double getZimmer()

{

return zimmer;

}

/\*\*

\* Übergabe der Küchenbezeichnung

\* @return kueche String

\* @author Markus Badzura

\* @since 1.0.001

\*/

public String getKueche()

{

return kueche;

}

/\*\*

\* Übergabe der Heizungsbezeichnung

\* @return heizung String

\* @author Markus Badzura

\* @since 1.0.001

\*/

public String getHeizung()

{

return heizung;

}

/\*\*

\* Übergabe der Badbezeichnung

\* @return bad String

\* @author Markus Badzura

\* @since 1.0.001

\*/

public String getBad()

{

return bad;

}

/\*\*

\* Übergabe der Zusatzausstattung

\* @return zusatz String

\* @author Markus Badzura

\* @since 1.0.001

\*/

public String getZusatz()

{

return zusatz;

}

/\*\*

\* Übergabe der Kaltmiete

\* @return kaltmiete Double

\* @author Markus Badzura

\* @since 1.0.001

\*/

public double getKaltmiete()

{

return kaltmiete;

}

/\*\*

\* Übergabe der Nebenkosten

\* @return nebenkosten String

\* @author Markus Badzura

\* @since 1.0.001

\*/

public double getNebenkosten()

{

return nebenkosten;

}

}

### Auszug ItemListener, wenn Auswahl „Neue Auswahl hinzufügen“ beim Auswahlfeld Mieter

if (e.getSource() == jcb\_whgMieter)

{

if("Neue Auswahl hinzufügen".equals(jcb\_whgMieter.getSelectedItem().toString()))

{

jcb\_whgMieter.setSelectedIndex(0);

showNewMieter();

}

}

### Auszug ActionListener zum Speichern und Resetten der Eingaben einer neuen Wohnung

// Button Neue Wohnung speichern

if (e.getSource() == jb\_whgSave)

{

pruefeWohnung();

}

// Button Neue Wohnung Eingabe zurücksetzen

if (e.getSource() == jb\_whgClear)

{

resetWohnung();

}

### Eingabedialogfenster für neuen Mieter

## Anlage Oberflächenentwürfe

### Menüführung

* Allgemein
  + Logdatei lesen
  + Datenbank
    - Ausstattungsmerkmal Bad ergänzen
    - Ausstattungsmerkmal Küche ergänzen
    - Ausstattungsmerkmal Heizung ergänzen
    - Datenbank sichern
  + Beenden (ALT + F4)
* Immobilien
  + Neu anlegen (STRG + N)
  + Eigentümerwechsel
  + Kostenübersicht anzeigen (F2)
  + Mieterübersicht anzeigen (F3)
  + Übersicht Mietzahlungen
    - Aktuelles Monat (F6)
    - Laufendes Jahr (F7)
    - Spezifischer Zeitraum (F8)
  + Immobilie löschen
  + Immobilienübersicht gesamt (F4)
* Mieter
  + Neues Mietverhältnis
  + Mietverhältnis beendet
  + Änderungen
* Wohnung
  + Neue Wohnung
  + Übersicht (F5)
  + Änderung Ausstattungsmerkmal
* Kostenkonto
  + Kontobewegungen (STRG + K)
  + Betriebskostenabrechnung
* ?
  + Hilfe aufrufen (F1)
  + Programminformation

### Entwurf Startseite

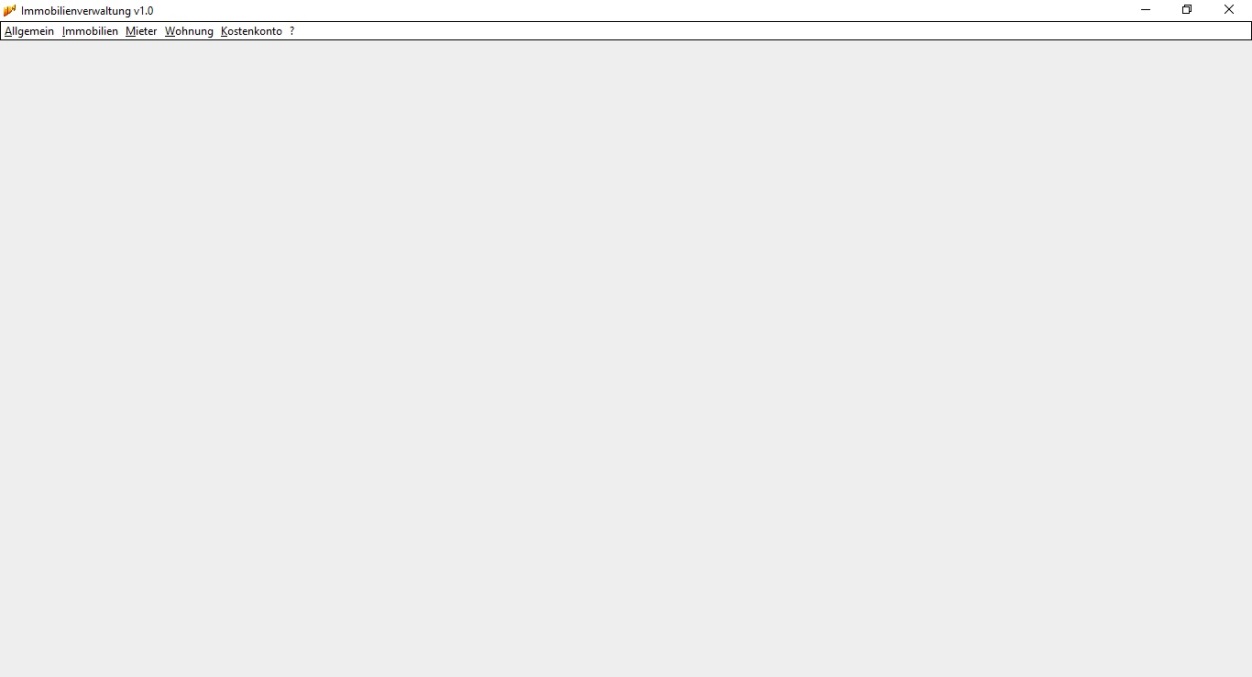


Abbildung : Entwurf Startseite

### Entwurf Hilfeseite

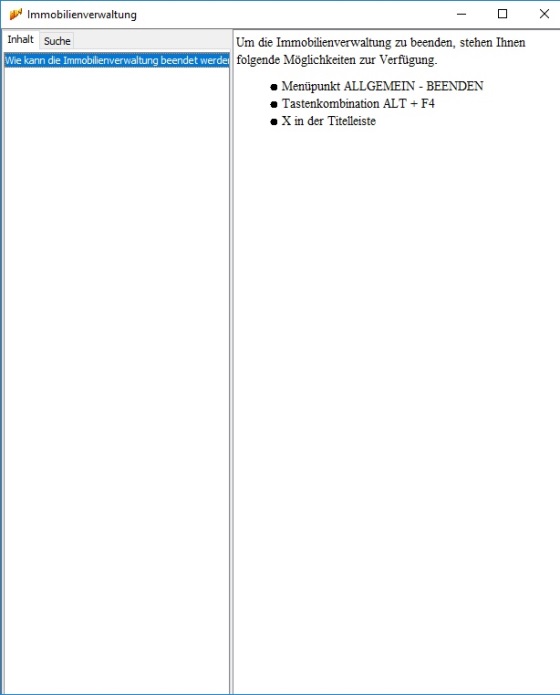


Abbildung : Entwurf Hilfeseite

### Entwurf Neuaufnahme Immobilien

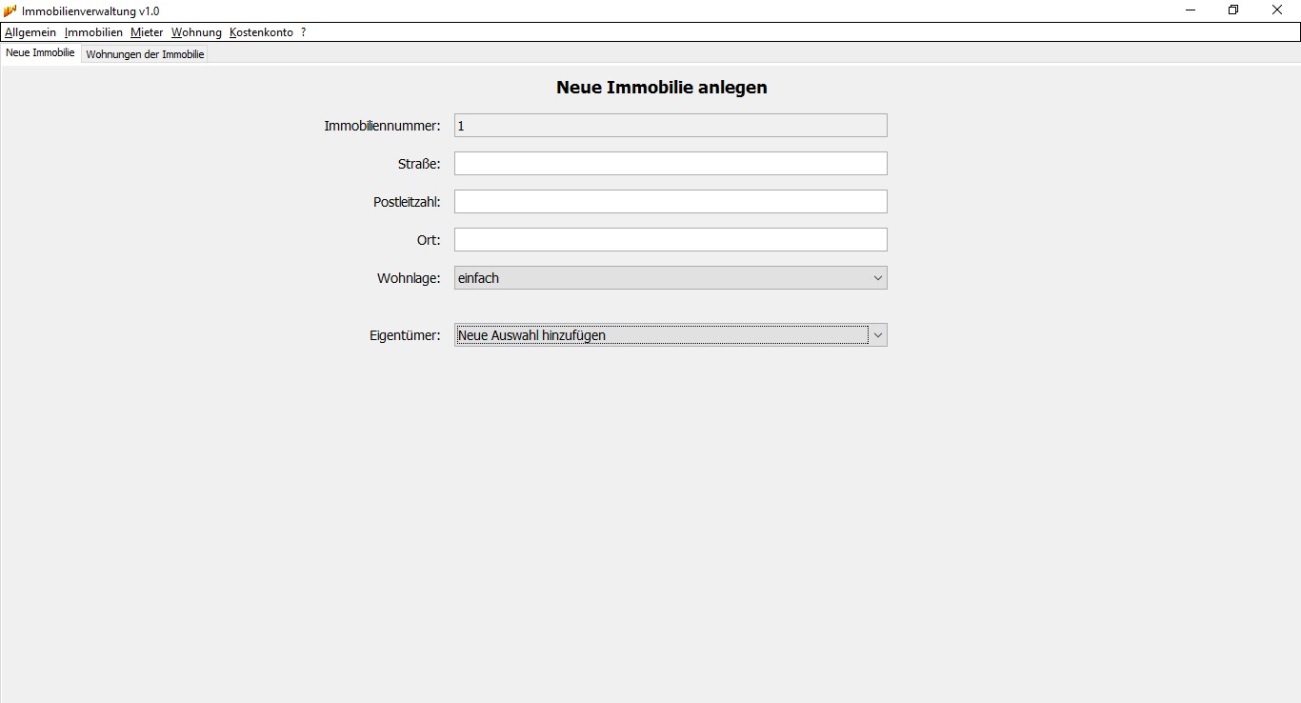


Abbildung : Benutzeroberfläche neue Immobilie

## Anlage Testberichte

### Funktionstest Tastaturkürzel

Der Funktionstest für die Tastaturkombination wurde nach Implementierung der Menüführung durchgeführt und ergab folgendes Ergebnis.

|  |  |  |
| --- | --- | --- |
| Tastaturkürzel | Erwartetes Ergebnis | Ergebnis erfüllt? |
| F1 | Aufruf Listener für F1 | ✓ |
| F2 | Aufruf Listener für F2 | ✓ |
| F3 | Aufruf Listener für F3 | ✓ |
| F4 | Aufruf Listener für F4 | ✓ |
| F5 | Aufruf Listener für F5 | ✓ |
| F6 | Aufruf Listener für F6 | ✓ |
| F7 | Aufruf Listener für F7 | ✓ |
| F8 | Aufruf Listener für F8 | ✓ |
| ALT + F4 | Aufruf Beenden-Dialog | ✓ |
| STRG + N | Aufruf Listener für STRG + N | ✓ |
| STRG + K | Aufruf Listener für STRG + K | ✓ |

### Funktionstest Menüpunkte

Der Funktionstest für die Menüpunkte wurde nach Implementierung der Menüführung durchgeführt und ergab folgendes Ergebnis:

|  |  |  |
| --- | --- | --- |
| Hauptmenüpunkt Allgemein | Testen der Listener-Aufrufe für jeden Menüpunkt | Ergebnis/Bemerkung |
| Hauptmenüpunkt Immobilien | Testen der Listener-Aufrufe für jeden Menüpunkt | ✓ |
| Hauptmenüpunkt Mieter | Testen der Listener-Aufrufe für jeden Menüpunkt | ✓ |
| Hauptmenüpunkt Wohnung | Testen der Listener-Aufrufe für jeden Menüpunkt | ✓ |
| Hauptmenüpunkt Kostenkonto | Testen der Listener-Aufrufe für jeden Menüpunkt | ✓ |
| Hauptmenüpunkt Hilfe | Testen der Listener-Aufrufe für jeden Menüpunkt | ✓ |

## Anlage Auszug aus dem Benutzerhandbuch

Um eine neue Immobilie anzulegen, wählen Sie den Menüpunkt IMMOBILIEN – NEUE IMMOBILIEN. Alternativ können Sie auch die Tastaturkombination STRG + N verwenden.

Es öffnet sich nun die Eingabemaske für neue Immobilien.

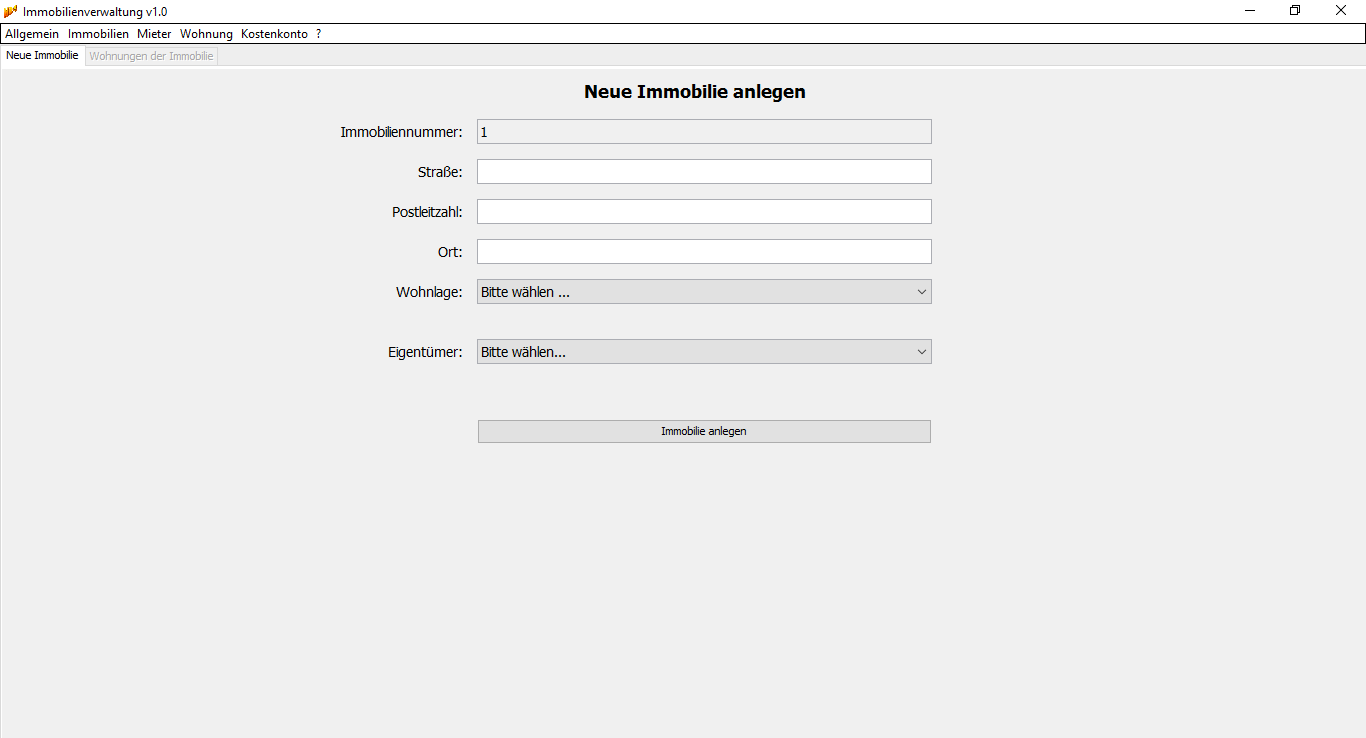


Abbildung : Benutzerhandbuch Neue Immobilie anlegen

Es ist nicht erforderlich, dass Sie den Eigentümer im Vorfeld angelegt haben, sondern Sie können den Eigentümer direkt in diesem Eingabeformular neu anlegen lassen. Öffnen Sie hierfür die Auswahl Eigentümer und wählen Sie den Eintrag ***Neue Auswahl hinzufügen*** aus.

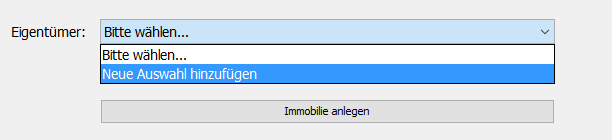


Abbildung : Benutzerhandbuch Neuen Auswahl hinzufügen

Es öffnet sich nun das Dialogfenster, in welchem Sie den neuen Eigentümer anlegen können. Die Eigentümernummer wird automatisch vergeben. In folgenden Feldern ist eine Auswahl, bzw. Eingabe notwendig, damit ein neuer Eigentümer angelegt wird.

* Anrede
* Nachname
* Vorname
* Strasse
* Postleitzahl
* Ort
* Telefon

Alle weiteren Angaben sind optional. Bitte beachten Sie bei der Eingabe des Geburtsdatum auf das richtige Eingabeformat. Dieses lautet: TT.MM.JJJJ. Sollten Sie eine Eingabe vergessen oder ein falsches Format benutzen, so werden Sie von der Anwendung darauf hingewiesen.

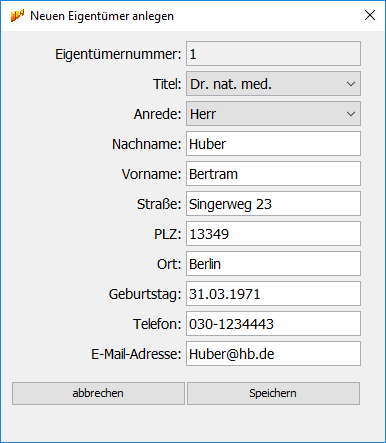


Abbildung : Benutzerhandbuch Neuen Eigentümer anlegen

Mit der Schaltfläche abbrechen beenden Sie den Vorgang der Eingabe ohne Speicherung. Mit der Schaltfläche Speichern können Sie den neuen Eigentümer in die Datenbank speichern.

Das Dialogfenster schließt sich nach dem erfolgreichen Speichern und Sie können nun im Auswahlfeld den neuen Eigentümer auswählen.

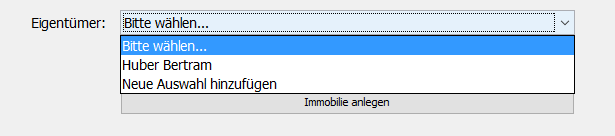


Abbildung : Benutzerhandbuch Neu angelegter Eigentümer

Wenn Sie alle erforderlichen Eingaben getroffen haben, können Sie die Immobilie neu anlegen. Da eine Immobilie auch über Wohnungen verfügt, erhalten Sie nun automatisch die Möglichkeit, diese Wohnungen anzulegen. Wenn Sie diese noch nicht zu diesem Zeitpunkt durchführen möchten, können Sie dies auch zu einem späteren Zeitpunkt durchführen.

1. (bew) = Bewertung [↑](#footnote-ref-1)
2. (gew) = Gewichtung [↑](#footnote-ref-2)