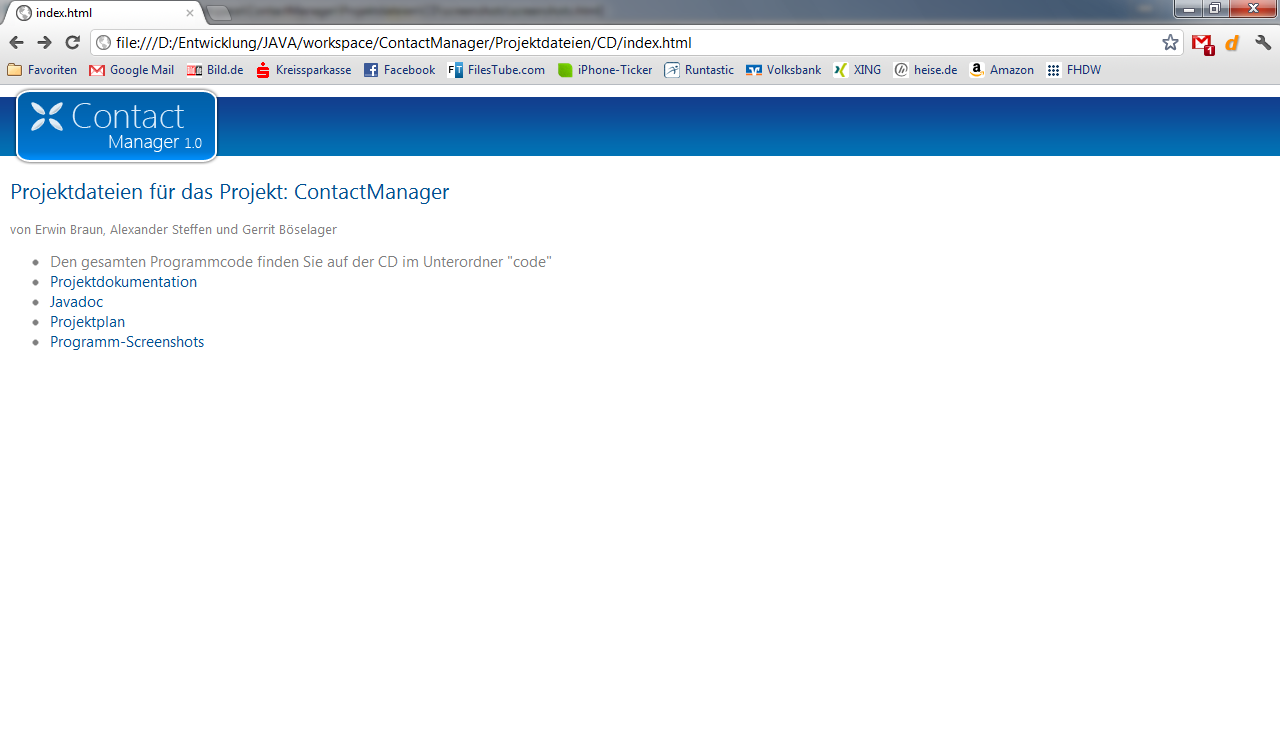
Dokumentation

-Projekt-  
*ContactManager*

****

-Projektmitglieder-  
*Erwin Braun, Alexander Steffen und Gerrit Böselager*

-Datum-  
*23. März 2012*

Inhaltsverzeichnis

[1 Anforderungsdefinition 3](#_Toc320251105)

[1.1 Muss-Funktionalitäten 3](#_Toc320251106)

[1.2 Optionale Funktionalitäten 4](#_Toc320251107)

[1.3 Abgrenzungskriterien 4](#_Toc320251108)

[2 Planung und Fachkonzeption 5](#_Toc320251109)

[2.1 Benutzerrechte 5](#_Toc320251110)

[2.2 Use-Case-Diagramm 6](#_Toc320251111)

[2.3 Klassendiagramm 7](#_Toc320251112)

[2.4 Entity-Relationship-Model (ERM) 8](#_Toc320251113)

[2.5 Prototypen 9](#_Toc320251114)

[2.6 Testplan 14](#_Toc320251115)

[3 Implementierung 15](#_Toc320251116)

[3.1 Team 15](#_Toc320251117)

[3.2 Programmierstandards 15](#_Toc320251118)

[3.3 Umgebung 16](#_Toc320251119)

[3.3.1 Webserver 16](#_Toc320251120)

[3.3.2 Datenbank Management System (DBMS) 16](#_Toc320251121)

[3.3.3 Programmiersprachen 16](#_Toc320251122)

[3.3.4 Entwicklungsumgebung 16](#_Toc320251123)

[3.3.5 Versionsverwaltung 16](#_Toc320251124)

[3.4 Architektur 17](#_Toc320251125)

[3.4.1 Abweichung zum Fachkonzept 17](#_Toc320251126)

[3.4.2 Drei-Schichten-Modell 17](#_Toc320251127)

[3.4.3 Technisches Klassendiagramm 18](#_Toc320251128)

[3.4.4 Package-Diagramm 19](#_Toc320251129)

[3.5 Externe Komponenten, Module und Quellen 20](#_Toc320251130)

[4 Testphase 22](#_Toc320251131)

[5 Projektabschluss & Fazit 23](#_Toc320251132)

[5.1 Erreichung der Anforderungen 23](#_Toc320251133)

[5.2 Soll / Ist - Vergleich 24](#_Toc320251134)

[5.3 Fazit 25](#_Toc320251135)

# Anforderungsdefinition

Als Ziel gilt die Umsetzung einer Kontaktverwaltung auf Basis einer Web-Applikation, bei der die gesamte Oberfläche in einem herkömmlichen Browser (z.B. Microsoft Internet Explorer, Mozilla Firefox oder Google Chrome) dargestellt werden kann. Die Pflege von Kontakten (Anlage, Bearbeitung, Löschung) wird von Benutzern durchgeführt.

Der ContactManager ist beschränkt auf den Einsatz innerhalb eines Unternehmens. Die von Administratoren gepflegten Benutzer haben die Möglichkeit private Kontakte, sowie auch Firmenkontakte zu pflegen. Zusätzlich können Kontakte als „öffentlich“ gekennzeichnet werden, wodurch der Kontakt auch von allen anderen Kollegen einsehbar, veränderbar und löschbar ist.

Die Funktionalitäten werden im Folgenden kurz konkret beschrieben.

## Muss-Funktionalitäten

Die Muss-Funktionalitäten dokumentieren die Funktionen, die explizit vom Auftraggeber verlangt werden und unbedingt umzusetzen sind.

* Benutzerlogin
  + Benutzer können sich mit Loginnamen und Passwort anmelden und können somit Funktionen, die nur für angemeldete Benutzer zugänglich sind, nutzen.
* Erfassung, Änderung und Löschung der Daten (Name, Adresse, Telefon, Freitext)
  + Nach dem Login kann der Nutzer Kontakte entsprechend verwalten.
* Suchen nach Kontakten (Volltextsuche)
  + Kontakte können von Benutzern per Volltextsuche gefunden werden.
* Plausibilitätsprüfung bei der Eingabe
  + Die angegebenen Kontaktdaten werden auf Plausibilität untersucht. Dazu gehören insbesondere E-Mail-Adresse und Geburtstag.
* Speicherung der Daten in einer Datenbank
  + Die Daten werden in einer MySQL-Datenbank gespeichert.

## Optionale Funktionalitäten

Zusätzlich zu den Muss-Funktionalitäten können die folgend gelisteten Features optional umgesetzt werden.

* Filter für die Suche
  + Filtermöglichkeit auf einzelne Spalten (z.B. Vorname, Nachname, Ort) eines Kontaktes
* Administratorfunktionen
  + Benutzerverwaltung (Anlegen, Bearbeiten, Löschen und Sperren von Benutzern)
* „Eigene Daten ändern“ für eingeloggte Benutzer
  + Benutzer können eigenen Daten, wie z.B. Vorname, Nachname, Passwort oder E-Mail eigenständig ändern
* „Kennwort vergessen“-Funktion
  + Benutzer können darüber beim Administrator ein neues Kennwort anfordern
* Druck von Einzelkontakten
  + Kontakte können zur Ausgabe auf Papier ausgedruckt werden.
* Export der Kontaktdaten in eine CSV-Datei
* Möglichkeit zur Erstellung von Kontaktgruppen
  + Selbst erstellte Kontakte können in Gruppen eingeteilt werden, die durch den Benutzer selbst zu verwalten sind.
* Auswertungsfunktionen
  + Welche und wie viele Benutzer sind aktuell online?
  + Welche Kontakte wurden zuletzt geändert?
  + etc.
* Anzeige der Adresse per GoogleMaps

## Abgrenzungskriterien

Folgende Abgrenzungskriterien wurden für das Projekt definiert.

* Schnittstellen zu externen Systemen (beispielsweise zu ERP oder CRM-Systemen) sind nicht vorgesehen.
* Ein Import von Kontaktdaten wird nicht unterstützt.
* Einmal gelöschte Kontakte können nicht wiederhergestellt werden, da diese direkt von der Datenbank entfernt werden.
* Die Datenübertragung zwischen Server und Client geschieht auf unverschlüsseltem Wege.
* Passwörter werden unverschlüsselt in der Datenbank abgespeichert.
* Mehrsprachigkeit ist nicht vorgesehen.

# Planung und Fachkonzeption

## Benutzerrechte

Im der folgenden Tabelle wird definiert, welche Funktionen von den entsprechenden Akteuren (angemeldeter Benutzer, nicht angemeldeter Benutzer und angemeldeter Administrator) genutzt werden dürfen.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | öffentliche Kontakte einsehen | öffentliche Kontakte bearbeiten und löschen | eigene private Kontakte einsehen | eigene private Kontakte bearbeiten und löschen | Neuen Benutzer anlegen, bearbeiten, löschen, sperren |
| Nicht angemeldeter Benutzer | **X** |  |  |  |  |
| Angemeldeter Benutzer | **X** | **X** | **X** | **X** |  |
| angemeldeter Administrator | **X** | **X** | **X** | **X** | **X** |

## Use-Case-Diagramm

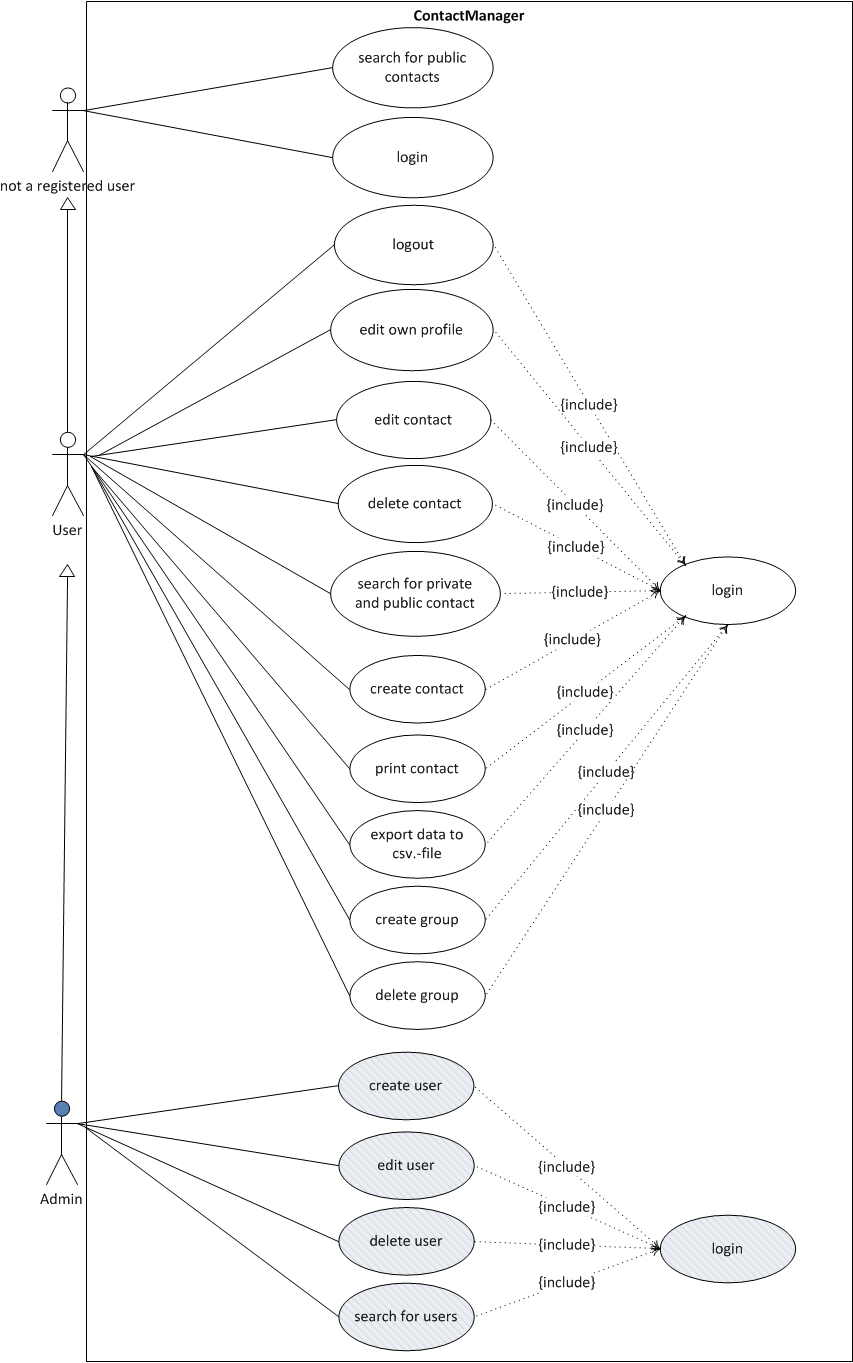


Abbildung : Use-Case-Diagramm

Die schraffierten Use-Cases und Akteure sind aus den optionalen Anforderungen entstanden und werden aus dem Grund auch nur dann umgesetzt, wenn entsprechende Kapazitäten im Projekt frei sind.

## Klassendiagramm

Folgend ist das fachliche Klassendiagramm aufgeführt, das die fachlichen Zusammenhänge/Beziehungen zwischen den einzelnen Klassen darstellt. Das technische Klassendiagramm (siehe 3.4.3) zeigt wie dieses Konzept technisch umgesetzt wurde und entsprechende Differenzen.

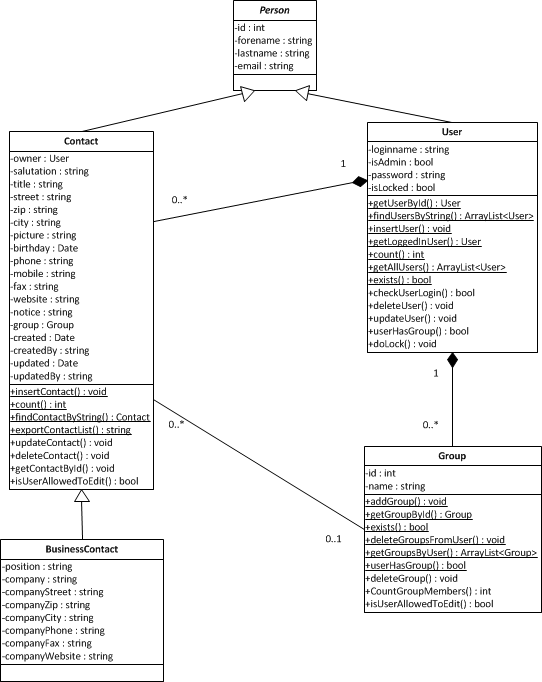


Abbildung : fachliches Klassendiagramm

## Entity-Relationship-Model (ERM)

Das folgende Entity-Relationship-Model zeigt die geplante Struktur der Datenbank.

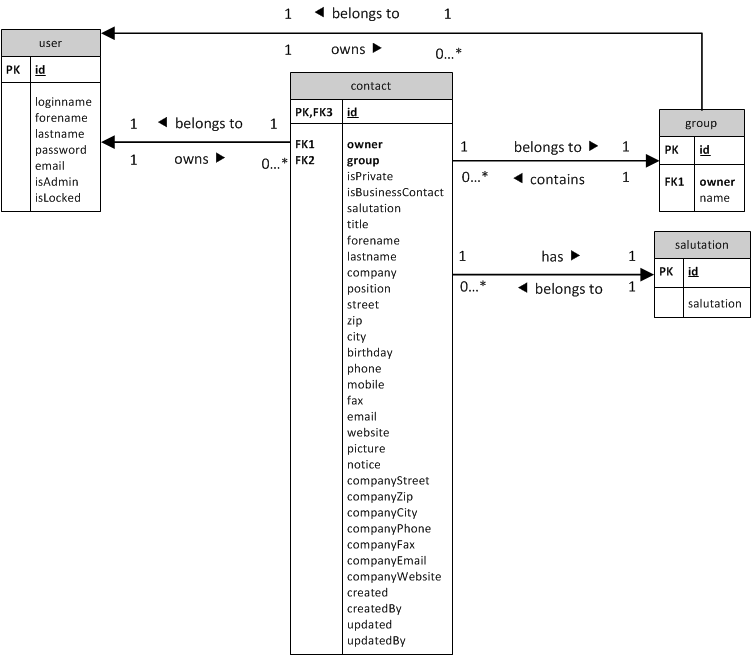


Abbildung : Entity-Relationship-Model

## Prototypen

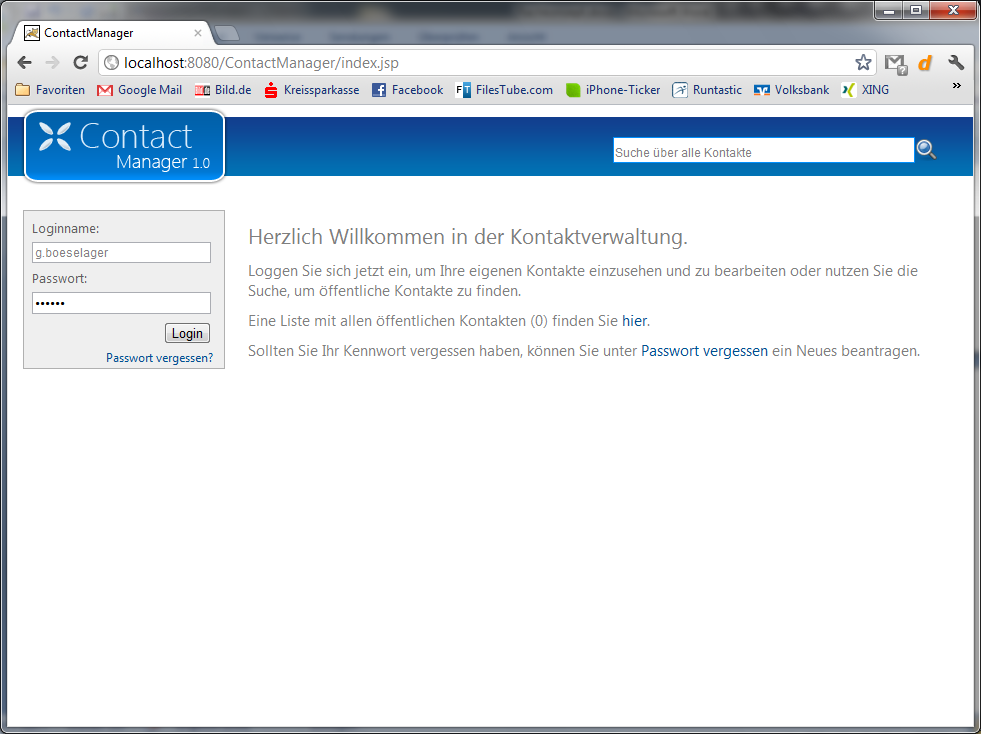
******Grundaufbau**

Abbildung : Prototyp des Grundaufbaus

Suche über alle öffentlichen Kontakte

Loginbereich

Inhaltsbereich

**Kontaktliste für eingeloggte Nutzer**

Such- und Filtermöglichkeit auf die aktuelle Liste

Informationen über eingeloggten User

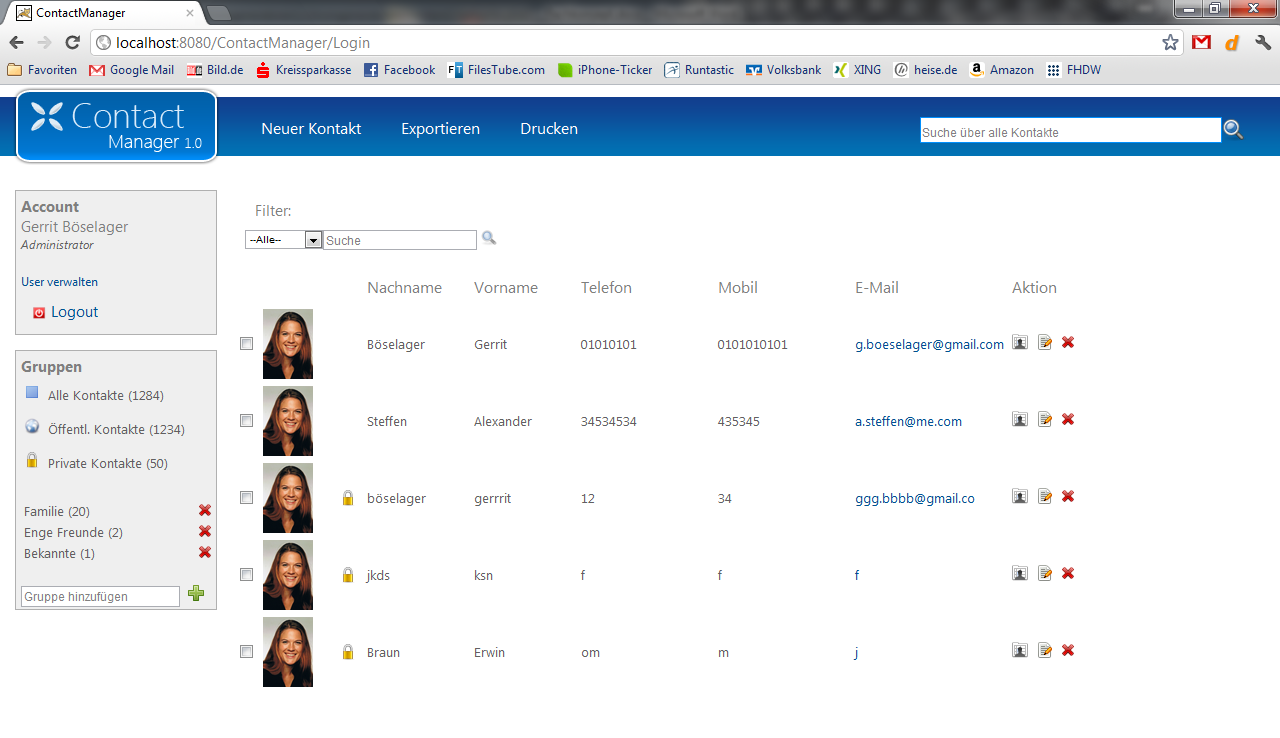


Abbildung : Prototyp der Kontaktliste

Filtern:  
- zeige „alle Kontakte“  
- zeige nur „öffentl. Kontakte“  
- zeige nur „private Kontakte“

Für jeden Kontakt die Möglichkeiten:   
Anzeigen, Bearbeiten und Löschen

Gruppen anzeigen, anlegen & löschen

**Kontakt anzeigen**

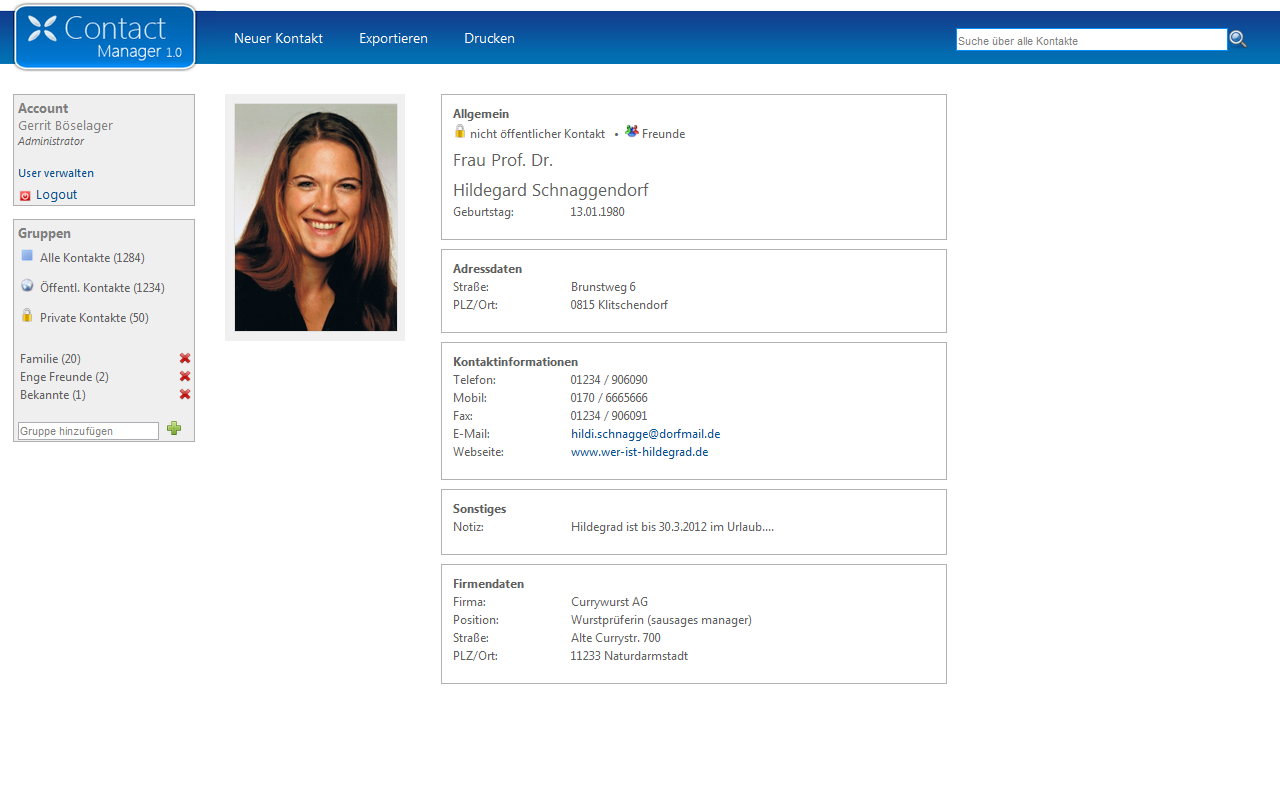
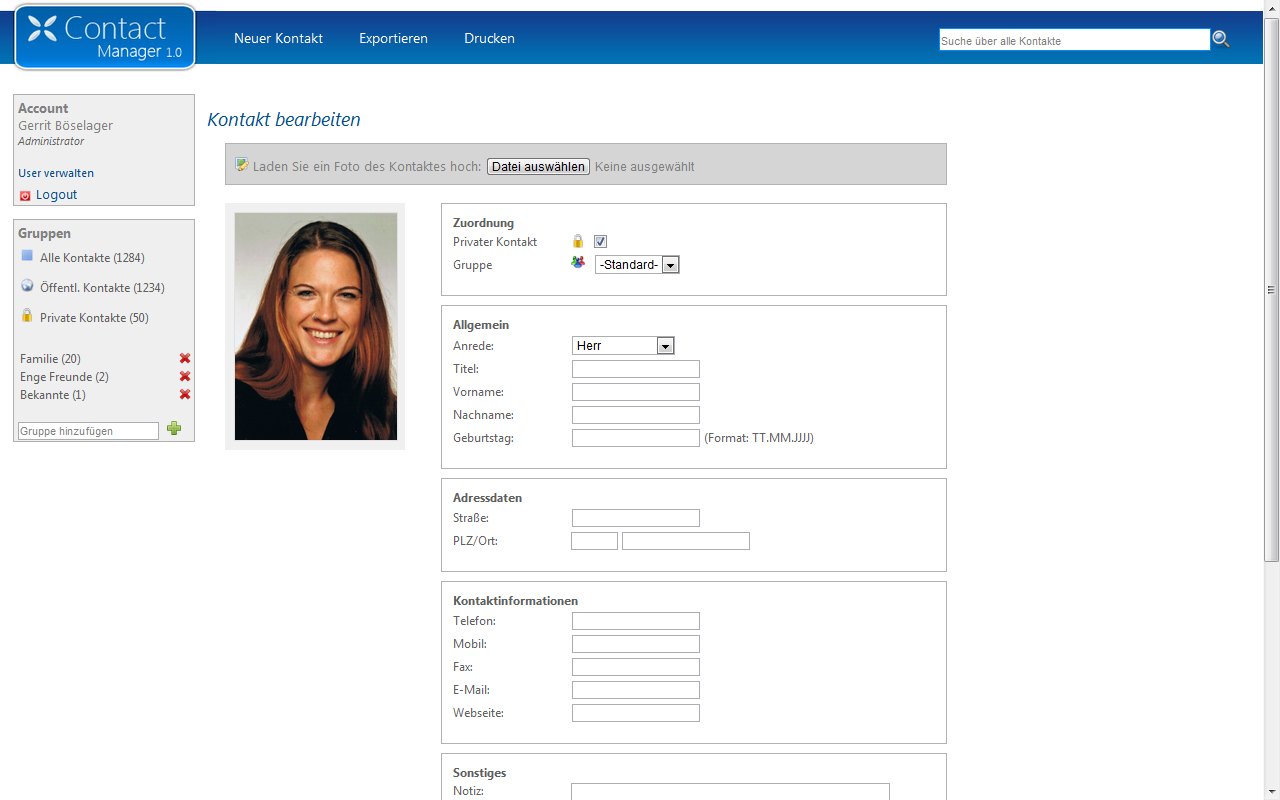


Abbildung : Prototyp der Kontaktansicht

******Kontakt bearbeiten/anlegen**

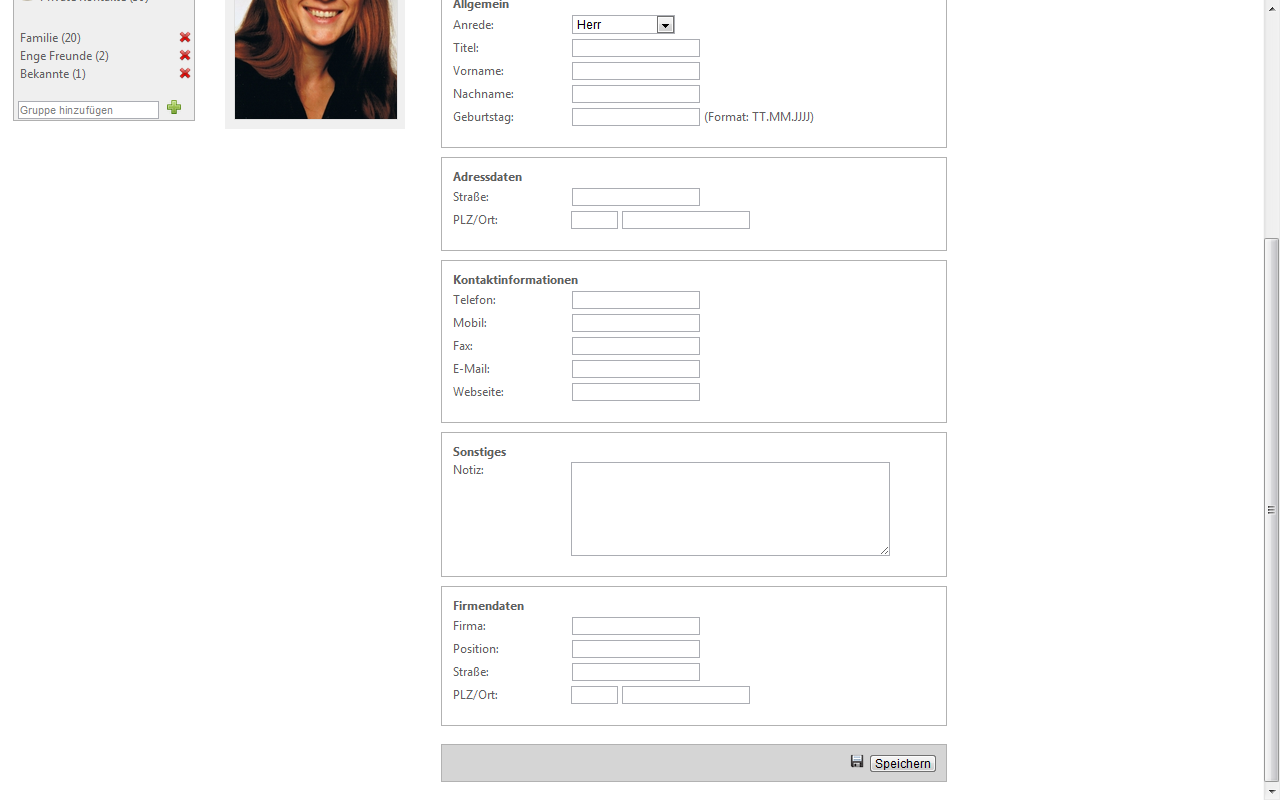


Abbildung : Prototyp der Kontakt bearbeiten Ansicht

**Kontakt drucken**

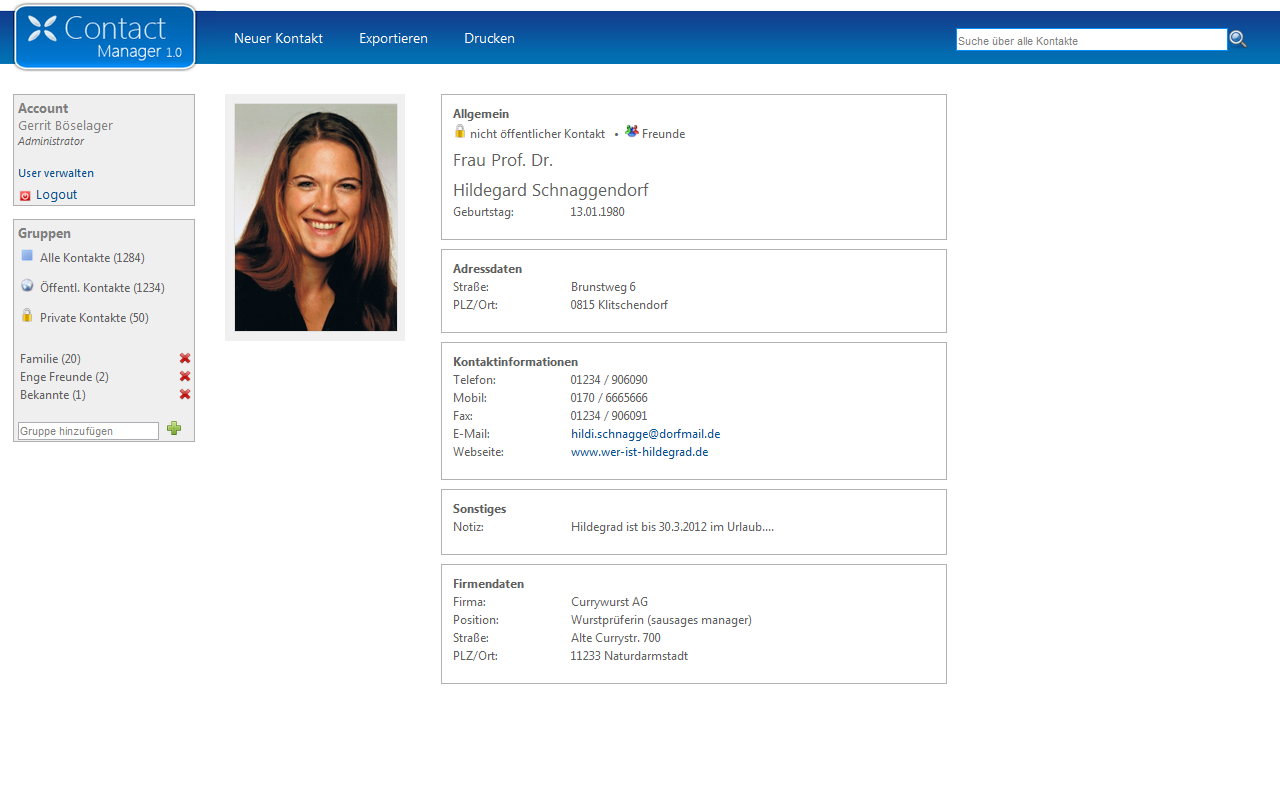


Abbildung : Prototyp der Druckansicht

## Testplan

In der folgenden Abbildung sind die geplanten Testfälle aufgelistet. Die detaillierte Testphase, inklusive entsprechendem Protokoll, kann aus Kapitel 4 entnommen werden.

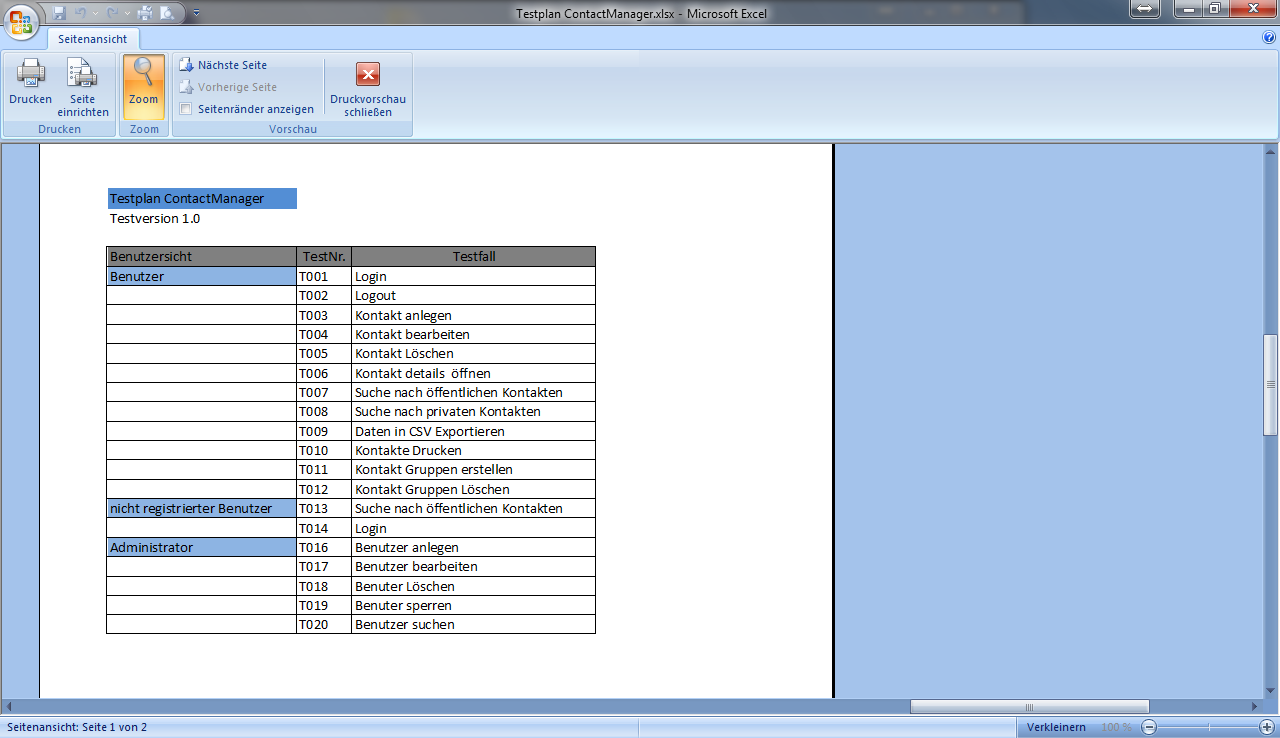


Abbildung :Gepante Testfälle

# Implementierung

## Team

Die Entwicklung wird in einem Drei-Mann-Team von den Entwicklern

* Braun, Erwin
* Steffen, Alexander
* Böselager, Gerrit

durchgeführt.

## Programmierstandards

Bei der *Benennung von Variablen, Eigenschaften, Methoden, Objekten, Klassen und sonstigen Konstruktionen*, ist darauf zu achten, dass *sinnvolle Bezeichnungen* gewählt werden, um so eine bessere Verständlichkeit des Programmcodes zu gewährleisten.

Die *Trennung von Wörtern* innerhalb eines Bezeichners wird durch *Großschreibung* des *ersten Buchstabens eines neuen Wortes* gekennzeichnet, wobei der *Anfangsbuchstabe des Bezeichners* immer *klein* zu schreiben ist. Als Sprache soll *Englisch*verwendet werden.

*Ausnahmen:*

Handelt es sich um einen *Klassenname*, so wird der Anfangsbuchstabe *großgeschrieben*.

Konstantenbezeichnungen werden in GROßBUCHSTABEN geschrieben und einzelne Wörter durch einen Unterstrich ( \_ ) voneinander getrennt.

Beispiele:

String myPerfectString;

public void getPerfectString() { … };

public class User { … }

public static final int CONSTANT\_VALUE = 815;

*Kommentare* sollen möglichst häufig an sinnvollen Stellen eingesetzt werden, wobei diese in Form von *„JavaDoc“* und in *englischer Sprache* anzufertigen sind.

Konventionen für Dateinamen

JSP-Dateien werden kleingeschrieben. Der Anfangsbuchstabe einer JAVA-Datei wird großgeschrieben und hat den gleichen Namen, wie die darin enthaltene Klasse.

## Umgebung

### Webserver

Als Webserver-Software kommt Apache Tomcat 7.0 (Release 7.0.23) zum Einsatz. Tomcat ist kostenlos und ein open source Projekt und ist dazu fähig *Java Servlets* sowie *JavaServerPages* zu verarbeiten. Zusätzlich wird dazu mindestens die JRE 6.0+ (Java Standard Edition Runtime Environment) benötigt.

Quelle: http://tomcat.apache.org

Download: http://tomcat.apache.org/download-70.cgi

### Datenbank Management System (DBMS)

Eingesetzt als DBMS (Datenbank Management System) wird MySQL (Version: 5.0.51b). MySQL ist eine der am häufigsten anzutreffenden Datenbank Web-Bereich und ist kostenfrei zu nutzen.

Quelle & Download: http://dev.mysql.com/downloads/mysql

### Programmiersprachen

Die hauptsächlich zu verwendende Programmiersprache im Bereich von Java Servlets und JavaServerPages ist *JAVA*. Zur Darstellung der Webseiten wird HTML (Hypertext Markup Language) und zum clientseitigen Scripting JavaScript (Achtung: JavaScript ≠ JAVA).

### Entwicklungsumgebung

Als Entwicklungsumgebung kommt „Eclipse Java EE for Web Developers“ in den Versionen „HELIOS“ und „INDIGO“ zum Einsatz.

### Versionsverwaltung

Um zu ermöglichen, dass die Teammitglieder zeitgleich am Projekt (bzw. auch an gleichen Dateien) arbeiten können und alle Änderungen an den Projektdateien protokolliert und versioniert werden, wird als Versionsverwaltung Git eingesetzt.

Das Projekt-Repository liegt dabei in der „Cloud“ beim Hoster www.github.com. Der Dienst ist kostenlos und alle Projektmitglieder haben ständig Zugriff auf die gemeinsamen Projektdateien.

Als Client-Tool auf den Windows-Rechnern kommt TortoiseGit (Version 1.7.6.0) zum Einsatz.

Quelle & Download: http://code.google.com/p/tortoisegit/downloads/list

## Architektur

### Abweichung zum Fachkonzept

Die im Entity-Relationship-Model gebildete Entität „salutation“ (Anrede) wird im Echtbetrieb nicht implementiert. Grund dafür ist, dass der Aufwand größer als der Nutzen ist, wenn die Anreden (Frau, Herr) in eine eigene Tabelle ausgelagert werden. Die Anreden sind daher fest als Text in die Tabelle „contact“ zu schreiben.

### Drei-Schichten-Modell

Die Architektur wird nach dem Drei-Schichten-Modell aufgebaut, das der Strukturierung der Software dient. Hierbei wird die Software in die Datenhaltungs-, Fachkonzept- und die Darstellungsschicht aufgeteilt.

In der Präsentationsschicht werden alle Oberflächenkomponenten, die der Interaktion mit dem Benutzer dienen, untergebracht. In der Fachkonzeptschicht befinden sich zentrale Logiken (bzw. Geschäftslogiken) die die Software steuern. Die Datenhaltungsschicht besteht, wie hier, meist aus einer Datenbank und ist für die Datenspeicherung und das Laden von Daten zuständig. [[1]](#footnote-1)

Folgend ist die in diesem Projekt gewählte Architektur aufgezeigt.

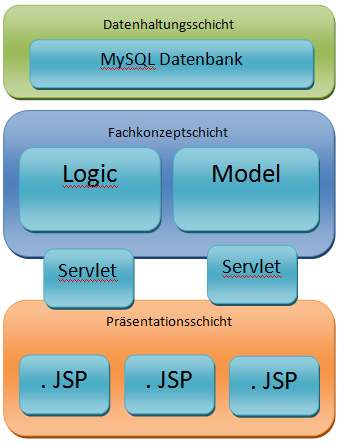


Abbildung : Drei-Schichten-Architektur

### Technisches Klassendiagramm

Das technische Klassendiagramm zeigt die umgesetzten Klassen und deren Beziehungen untereinander. Im Projekt werden statische Klassen als so genannte „Service-Klassen“ eingesetzt. (ContactManager, DataSource, UserManager, GroupManager, Validation, LayoutModules, Convert) Es ist nicht möglich von diesen Klassen Objekte/Instanzen zu bilden, wodurch diese keine direkten Beziehungen zu den fachlichen Klassen haben.

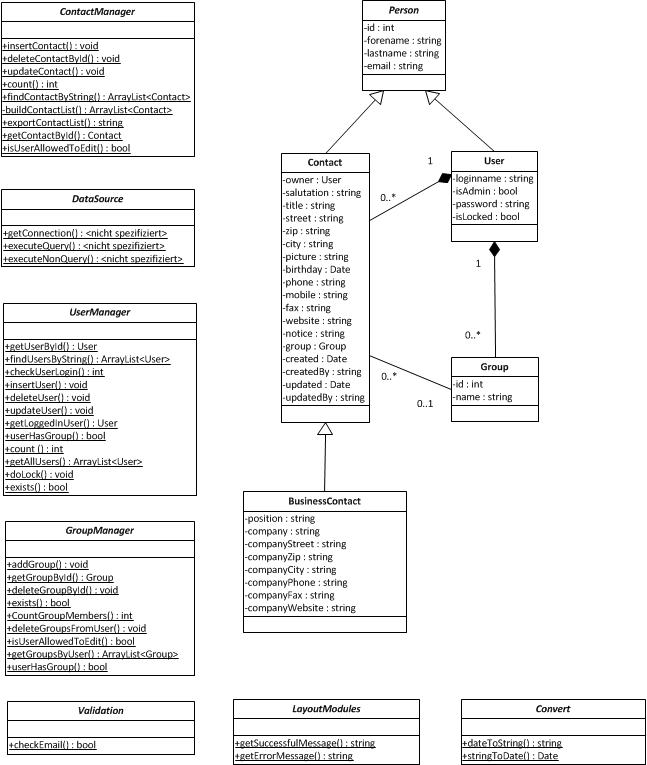


Abbildung : Technisches Klassendiagramm

### Package-Diagramm

Anhand des folgenden Package-Diagramms wird gezeigt, welche Packages importiert und verwendet werden.

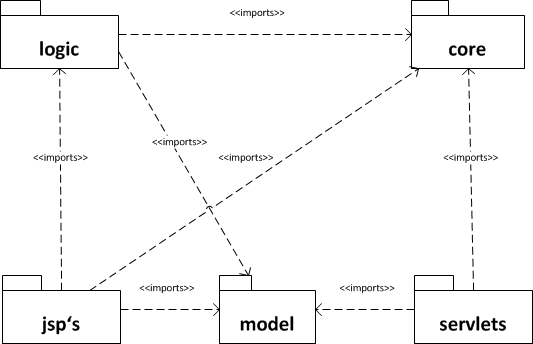


Abbildung : Package-Diagramm

## Externe Komponenten, Module und Quellen

Zur Erweiterung des Funktionsumfanges wurden während der Implementierungsphase verschiedene externe Komponenten und Module im Projekt eingesetzt, die folgend beschrieben werden.

**Apache Fileupload (Version 1.2.2)**

Diese Komponente wurde eingesetzt, um einen Upload von Kontaktfotos zu ermöglichen.

Quelle & Download: http://commons.apache.org/fileupload

Dokumentation: http://commons.apache.org/fileupload/apidocs/index.html

**Apache Commons IO (Version 2.1)**

Ist eine weitere Komponente von Apache, die zur Benutzung der Fileupload Library (siehe oben) benötigt wird.

Quelle & Download: http://commons.apache.org/io

Dokumentation: http://commons.apache.org/io/api-release/index.html

Um die oben beschriebenen Libraries nutzen zu können, müssen die entsprechenden .jar Dateien (commons-fileupload-1.2.2.jar bzw. commons-io-2.1.jar) in das lib-Verzeichnis des Webservers (hier: D:\Programme\Tomcat\lib) kopiert werden und ein Neustart des Tomcat durchgeführt werden.

**jQuery JavaScript Library (Version 1.7.1)**

Die jQuery ist eine kostenlose JavaScript Library, die das Entwickeln von JavaScript’s vereinfacht. Zusätzlich gibt es viele fertige JavaScript-Module, die man schnell auf eigenen Webseiten einbinden kann.

Quelle & Download: http://jquery.com

Dokumentation: http://docs.jquery.com/Main\_Page

In diesem Projekt wurde zusätzlich die **jQuery UI Library (Version** **1.8.17)** eingesetzt, um ein Kalenderelement zum Auswählen eines Geburtstagsdatums zu vereinfachen.

Quelle & Download: http://jqueryui.com

Dokumentation: http://jqueryui.com/demos

**Regular Expression für Überprüfung einer E-Mail Adresse**

Zur Validierung einer E-Mail-Adresse wurde folgender „Regulärer Ausdruck“ verwendet.

Quelle: http://www.mkyong.com/regular-expressions/how-to-validate-email-address-with-regular-expression/

**Grafiken**

Alle im Projekt verwendeten Grafiken wurden aus der „Open Icon Library“ entnommen. Die Verwendung ist kostenlos, wie ein Ausschnitt aus den LICENSES zeigt:

„ All these are icons are free and open source. That means they are free to use, modify, and redistribute, without any cost. They can be used for commercial and non-commercial projects or products, royalty free. The only requirement is that they remain free and open source, and you have to give credit where credit is due. So if you use the icons in your project, you have to offer the icons or anyderivatives of the icons with the same free/open license.”

Quelle: http://openiconlibrary.sourceforge.net/

**Session Counter**

Zum Zählen der angemeldeten Benutzer wird der SessionCounter von www.startdeveloper.com verwendet.

Quelle & Dokumentation:

http://www.stardeveloper.com/articles/display.html?article=2001112001&page=1

# Testphase

In der Testphase wurden alle im Testplan (siehe Kapitel 2.6) vorgesehenen Szenarien getestet und ggf. aufgetretene Fehler behoben.

Besonders beachtet wird, dass die Entwickler nicht ihre jeweils eigens entwickelten Programmkomponenten testen, sondern jeweils die eines anderen Entwicklers. Dadurch kann das Risiko verringert werden, dass nur das getestet wird, was auf jeden Fall funktioniert.

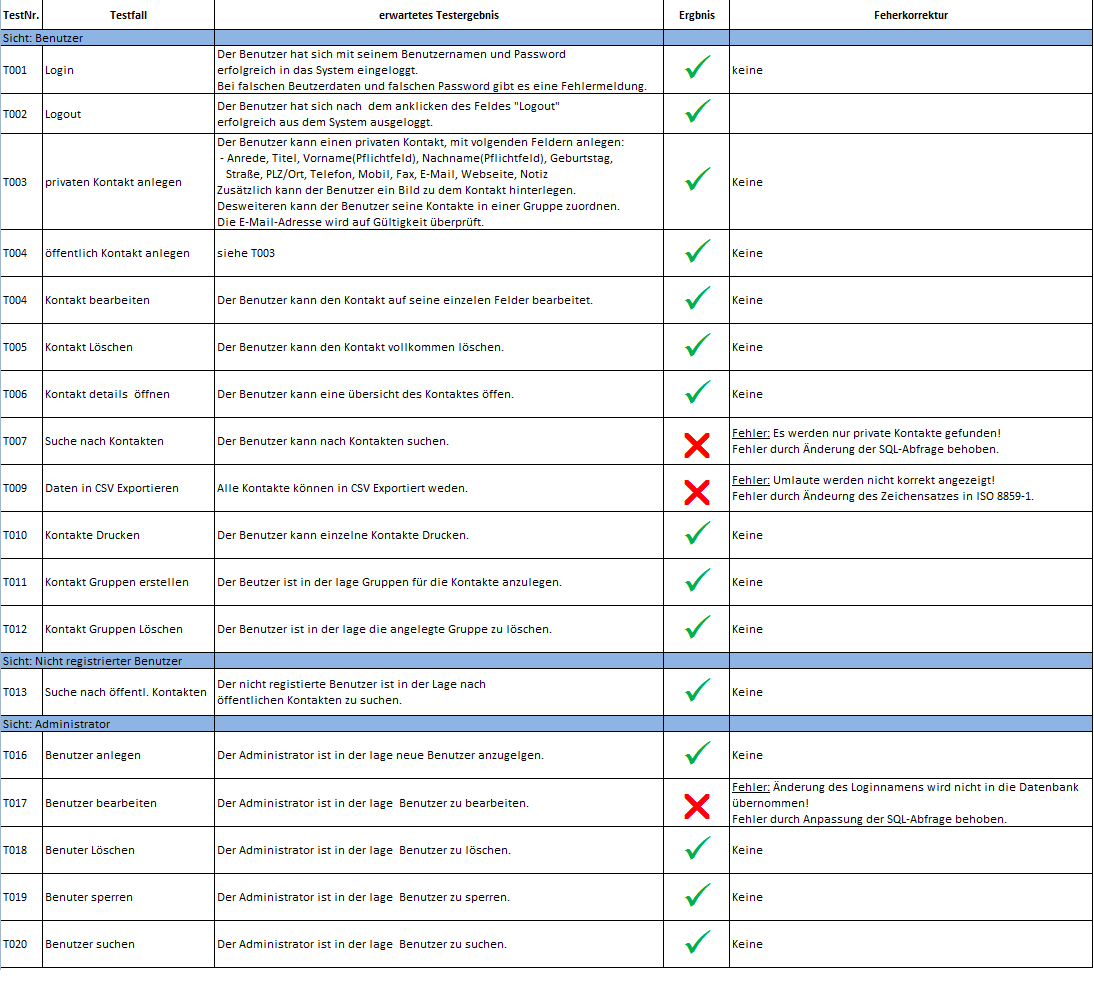


Abbildung : Testplan

# Projektabschluss & Fazit

## Erreichung der Anforderungen

Die in Kapitel 1.1 definierten Muss-Funktionalitäten der Software konnten vollständig implementiert werden und sind folgend nochmals aufgeführt:

* Benutzerlogin
* Erfassung, Änderung und Löschung der Daten (Name, Adresse, Telefon, Freitext, …)
* Suchen nach Kontakten (Volltextsuche)
* Kontakte können von Benutzern per Volltextsuche gefunden werden
* Benutzereingaben werden auf Plausibilität untersucht (insbesondere E-Mail-Adresse und Geburtstag)
* Speicherung der Daten in einer Datenbank

Die als optional eingestuften Funktionalitäten

* Filtermöglichkeit auf einzelne Spalten eines Kontaktes
* Benutzerverwaltung durch Administratoren (Anlegen, Bearbeiten, Löschen und Sperren von Benutzern)
* Druck von Einzelkontakten
* Export der Kontaktdaten in eine CSV-Datei
* Möglichkeit zur Erstellung von Kontaktgruppen für persönliche Kontakte
* Auswertungsfunktionen
  + Welche und wie viele Benutzer sind aktuell online?
  + Welche Kontakte wurden zuletzt geändert?

wurden ebenfalls erfolgreich in die Software integriert und sind als zusätzliche Funktionen nutzbar.

Um das Zeitbudget des Projektes nicht zu überschreiten, wurden die optionalen Funktionalitäten

* „Kennwort vergessen“-Funktion
* Anzeige der Adresse per GoogleMaps

nicht umgesetzt.

## Soll / Ist - Vergleich

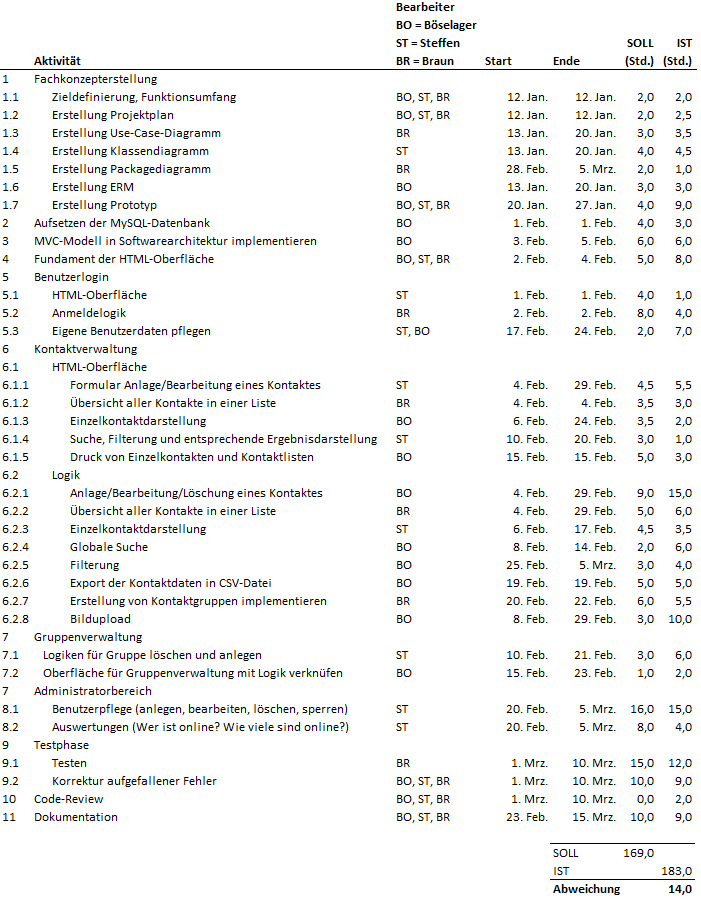
Der geplante Zeitaufwand für die 3 Projektmitglieder von insgesamt 169 Stunden (≈ 57 Stunden/pro Projektmitglied) konnte bis auf eine geringe Abweichung von 14 Stunden eingehalten werden, sodass bei Projektabschluss 183 Stunden Arbeitsaufwand verbucht wurden. Die zusätzlichen Stunden entstanden durch nicht eingeplante Zeiten für Code-Review und Schwierigkeiten im Bereich des Bilduploads.

Abbildung : Projektplan (inkl. SOLL/IST-Vergleich)

## Fazit

Es kann festgehalten werden, dass der Einsatz des Tools „Git“ zur Verwaltung und Versionierung des Quellcodes die Zusammenarbeit der Projektmitglieder während der Implementierungsphase positiv beeinflusst hat. Nur so war gewährleistet, dass keine Änderungen/Anpassungen überschrieben und damit verloren gehen und, dass die Arbeit von mehreren Projektmitgliedern an einer Datei möglich war.

Außerdem lässt sich festhalten, dass die intensive Planung des Projektes erheblich zum Projekterfolg beigetragen hat. Besonders die Einführung von Warnstufen (siehe Projektplan auf der beiliegenden CD) bei Nichteinhaltung von End-Terminen ließ die Projektmitglieder schnell erfassen, welche Arbeitspakete das zeitliche Projektziel akut gefährdeten.

In Anbetracht der Ergebnisse lässt sich abschließend sagen, dass das Projekt „ContactManager“ erfolgreich beendet wurde und die erlangten Kenntnisse in zukünftigen Projekten eingesetzt werden können.

1. vgl. http://www.itwissen.info/definition/lexikon/Drei-Schichten-Architektur-three-tier-architecture.html [↑](#footnote-ref-1)