Dokumentation

-Projekt-  
*RentACar*

****

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

-Datum-  
*23. September 2012*

Inhaltsverzeichnis

1 Anforderungsdefinition 2

1.1 Muss-Funktionalitäten 2

1.2 Optionale Funktionalitäten 3

1.3 Abgrenzungskriterien 3

2 Planung und Fachkonzeption 4

2.1 Use-Case Diagramm (Kundensicht) 4

2.2 Use-Case Diagramm (Clientsicht) 5

2.3 Entity-Relationship-Model (ERM) 6

2.4 Fachliches Klassendiagramm 7

2.5 Prototypen 8

2.6 Projektplan 10

3 Entwicklungsumgebung 11

3.1 Programmiersprachen 11

3.2 Entwicklungswerkzeuge 11

3.3 Versionsverwaltung 11

3.4 Tomcat Webserver (inkl. Axis2-Framework) 12

3.5 XAMPP Webserver und MySQL-Server 13

4 Umsetzung 14

4.1 Technisches Klassendiagramm 14

4.2 Externe Komponenten, Module und Quellen 14

5 Testphase 15

6 Projektabschluss & Fazit 16

6.1 Erreichung der Anforderungen 16

# Anforderungsdefinition

Als Ziel gilt die Umsetzung einer Software zu Verwaltung einer Autovermietung. In der Software müssen Standardprozesse, wie z.B. Suche nach Verfügbaren Fahrzeugen, Reservierung/Buchung eines Fahrzeugs und Preiskalkulation, abgebildet werden.

Aus technischer Sicht ist darauf zu achten, dass als Middleware Webservices zum Einsatz kommen sollen.

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.

* Als Client soll eine Webseite dienen, die über einen Browser aufgerufen werden kann.
* Webservices als Middleware
  + Um die Unabhängigkeit von Programmiersprache und Betriebssystem (Interoperabilität) zu gewährleisten, sollen als Middleware Webservices eingesetzt werden.
* Speicherung der Daten in einer Datenbank
  + Die Daten werden in einer MySQL-Datenbank gespeichert.
* Autosuche (Verfügbarkeit)
  + Potentielle Kunden können verfügbare Mietwagen zu einem bestimmten Zeitpunkt und Ort anzeigen lassen.
  + Ergebnis ist eine Liste verfügbarer Autos.
* Autodetailansicht
  + Dem Kunden wird ein Fahrzeugsteckbrief mit Detailinformationen gezeigt.
  + Beispielsweise: Anzahl Türen, PS, Farbe, Typ, Modell, Kraftstoff, usw.
* Autoreservierung (bzw. -buchung)
  + Nach Auswahl eines Fahrzeuges kann der Kunde eine Reservierung durchführen.
  + Die Angabe von persönlichen Kundendaten (z.B. E-Mail, Passwort, Vorname, Nachname, Anschrift) bei der ersten Reservierung ist notwendig.
  + Als Bestandskunde ist durch Angabe von E-Mail-Adresse und Passwort ein Login möglich, sodass die persönlichen Kundendaten nicht nochmals eingegeben werden müssen.
  + Auf Grundlage der Reservierungsinformationen muss eine Preiskalkulation durchgeführt.
* Plausibilitätsprüfung bei der Eingabe
  + Die angegebene E-Mail-Adresse des Kunden wird auf Plausibilität untersucht.

## Optionale Funktionalitäten

Zusätzlich zu den Muss-Funktionalitäten können die im Folgenden aufgelisteten Features optional umgesetzt werden.

* Filterfunktion
  + Der Kunde hat die Möglichkeit die Ergebnisliste der verfügbaren Fahrzeuge nach Automarke, Typ, Modell zu filtern.
* Autostandort anzeigen lassen
  + Der Standort des Fahrzeugs kann in einer Maps-Ansicht angezeigt werden.
* Login für Bestandskunden
  + Kunden, die bereits in Vergangenheit Fahrzeuge angemietet haben, können sich mit ihren Logindaten (E-Mail; Passwort) am System anmelden und haben so Einblick in aktuelle Reservierungsdetails.
* Fahrzeugbewertung
  + Eingeloggte Kunden können die von ihnen angemieteten Fahrzeuge bewerten.

## Abgrenzungskriterien

Folgende Abgrenzungskriterien wurden für das Projekt definiert.

* Ein Backend zur Pflege jeglicher Stammdaten (Fahrzeuge, Standorte und Kunden) ist nicht notwendig.
* Nach erfolgreicher Registrierung können Kunden ihre persönlichen Kundendaten nicht über die Webseite ändern.
* Schnittstellen zu externen Systemen (beispielsweise zu ERP oder CRM-Systemen) sind nicht vorgesehen.
* Die Datenübertragung zwischen Server und Client geschieht auf unverschlüsseltem Wege.
* Passwörter werden unverschlüsselt in der Datenbank abgespeichert.
* Mehrsprachigkeit ist nicht vorgesehen.

Pseudosicherheit mit aufnehmen, so dass wir dran gedacht haben, das allerdings abgrenzen. (login)

Wie sieht dier Client aus. Kommentare 2-3 zeilen top of document

Java webservice: javadoc

Technisches klassendiagramm

Testplan und testdokumentation (benutzertests)

USE-Case: anstatt person -> maschine

# Planung und Fachkonzeption

## Use-Case Diagramm (Kundensicht)

Abbildung 1 zeigt das Use-Case Diagramm aus Sicht des potentiellen Kunden.

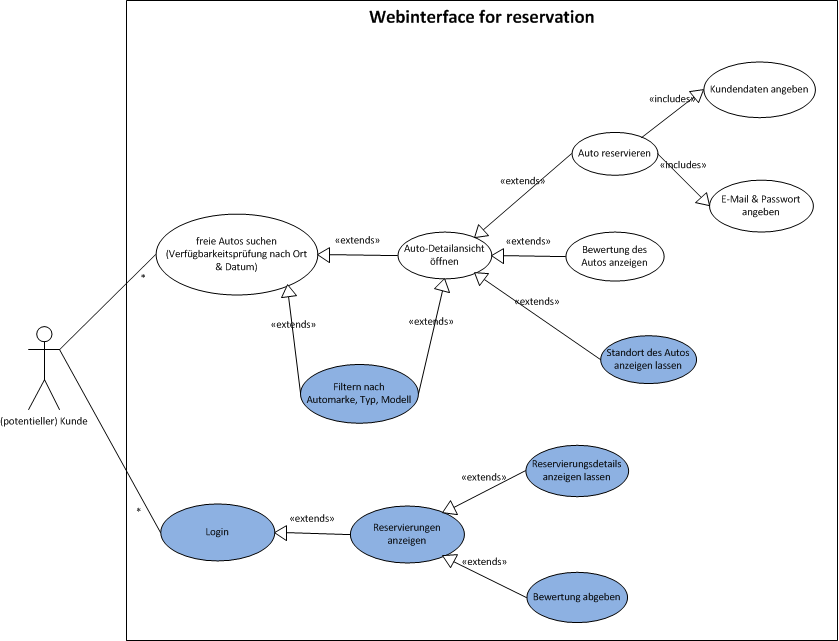


Abbildung 1: Use-Case Diagramm aus Kundensicht

## Use-Case Diagramm (Clientsicht)

Abbildung 2 zeigt das Use-Case Diagramm aus der Sicht des Clients mit Blick auf den Webservice.

## Disk1:Users:gboeselager:Develop:Github:RentACar:Projektdokumente:UseCase Client.png

Abbildung 2: Use-Case Diagramm aus Clientsicht

## Entity-Relationship-Model (ERM)

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

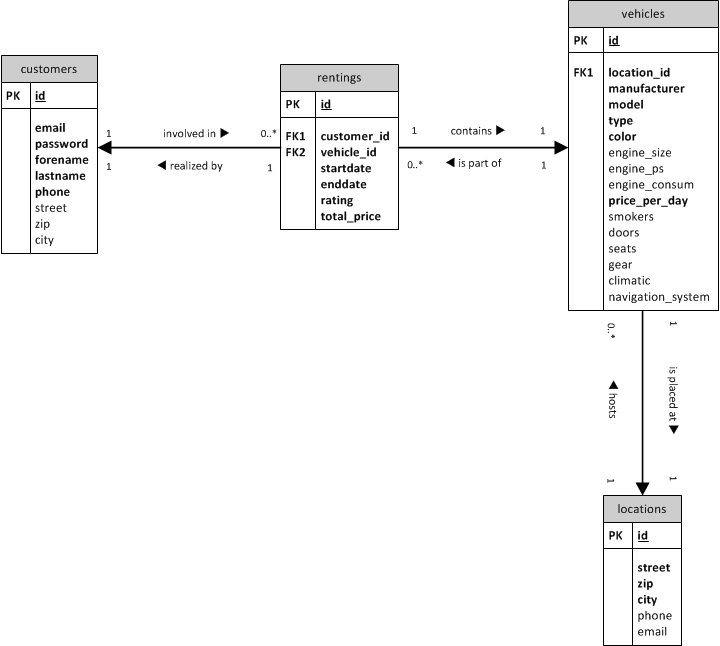


Abbildung 3: Entity-Relationship-Model

## Fachliches Klassendiagramm

Folgend ist das fachliche Klassendiagramm aufgeführt, das die fachlichen Zusammenhänge/Beziehungen zwischen den einzelnen Klassen darstellt. Das technische Klassendiagramm (siehe 4.2) zeigt wie dieses Konzept technisch umgesetzt wurde.

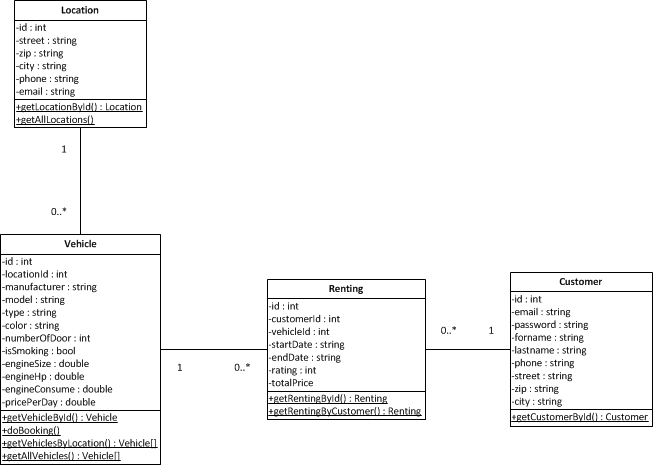


Abbildung 4: Fachliches Klassendiagramm

## Prototypen

**Startseite (Suchmaske)**



Abbildung 5: Prototyp Startseite

**Fahrzeugliste der (verfügbaren) Fahrzeuge**

Abbildung 6: Prototyp Fahrzeugliste

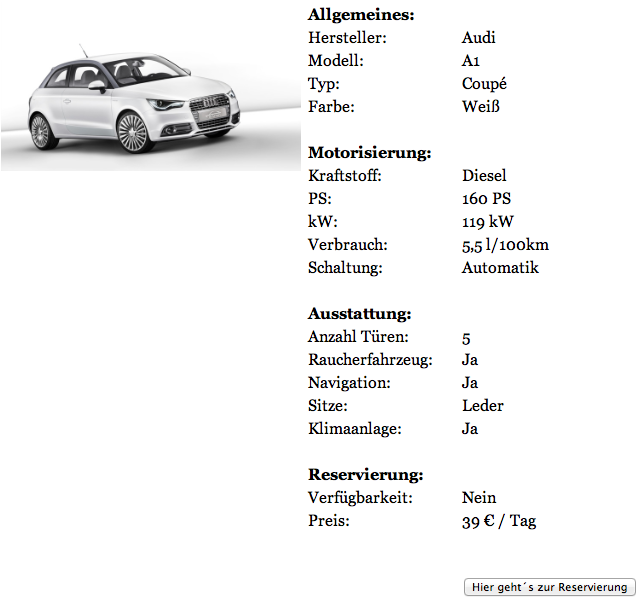
**Detailansicht für ein Fahrzeug**

Abbildung 7: Prototyp Detailansicht

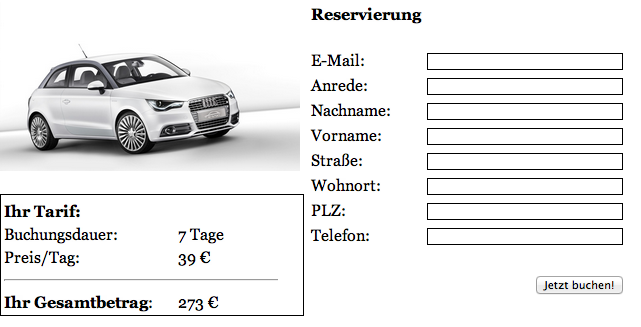
**Reservierung eines Fahrzeuges**

Abbildung 8: Prototyp Reservierung

## Projektplan

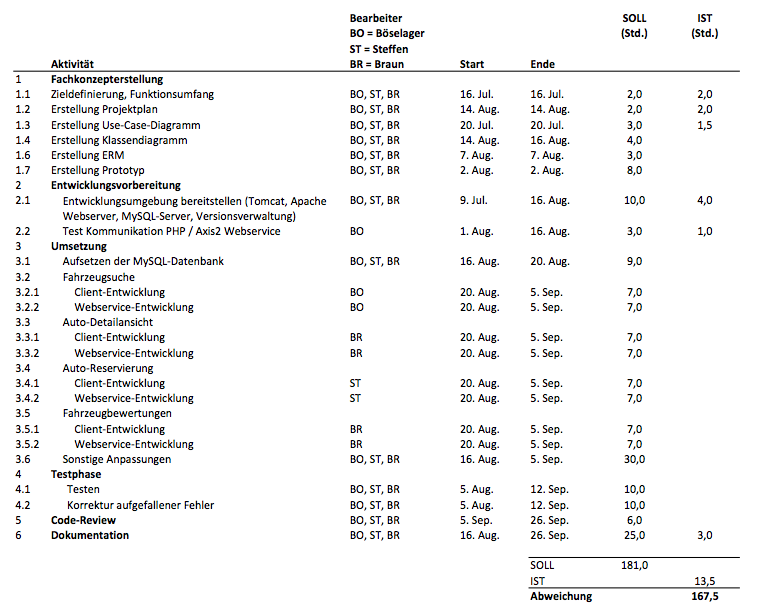


Abbildung 9: Projektplan

# Entwicklungsumgebung

Die Entwicklung findet auf Rechnern mit dem Betriebssystem Apple Mac OSX Mountain Lion (Version 10.8) statt. Alle Schritte dieser Dokumentation beziehen sich auf dieses Umfeld.

## Programmiersprachen

Abbildung 10 zeigt die im Projekt verwendeten Programmiersprachen.

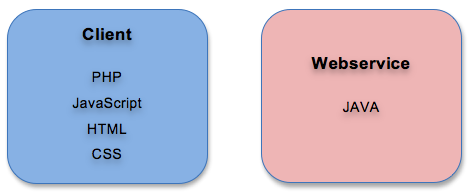


Abbildung 10: Programmiersprachen

## Entwicklungswerkzeuge

Als Entwicklungsumgebung für die JAVA-Webservices kommt „Eclipse Java EE for Web Developers“ in der Version „JUNO“ zum Einsatz.

Der Quellcode für die Client-Seite wird mit dem Tool „Komodo Edit 7.1.1“ entwickelt.

## 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.

Das Client-Tool zur Synchronisierung der Projektdateien heißt „GitHub for Mac“.

Quelle & Download: http://mac.github.com/

## Tomcat Webserver (inkl. Axis2-Framework)

Als Webserver-Software kommt Apache Tomcat inkl. Axis2-Framework für Webservices (Version 1.6.2) zum Einsatz. Zusätzlich wird dazu mindestens die JRE 6.0+ (Java Standard Edition Runtime Environment) benötigt.

Zur Installation reicht es aus die Axis2 „Binary Distribution“ herunterzuladen und an einem geeigneten Ort zu entpacken (Hier: /Applications/axis2-1.6.2). Diese Version enthält bereits einen Tomcat Webserver, sodass dieser nicht explizit installiert werden muss.

Quelle: http://axis.apache.org/axis2/java/core/download.cgi

Binary Distribution: http://apache.heikorichter.name//axis/axis2/java/core/1.6.2/axis2-1.6.2-bin.zip

WAR Distribution: http://apache.heikorichter.name//axis/axis2/java/core/1.6.2/axis2-1.6.2-war.zip

Zum Starten des Servers sind folgende Befehle im Mac OSX-Terminal auszuführen:

1. Setzen der Umgebungsvariable „JAVA\_HOME“ zum Java SDK. Bei Mac OSX ist dies standardmäßig unter /Library/Java/Home zu finden.

export JAVA\_HOME="/Library/Java/Home"

1. Start des Axis2-Servers.

sh /Applications/axis2-1.6.2/bin/axis2server.sh



Abbildung 11: Erfolgreicher Start des Webservers

Wurde der Server erfolgreich gestartet, kann die Startseite auf Port 8080 (http://localhost:8080) aufgerufen werden. Um festzustellen, welche Webservices derzeit verfügbar sind, wird unter http://localhost:8080/axis2 eine Informationsseite bereitgestellt.

## XAMPP Webserver und MySQL-Server

Als Client soll eine PHP-Webseite auf die bereitgestellten Axis2-Webservices zugreifen. Dazu wird auf den Entwicklungsrechnern ein Webserver benötigt, der PHP-Code interpretieren kann. Zur Speicherung der Daten soll eine MySQL-Datenbank (Datenbank Management System) zum Einsatz kommen.

Die Software „XAMPP for Mac OSX“ des Herstellers „Apache Friends“ enthält bereits alle oben aufgeführten Komponenten (Webserver, PHP und MySQL) in einer praktischen Gesamtlösung und wird in der Version 1.7.3 verwendet.

Quelle: http://www.apachefriends.org

Download: http://www.apachefriends.org/download.php?xampp-macosx-1.7.3.dmg

Nach der unkomplizierten Installation kann das Programm „XAMPP-ControlPanel“ zum Starten der Komponenten geöffnet werden.

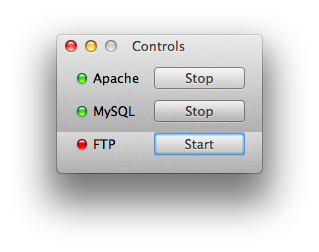


Abbildung 12: XAMPP ControlPanel

**Wichtige Pfade und URL’s:**

Home-Verzeichnis für PHP-Seiten: /Applications/XAMPP/xamppfiles/htdocs

Webseite: http://localhost

MySQL-Administration: http://localhost/phpmyadmin

# Umsetzung

## Technische Vorgaben / Sicherheit

**DateTime**

Zur Vereinfachung des Datenaustausches zwischen PHP-Client und Webservice werden Datums- und Zeitangaben nicht mit dem JAVA Datentyp „DateTime“ übermittelt, sondern als Strings, die immer dem folgenden Aufbau entsprechen müssen:

JJJJ-MM-TT HH:MM:SS

Beispiel: „2012-08-19 17:28:19“

**Sicherheit**

Da die Kommunikation mit dem Webservice unverschlüsselt geschieht, könnte der Datenverkehr von potentiellen „Angreifern“ mitgelesen werden. Aus dem Grund wird im Projekt darauf verzichtet weitere Sicherheitsmechanismen zu implementieren, die z.B. die Webservice-Aufrufe, für die ein Login auf Clientseite Voraussetzung ist, absichern. Diese Sicherheitsmechanismen würden nur eine Pseudo-Sicherheit suggerieren.

## Technisches Klassendiagramm

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

Die Klassen Location, Vehicle, Renting und Customer sind auf der Client-Seite identisch vorhanden, sodass auf ein weiteres Klassendiagramm verzichtet wird.

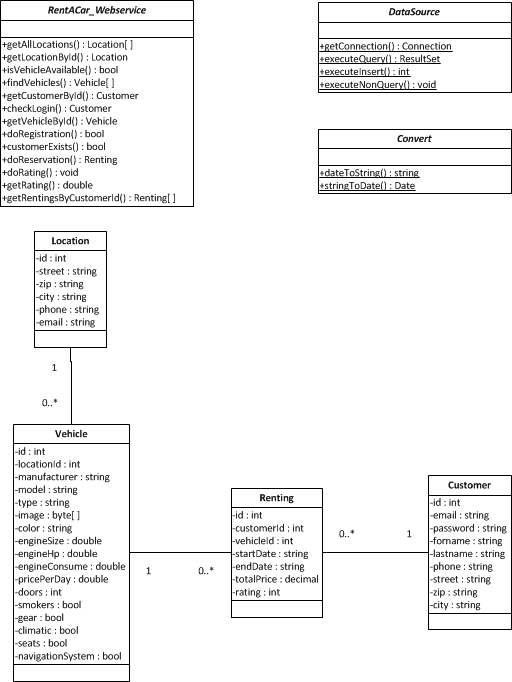


Abbildung 13: Technisches Klassendiagramm

## Externe Komponenten, Module und Quellen

**jQuery JavaScript Library (Version 1.7.2)**

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 1.8.21) eingesetzt, um ein Kalenderelement zum Auswählen eines Datums zu vereinfachen.

Quelle & Download: http://jqueryui.com

Dokumentation: http://jqueryui.com/demos

# Testphase

In der Testphase wurden alle im Testplan (siehe Kapitel ) 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.

# Projektabschluss & Fazit

## Erreichung der Anforderungen

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

* Webseite als Client
* Webservices als Middleware
* Speicherung der Daten in einer Datenbank
* Autosuche (Verfügbarkeit)
* Autodetailansicht
* Autoreservierung (bzw. -buchung)
* Plausibilitätsprüfung bei der Eingabe

Die als optional eingestufte Funktionalitäten

* Login für Bestandskunden
* Fahrzeugbewertungen

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

* Filtermöglichkeit
* Autostandort anzeigen lassen

nicht umgesetzt.

## Soll / Ist - Vergleich

Der geplante Zeitaufwand für die 3 Projektmitglieder von insgesamt 181 Stunden (≈ 60 Stunden/pro Projektmitglied) konnte bis auf eine geringe Abweichung von 6,5 Stunden eingehalten werden, sodass bei Projektabschluss 187,5 Stunden Arbeitsaufwand verbucht wurden.

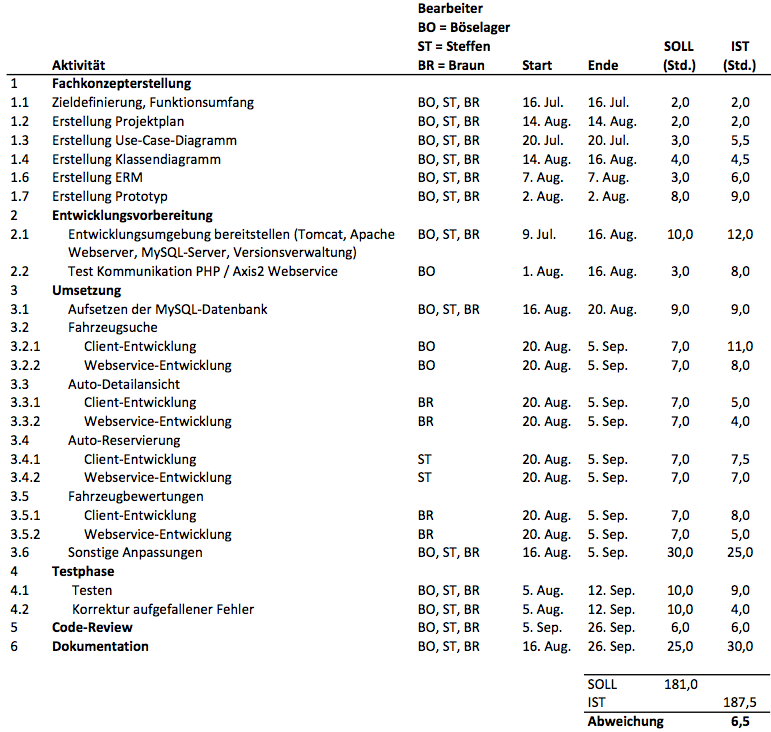


Abbildung 14: Projektplan (inkl. SOLL/IST-Vergleich)

## Fazit

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.