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 Testplan 10

2.7 Projektplan 11

3 Entwicklungsumgebung 12

3.1 Programmiersprachen 12

3.2 Entwicklungswerkzeuge 12

3.3 Versionsverwaltung 12

3.4 Tomcat Webserver (inkl. Axis2-Framework) 13

3.5 XAMPP Webserver und MySQL-Server 14

3.6 Einrichtung Produktivumgebung 15

4 Umsetzung 16

4.1 Technische Vorgaben / Sicherheit 16

4.2 Technisches Klassendiagramm 17

4.3 Externe Komponenten, Module und Quellen 18

5 Testphase 19

6 Projektabschluss & Fazit 20

6.1 Erreichung der Anforderungen 20

6.2 Soll / Ist - Vergleich 21

6.3 Fazit 22

# 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.

# Planung und Fachkonzeption

## Use-Case Diagramm (Kundensicht)

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

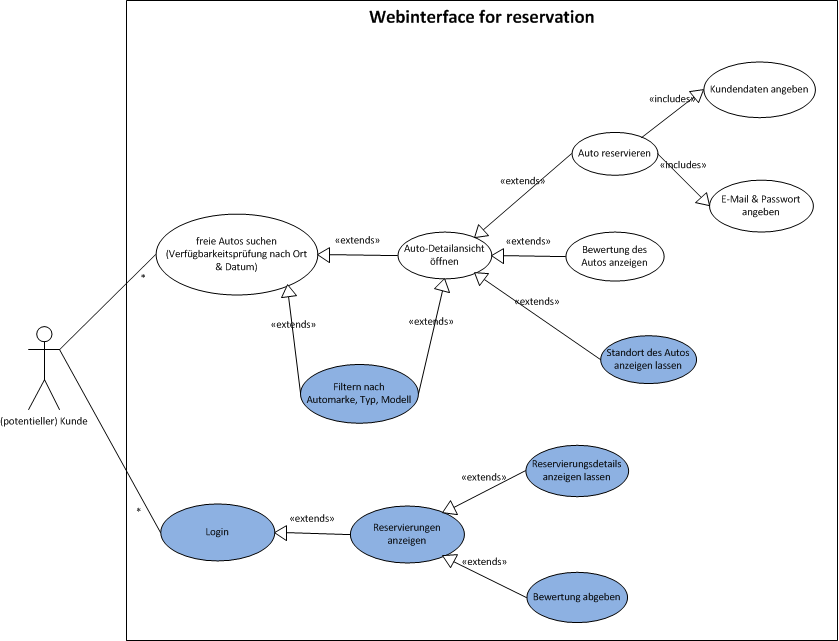


Abbildung 1: Use-Case Diagramm aus Kundensicht

## Use-Case Diagramm (Clientsicht)

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

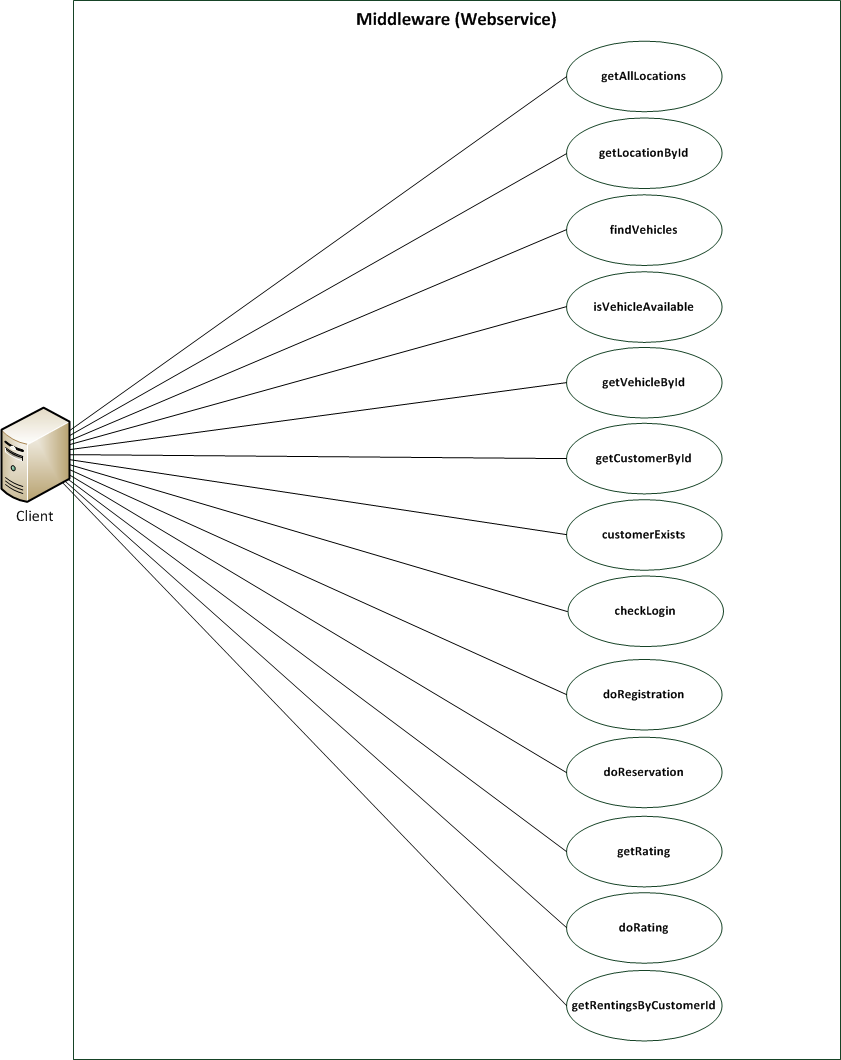


Abbildung 2: Use-Case Diagramm aus Clientsicht

## Entity-Relationship-Model (ERM)

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

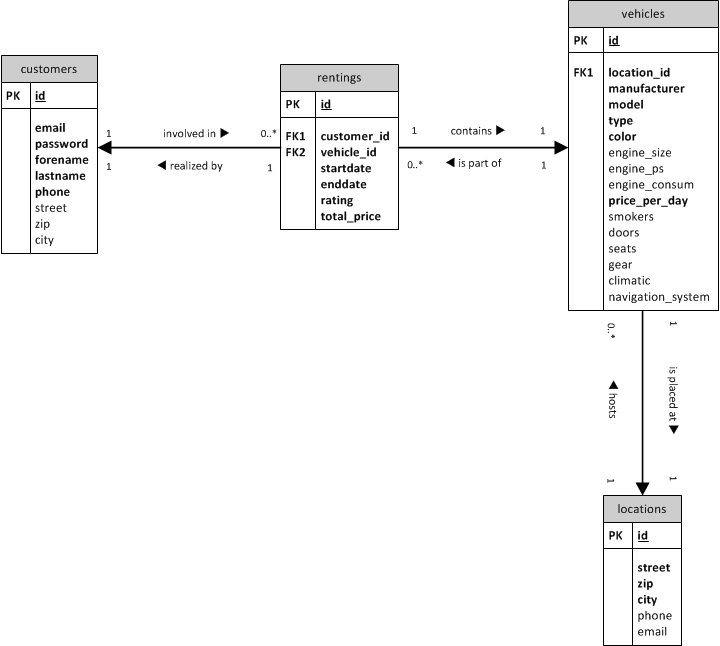


Abbildung 3: Entity-Relationship-Model

## 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.

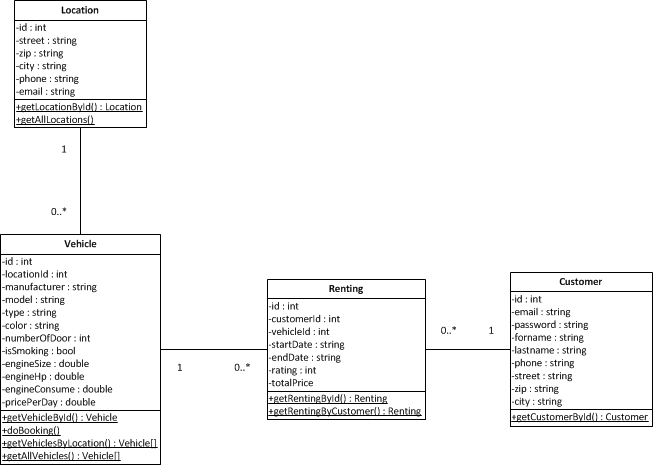


Abbildung 4: Fachliches Klassendiagramm

## Prototypen

**Startseite (Suchmaske)**



Abbildung 5: Prototyp Startseite

**Fahrzeugliste der (verfügbaren) Fahrzeuge**

Abbildung 6: Prototyp Fahrzeugliste

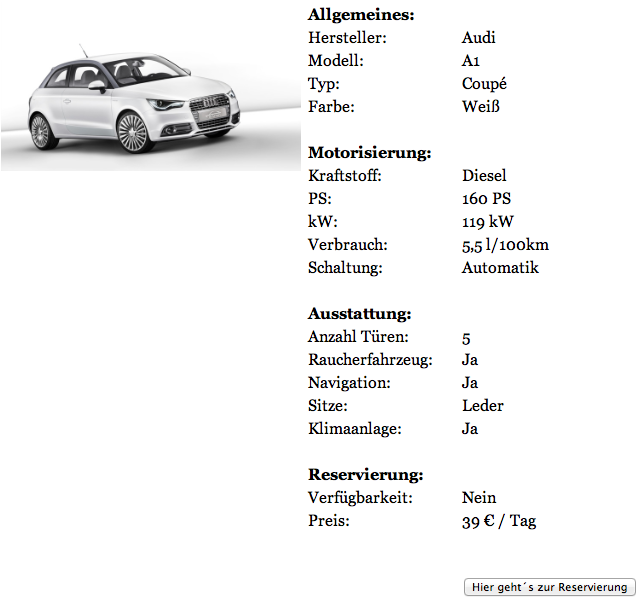
**Detailansicht für ein Fahrzeug**

Abbildung 7: Prototyp Detailansicht

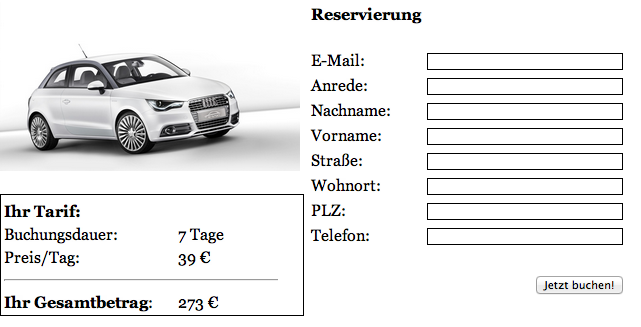
**Reservierung eines Fahrzeuges**

Abbildung 8: Prototyp Reservierung

## Testplan

In der folgenden Abbildung sind die geplanten Testfälle aufgelistet. Die detaillierte Testphase, inklusive entsprechendem Protokoll, kann aus Kapitel 5 entnommen werden.

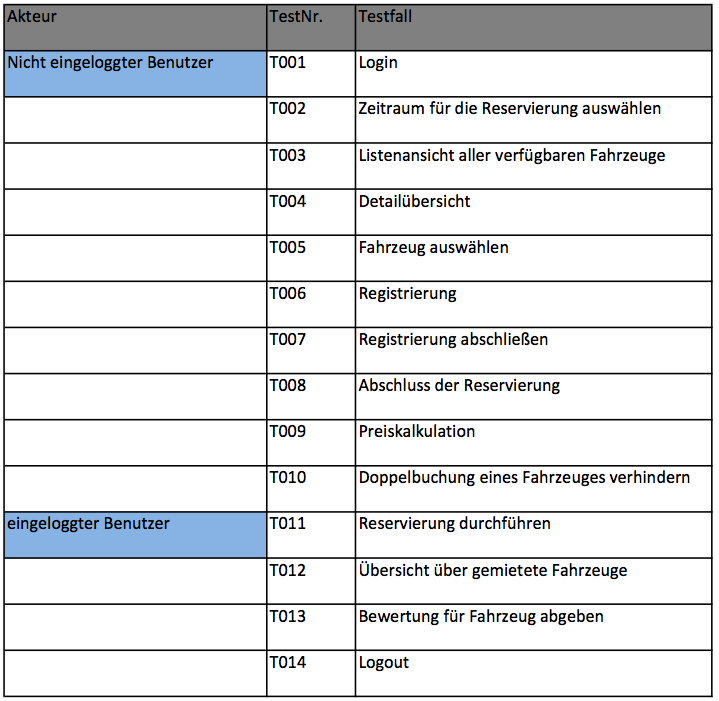


Abbildung 9: Testplan

## Projektplan

Dem Projektplan ist zu entnehmen, bis wann Arbeitspakete von welchem Projektmitglied erledigt werden müssen, damit das Projektziel nicht gefährdet wird. Das Ergebnis ist im abschließenden SOLL / IST - Abgleich (Kapitel 6.2) einzusehen.

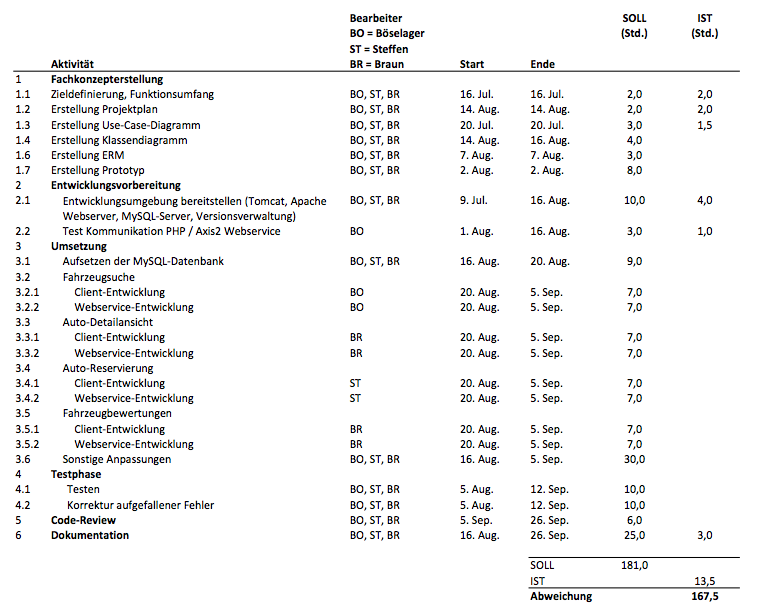


Abbildung 10: Projektplan

# 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.

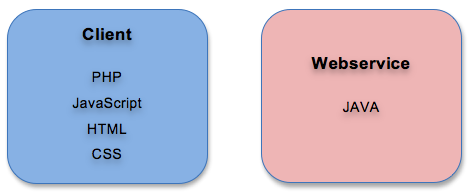


Abbildung 11: Programmiersprachen

## 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 12: 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.

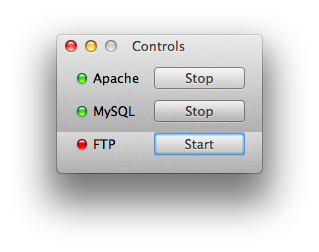


Abbildung 13: XAMPP ControlPanel

**Wichtige Pfade und URL’s:**

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

Webseite: http://localhost

MySQL-Administration: http://localhost/phpmyadmin

## Einrichtung Produktivumgebung

Als Produktivumgebung kommt ein dedizierter Server zum Einsatz, der als virtuelle Maschine im FHDW-Rechenzentrum gehostet wird.

Der Server wurde von Prof. Dr. Ulrich Reus in der Grundkonfiguration (lediglich das Betriebssystem Ubunto 10.4LTS ist installiert) bereitgestellt und ist über die IP-Adresse 193.22.73.246 erreichbar. Voraussetzung für den Zugriff ist eine bestehende VPN-Verbindung zum FHDW-Netzwerk.

|  |  |
| --- | --- |
| **Spezifikationen** | |
| IP-Adresse: | 193.22.73.246 |
| Betriebssystem: | Ubuntu 10.4LTS |
| CPU: | 1 |
| RAM: | 512 MB |
| Festplatte: | 25 GB |
| User / Passwort: | gb / ifw410fhdw |

Ziel ist es, dass der Server als Host für den Webservice und die Datenbank eingesetzt wird. Dazu ist die Installation der Komponenten „Apache2“, „MySQL“, „PhpMyAdmin“ und „Axis2 Tomcat Webserver“ notwendig

1. SSH-Verbindung zum Server herstellen

ssh gb@193.22.73.246

1. Apache2 Webserver installieren und anschließend starten

sudo apt-get install apache2

sudo /etc/init.d/apache2 start

1. PHP5 installieren

sudo apt-get install php5 libapache2-mod-php5

1. MySQL installieren

sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql

1. PhpMyAdmin installieren

apt-get install phpmyadmin

1. Axis2 Installation: Download wie in Kapitel 3.4 beschrieben und nach /var/tmp/axis2-1.6.2 entpacken. Zum Start des Axis2-Server sind dann folgende Befehle im Terminal einzugeben:

export JAVA\_HOME="/home/gb/jdk1.7.0/bin/java"

sh /var/tmp/axis2-1.6.2/bin/axis2server.sh

|  |  |
| --- | --- |
| **Wichtige Pfade** | |
| PhpMyAdmin (Datenadministration): | http://193.22.73.246/mysqladmin  User: root; Passwort: *leer* |
| Axis2 (Infoseite): | 193.22.73.246/axis2 |

# Implementierung

## Team

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

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

durchgeführt.

## Technische Vorgaben / Sicherheit

**Austausch von Datums- und Zeitangaben (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**

Wie in den Abgrenzungskriterien (siehe Kapitel 1.3) bereits erwähnt wird der Datenverkehr zwischen Client und Webservice nicht verschlüsselt. Da das Ziel dieses Projektes nicht die Absicherung gegen potentieller „Angreifer“ ist, wird auf die Implementierung weiterer Sicherheitsmechanismen verzichtet.

Jedoch sollte im Falle einer Produktivsetzung Beachtung finden, dass der Datenverkehr verschlüsselt und z.B. die Webservice-Aufrufe (für die ein Login auf Clientseite Voraussetzung ist) abgesichert werden.

Diese Sicherheitsmechanismen würden derzeit nur eine Pseudo-Sicherheit suggerieren, da die Datenübertragung unverschlüsselt geschieht.

## 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.

## 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.

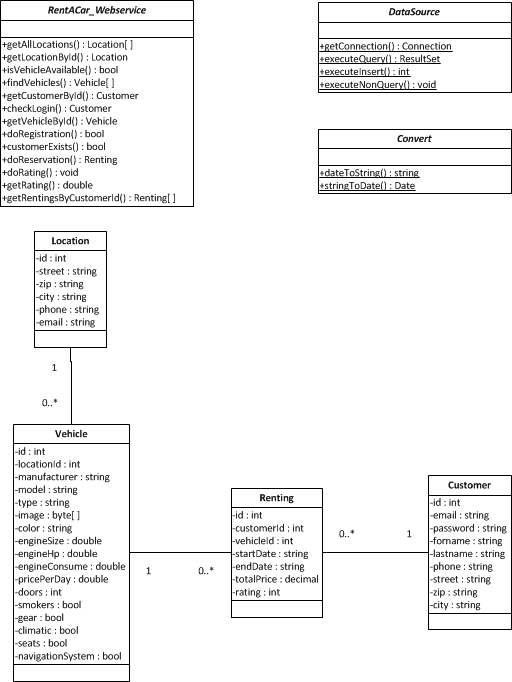


Abbildung 14: Technisches Klassendiagramm

## Externe Komponenten, Module und Quellen

**JAVA MySQL Connector (Version 5.1.21)**

Für das Herstellen einer Verbindung zur MySQL-Datenbank wird auf Webservice-Seite ein JDBC Datenbank Connector benötigt. In diesem Projekt wird dazu der JAVA MySQL-Connector 5.1.21 verwendet.

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

Dokumentation: http://dev.mysql.com/doc/refman/5.1/en/connector-j.html

**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 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.

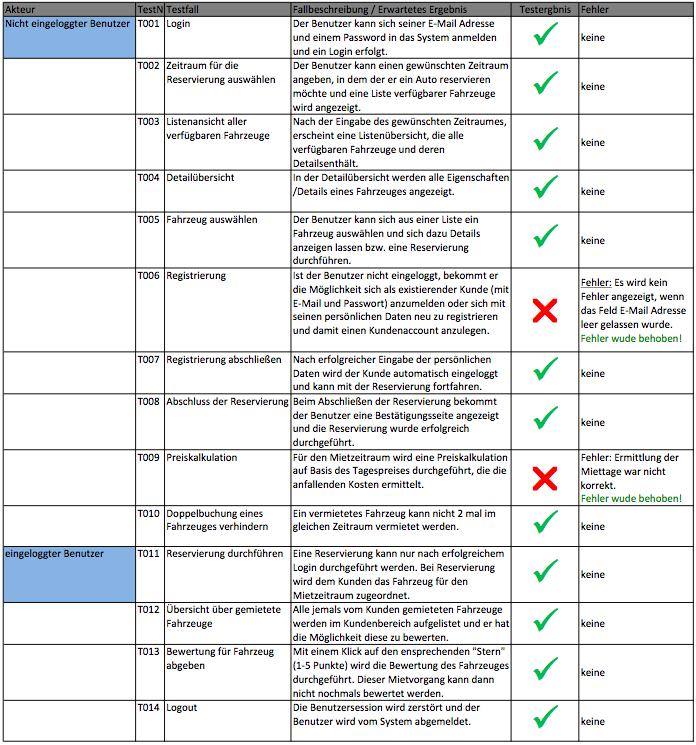


Abbildung 15: Testplan & Fehlerkorrektur

# 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. Nicht im Projektplan enthalten sind ca. 30 Stunden Entwicklungsaufwand für eine iPhone-App, die das Projektteam aufgrund freier Ressourcen zusätzlich durchgeführt hat.

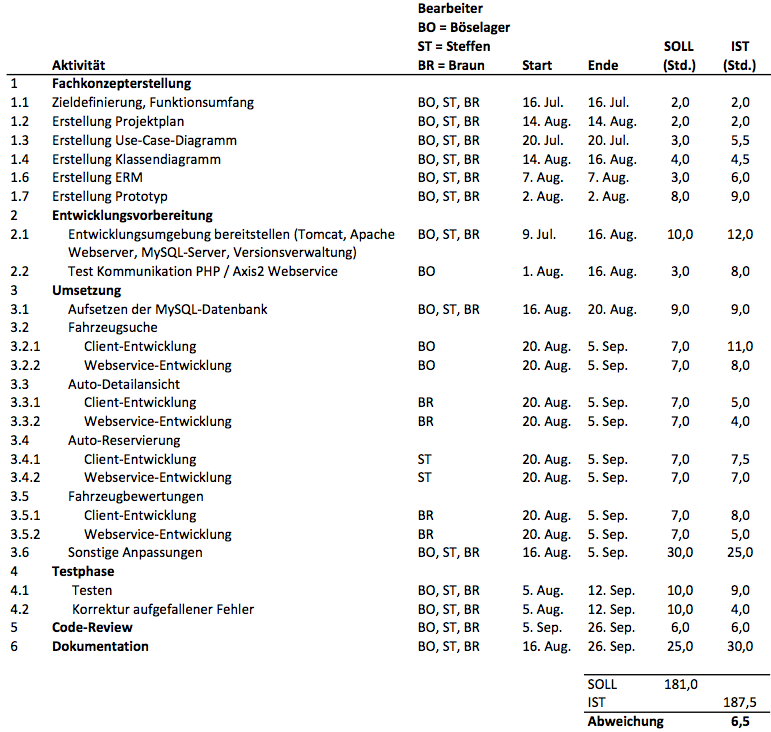


Abbildung 16: Projektplan (inkl. SOLL/IST-Vergleich)

## Fazit

Wie auch in den letzten Projekten hat der Einsatz des Tools „GitHub“ zur Verwaltung und Versionierung des Quellcodes die Zusammenarbeit der Projektmitglieder während der Implementierungsphase positiv beeinflusst.

Weiterhin kann die Entwicklung der iPhone-App in der Programmiersprache „Objective C“ als voller Erfolg gewertet werden. Der dadurch entstandene Zusatzaufwand von ca. 30 Stunden Arbeit hat sich in sofern gelohnt, dass folgende Erkenntnisse gewonnen wurden konnten:

* Kenntnisse über die Grundlagen der Entwicklung von mobilen Anwendungen wurden erlangt.
* Die Syntax von „Objective C“ unterscheidet sich relativ stark zu anderen objektorientierten Programmiersprachen.
* Der Zugriff auf Webservices von „Objective C“ ist im Verhältnis zu PHP deutlich komplizierter, da das Serialisieren bzw. Deserialisieren eines SOAP-Requests bzw. Responses vom Entwickler selbst durchgeführt werden muss.

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