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-Cases 4

# 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.

* Filtermöglichkeit
  + 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.
  + 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.

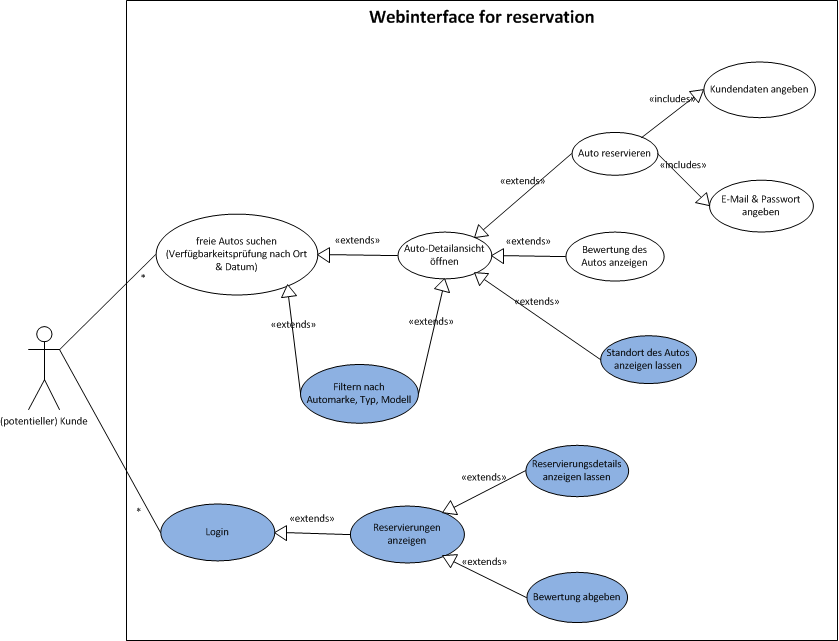


Abbildung : Use-Case Diagramm aus Kundensicht

## 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 : Use-Case Diagramm aus Clientsicht

## Entity-Relationship-Model (ERM)

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

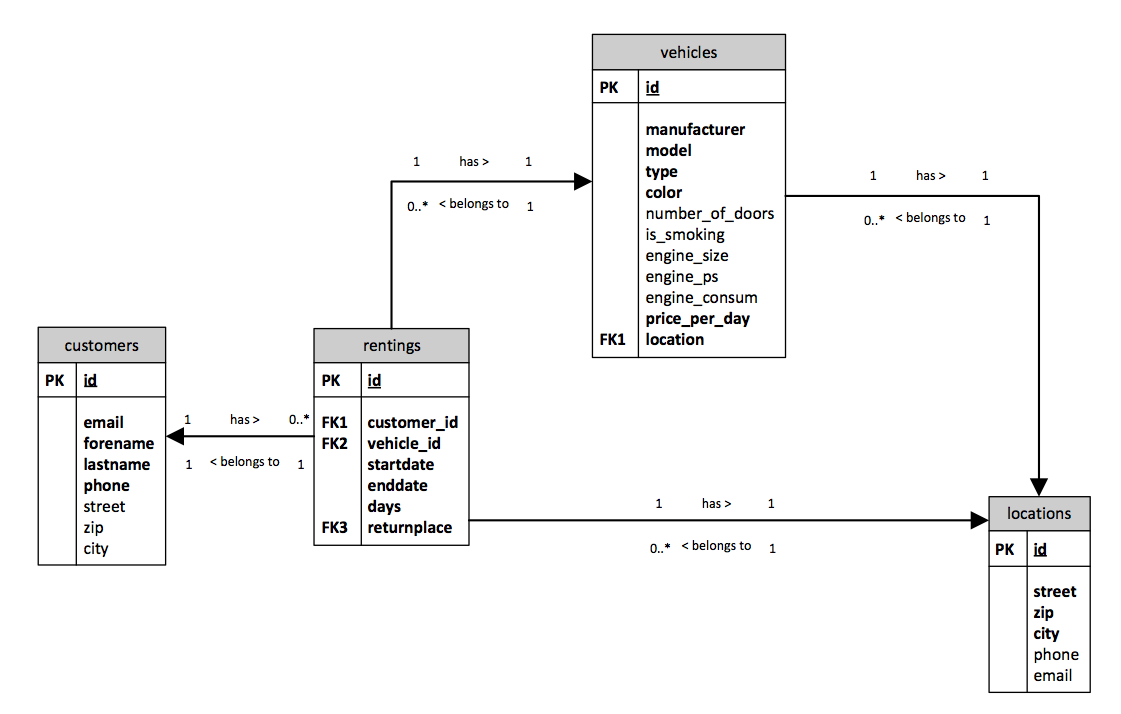


Abbildung : Entity-Relationship-Model

## 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.

## Disk1:Users:gboeselager:Develop:Github:RentACar:Projektdokumente:Klassendiagramm.png

Abbildung : Klassendiagramm

## Prototypen

**Startseite (Suchmaske)**



Abbildung : Prototyp Startseite

**Fahrzeugliste der (verfügbaren) Fahrzeuge**

Abbildung : Prototyp Fahrzeugliste

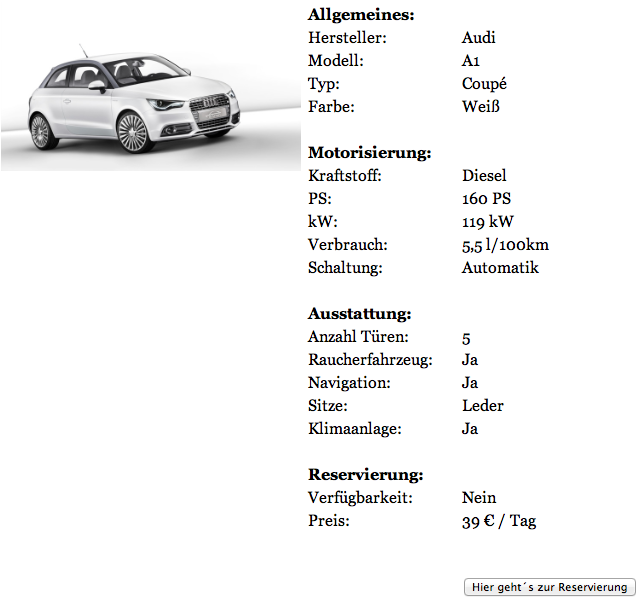
**Detailansicht für ein Fahrzeug**

Abbildung : Prototyp Detailansicht

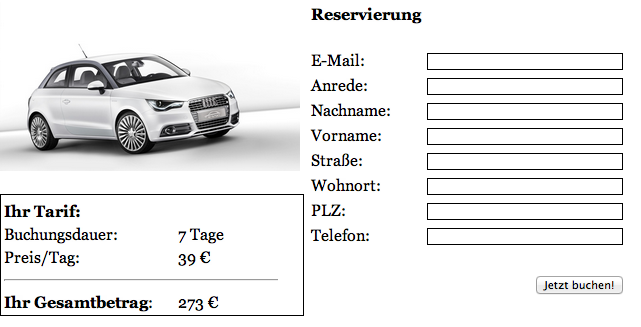
**Reservierung eines Fahrzeuges**

Abbildung : Prototyp Reservierung

## Projektplan

Klassendiagramm auf Clientseite? Gleich wie auf Server?

# 

# 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.

## Tomcat Webserver (Axis2)

Apache AXIS2 herunterladen

Webseite: <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>

ZIP-Archiv an einem geeigneten Ort entpacken

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 Terminal auszuführen:

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

sh /Users/gboeselager/Develop/axis2-1.6.2/bin/axis2server.sh

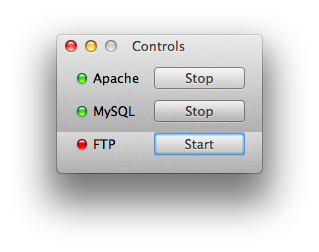
Der Server ist gestartet und kann per http://localhost:8080 (bzw. die Seite mit Informationen über bereitgestellte Webservices http://localhost:8080/axis2)

## XAMPP Webserver und MySQL-Server

XAMMP for Mac herunterladen (enthält Apache und Mysql) und standardmäßig installieren.

Quelle: <http://www.apachefriends.org/de/xampp-macosx.html>

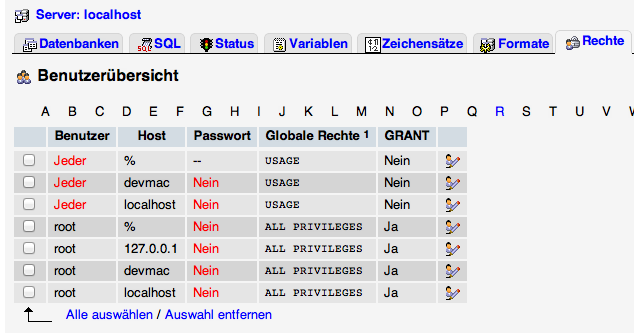
XAMPP-ControlPanel starten



Per Klick auf Start können Apache und MySQl-Server gestartet werden!

## MySQL einrichten

Auf dem MySQL-Server muss der Benutzer „root“ kopiert und bei Host „Jeder Host“ (%) eingetragen werden, damit ist der Zugriff von allen Hosts möglich, auch von nicht-localhost.



## Webserver einrichten

xampp alias anlegen, um auf ein github-verzeichnis zu verweisen:

in der Datei /etc/httpd.conf müssen die Zeilen hinzugefügt werden:

<Directory "/Users/gboeselager/Develop/Github/RentACar/Webinterface">

AllowOverride FileInfo AuthConfig Limit Indexes

Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec

<Limit GET POST OPTIONS>

Order allow,deny

Allow from all

</Limit>

<LimitExcept GET POST OPTIONS>

Order deny,allow

Deny from all

</LimitExcept>

</Directory>

Alias /RentACar "/Users/gboeselager/Develop/Github/RentACar/Webinterface"

Anschließend den Apache-Server neustarten. Damit kann im Browser localhost/RentACar die Webseite aufgerufen werden.

Da per Soap keine Java DateTime-Typen übergeben werden können werden als Ersatz Strings zwischen PHP und Java übergeben, die immer dem folgenden Aufbau entsprechen müssen:

JJJJ-MM-TT HH:MM:SS

Beispiel: „2012-08-19 17:28:19“

