

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	Implementierung	16
4.1	Team.....	16
4.2	Technische Vorgaben / Sicherheit	16
4.3	Programmierstandards	17
4.4	Architektur	18
4.5	Technisches Klassendiagramm	19
4.6	Externe Komponenten, Module und Quellen	20
4.7	Implementierung einer iPhone-App	21
4.7.1	Zielsetzung.....	21
4.7.2	Voraussetzungen	21
4.7.3	Umsetzung	21
4.7.4	Ergebnis	21
5	Testphase	22
6	Projektabschluss & Fazit.....	23
6.1	Erreichung der Anforderungen.....	23
6.2	Soll / Ist - Vergleich	24
6.3	Fazit	25

1 Anforderungsdefinition

Als Ziel gilt die Umsetzung einer Software zur Verwaltung einer Autovermietung. In der Software müssen Standardprozesse, wie z.B. Suche nach verfügbaren Fahrzeugen, Reservierung/Buchung eines Fahrzeuges 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.

1.1 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
 - o Um die Unabhängigkeit von Programmiersprache und Betriebssystem (Interoperabilität) zu gewährleisten, sollen als Middleware Webservices eingesetzt werden.
- Speicherung der Daten in einer Datenbank
 - o Die Daten werden in einer MySQL-Datenbank gespeichert.
- Autosuche (Verfügbarkeit)
 - o Potentielle Kunden können verfügbare Mietwagen zu einem bestimmten Zeitpunkt und Ort anzeigen lassen.
 - o Ergebnis ist eine Liste verfügbarer Autos.
- Autodetailansicht
 - o Dem Kunden wird ein Fahrzeugsteckbrief mit Detailinformationen gezeigt.
 - o Beispielsweise: Anzahl Türen, PS, Farbe, Typ, Modell, Kraftstoff, usw.
- Autoreservierung (bzw. -buchung)
 - o Nach Auswahl eines Fahrzeuges kann der Kunde eine Reservierung durchführen.
 - o Die Angabe von persönlichen Kundendaten (z.B. E-Mail, Passwort, Vorname, Nachname, Anschrift) ist bei der ersten Reservierung notwendig.
 - o Als Bestandskunde ist durch das Angeben von E-Mail-Adresse und Passwort ein Login möglich, sodass die persönlichen Kundendaten nicht nochmals eingegeben werden müssen.
 - o Auf Grundlage der Reservierungsinformationen muss eine Preiskalkulation durchgeführt werden.
- Plausibilitätsprüfung bei der Eingabe
 - o Die angegebene E-Mail-Adresse des Kunden wird auf Plausibilität untersucht.

1.2 Optionale Funktionalitäten

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

- Filterfunktion
 - o Der Kunde hat die Möglichkeit die Ergebnisliste der verfügbaren Fahrzeuge nach Automarke, Typ, Modell zu filtern.
- Autostandort anzeigen lassen
 - o Der Standort des Fahrzeugs kann in einer Maps-Ansicht angezeigt werden.
- Login für Bestandskunden
 - o 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
 - o Eingeloggte Kunden können die von ihnen angemieteten Fahrzeuge bewerten.

1.3 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.

2 Planung und Fachkonzeption

2.1 Use-Case Diagramm (Kundensicht)

Abbildung 1 zeigt das Use-Case Diagramm aus Sicht des potentiellen Kunden. Blau hinterlegte Use-Cases sind optional.

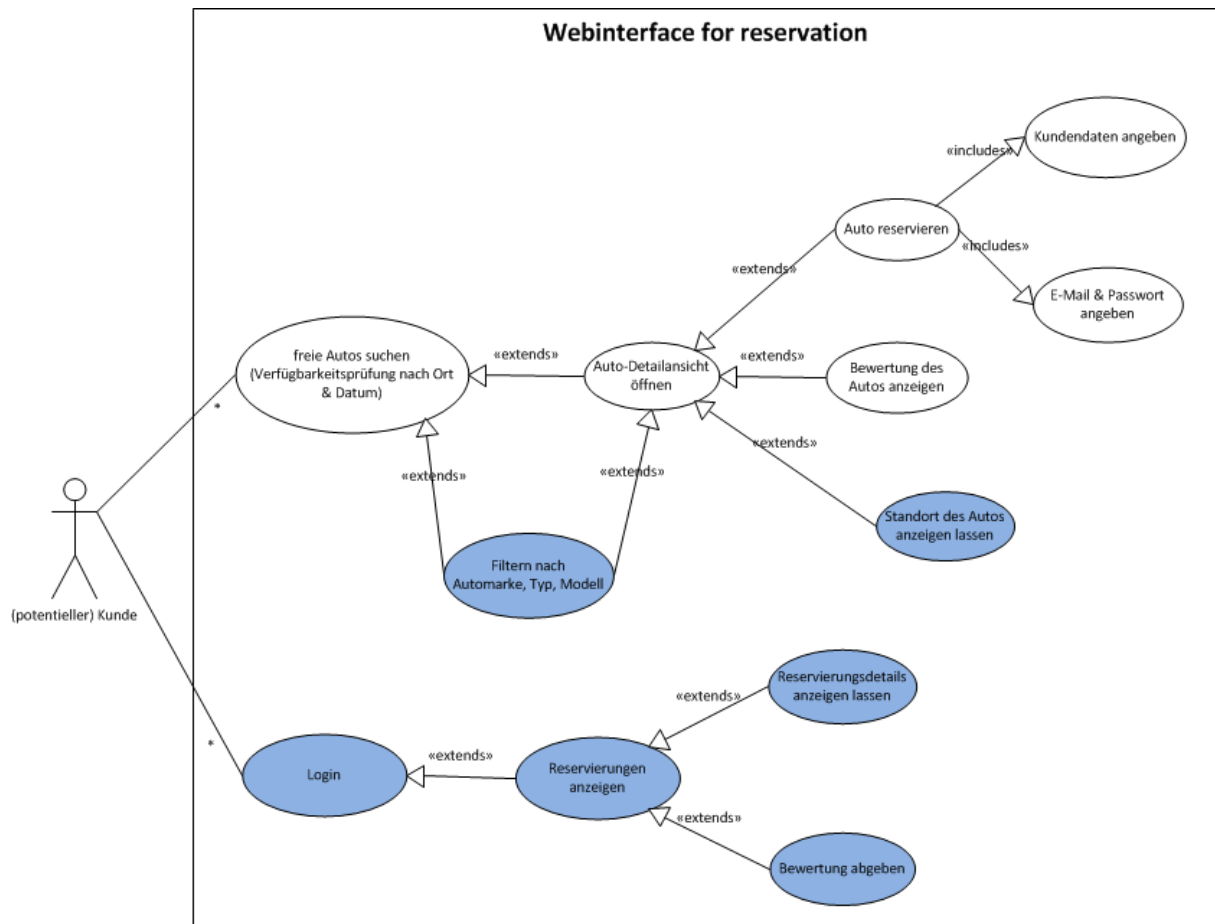


Abbildung 1: Use-Case Diagramm aus Kundensicht

2.2 Use-Case Diagramm (Clientsicht)

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

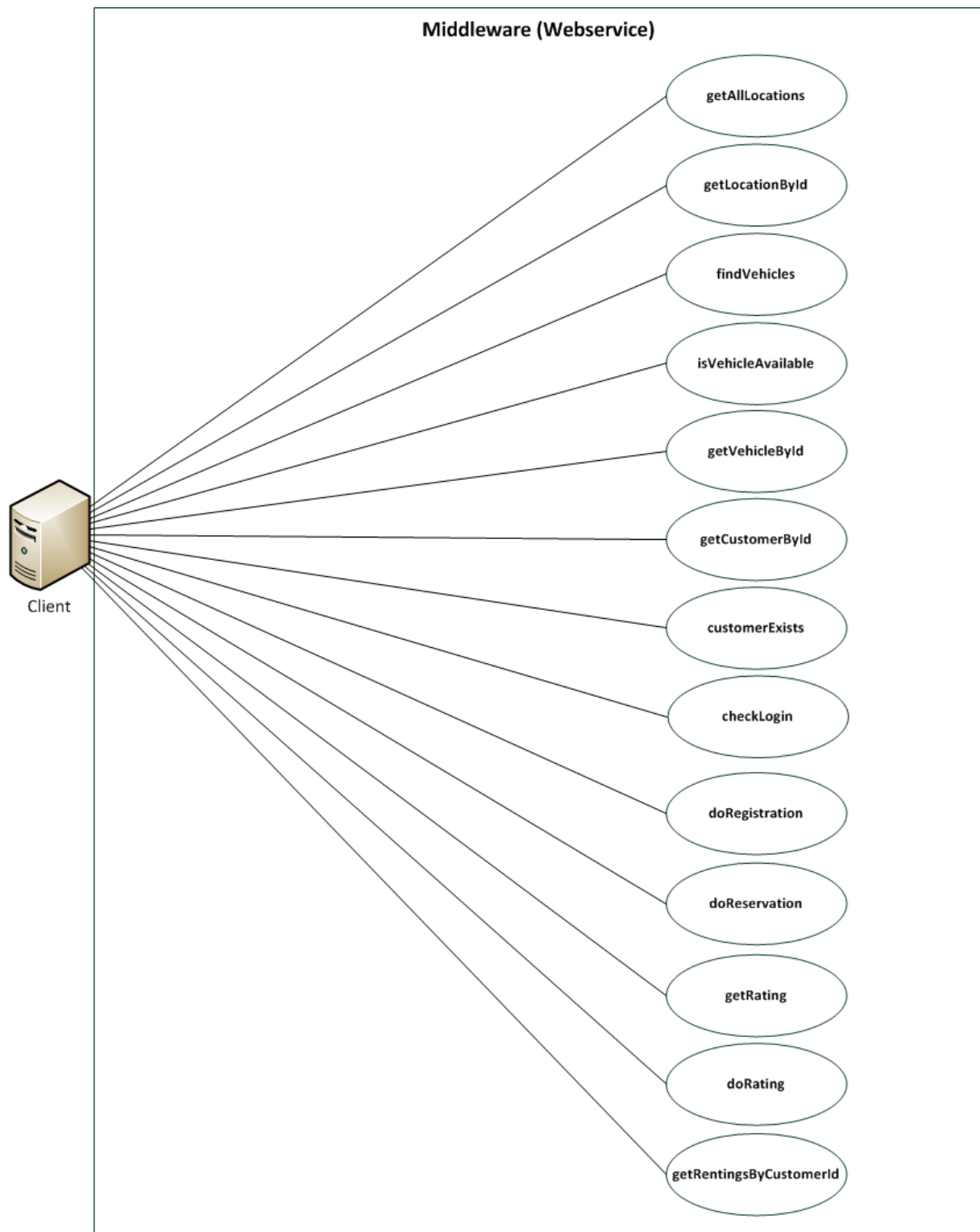


Abbildung 2: Use-Case Diagramm aus Clientsicht

2.3 Entity-Relationship-Model (ERM)

Das folgende Entity-Relationship-Model zeigt die geplante Struktur der Datenbank und die Beziehungen zwischen den einzelnen Entitäten/Tabellen.

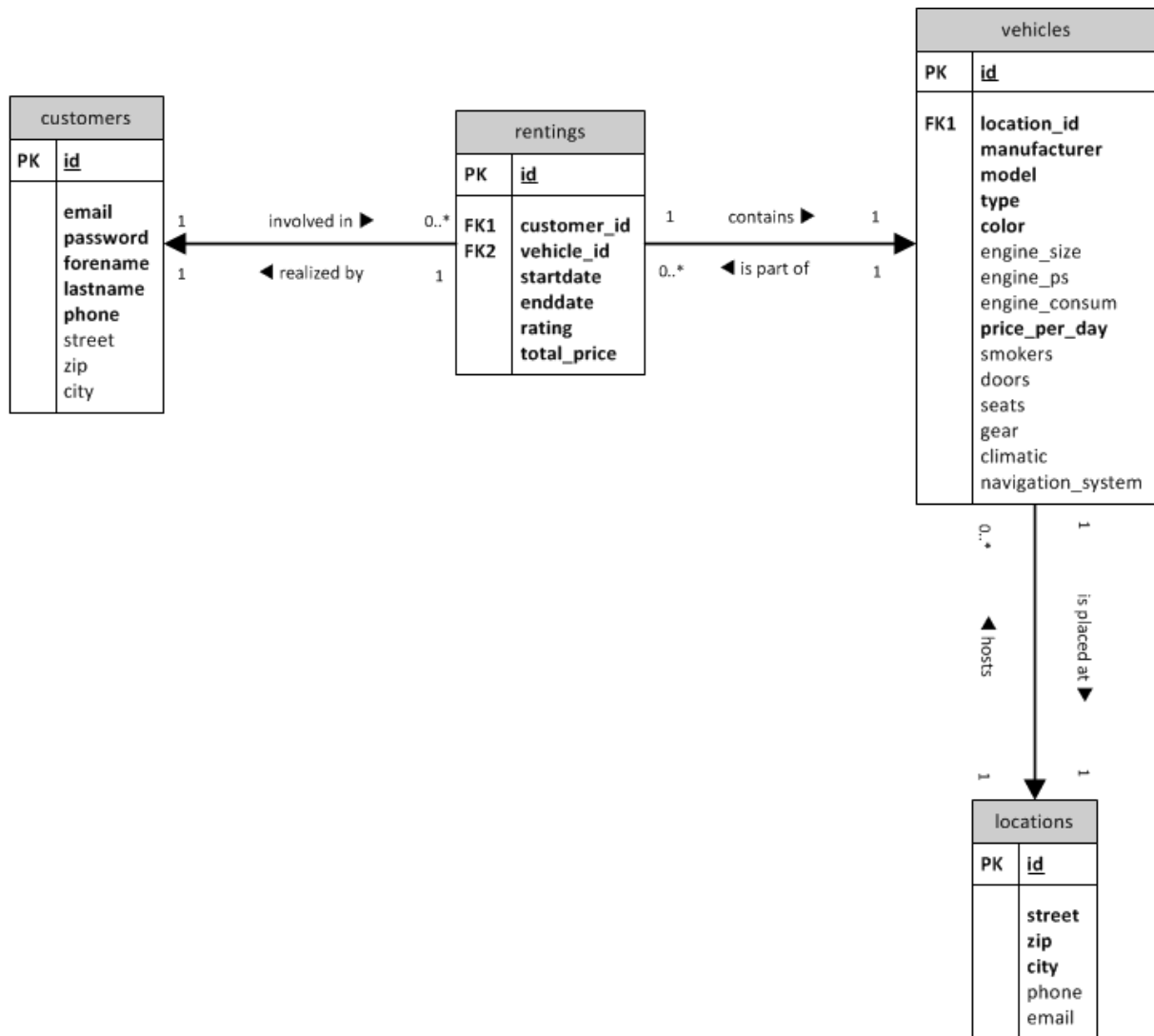


Abbildung 3: Entity-Relationship-Model

2.4 Fachliches Klassendiagramm

Folgend ist das fachliche Klassendiagramm aufgeführt, das die fachlichen Zusammenhänge/Beziehungen zwischen den einzelnen Klassen darstellt. Das technische Klassendiagramm (siehe 4.5) zeigt wie dieses Konzept technisch umgesetzt wurde. Hallo Arsch!

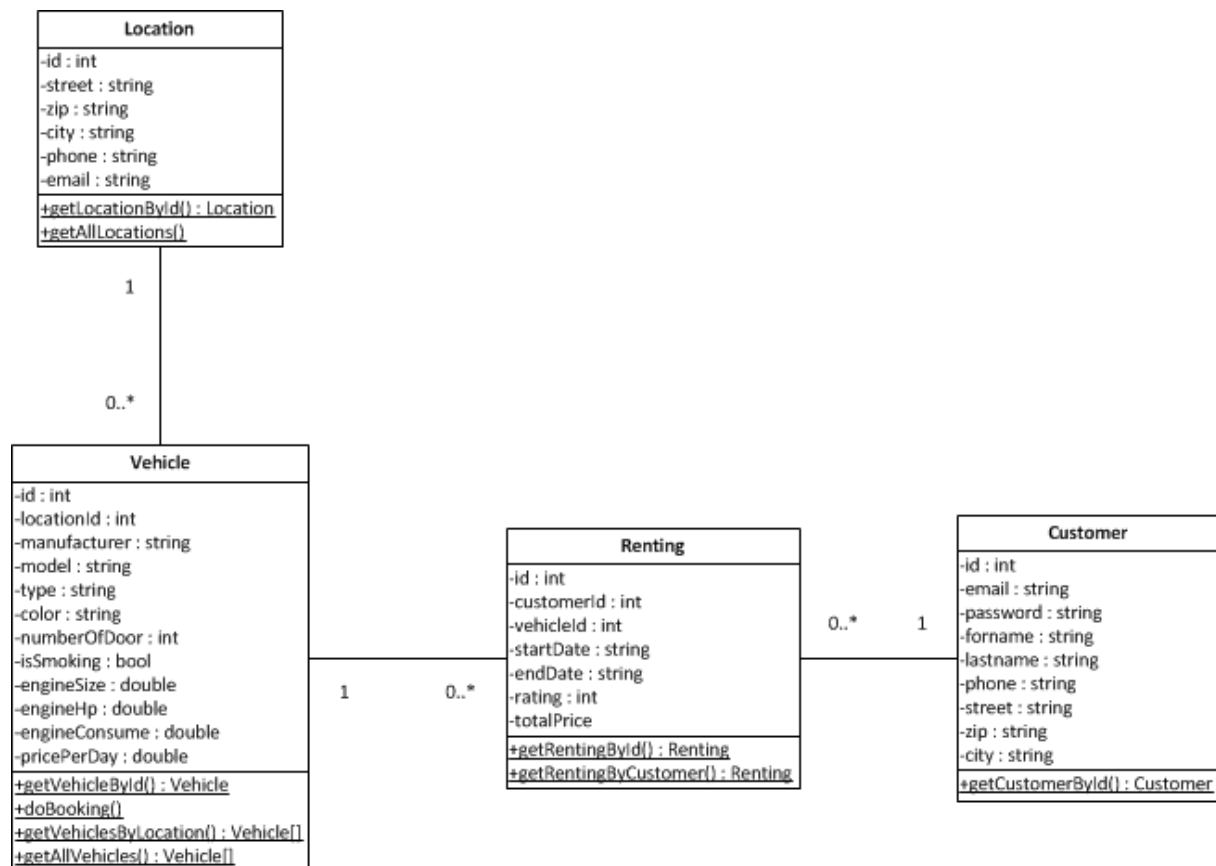


Abbildung 4: Fachliches Klassendiagramm

2.5 Prototypen

Startseite (Suchmaske)

RentACar Autovermietung

Kundenlogin:
E-Mail: Passwort:

Abholung
Ort wählen:
Datum: Uhrzeit:

Rückgabe
Datum: Uhrzeit:

August 2012						
Mo	Di	Mi	Do	Fr	Sa	So
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Abbildung 5: Prototyp Startseite

Fahrzeugliste der (verfügbaren) Fahrzeuge

RentACar Autovermietung			
	Audi A1 Standort: Osnabrück	Türen: 4 PS: 123 Farbe: rot	Details anzeigen Jetzt reservieren
	Mini Cabrio Standort: Bielefeld	Türen: 3 PS: 272 Farbe: gelb	Details anzeigen Jetzt reservieren
	Mercedes A-Klasse Standort: Bielefeld	Türen: 4 PS: 250 Farbe: rot	Details anzeigen Jetzt reservieren

Abbildung 6: Prototyp Fahrzeugliste

Detailansicht für ein Fahrzeug

	Allgemeines:	
	Hersteller:	Audi
	Modell:	A1
	Typ:	Coupé
	Farbe:	Weiß
	Motorisierung:	
	Kraftstoff:	Diesel
	PS:	160 PS
	kW:	119 kW
	Verbrauch:	5,5 l/100km
	Schaltung:	Automatik
	Ausstattung:	
	Anzahl Türen:	5
	Raucherfahrzeug:	Ja
	Navigation:	Ja
Sitze:	Leder	
Klimaanlage:	Ja	
Reservierung:		
Verfügbarkeit:	Nein	
Preis:	39 € / Tag	
Hier geht's zur Reservierung		

Abbildung 7: Prototyp Detailansicht

Reservierung eines Fahrzeuges


	Reservierung										
	E-Mail:	<input type="text"/>									
	Anrede:	<input type="text"/>									
	Nachname:	<input type="text"/>									
	Vorname:	<input type="text"/>									
	Straße:	<input type="text"/>									
	Wohnort:	<input type="text"/>									
	PLZ:	<input type="text"/>									
	Telefon:	<input type="text"/>									
	<table border="1"> <tr> <td colspan="2">Ihr Tarif:</td> </tr> <tr> <td>Buchungsdauer:</td> <td>7 Tage</td> </tr> <tr> <td>Preis/Tag:</td> <td>39 €</td> </tr> <tr> <td colspan="2"><hr/></td> </tr> <tr> <td>Ihr Gesamtbetrag:</td> <td>273 €</td> </tr> </table>		Ihr Tarif:		Buchungsdauer:	7 Tage	Preis/Tag:	39 €	<hr/>		Ihr Gesamtbetrag:
Ihr Tarif:											
Buchungsdauer:	7 Tage										
Preis/Tag:	39 €										
<hr/>											
Ihr Gesamtbetrag:	273 €										
Jetzt buchen!											

Abbildung 8: Prototyp Reservierung

2.6 Testplan

In der folgenden Abbildung sind die geplanten Testfälle aufgelistet. Die detaillierte Testphase, inklusive entsprechendem Protokoll, kann aus Kapitel 4.7 entnommen werden.

Akteur	TestNr.	Testfall
Nicht eingeloggter Benutzer	T001	Login
	T002	Zeitraum für die Reservierung auswählen
	T003	Listenansicht aller verfügbaren Fahrzeuge
	T004	Detailübersicht
	T005	Fahrzeug auswählen
	T006	Registrierung
	T007	Registrierung abschließen
	T008	Abschluss der Reservierung
	T009	Preiskalkulation
	T010	Doppelbuchung eines Fahrzeuges verhindern
eingeloggter Benutzer	T011	Reservierung durchführen
	T012	Übersicht über gemietete Fahrzeuge
	T013	Bewertung für Fahrzeug abgeben
	T014	Logout

Abbildung 9: Testplan

2.7 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.

		Bearbeiter BO = Böselager ST = Steffen BR = Braun	Start	Ende	SOLL (Std.)	IST (Std.)
Aktivität						
1	Fachkonzepterstellung					
1.1	Zieldefinierung, Funktionsumfang	BO, ST, BR	16. Jul.	16. Jul.	2,0	2,0
1.2	Erstellung Projektplan	BO, ST, BR	14. Aug.	14. Aug.	2,0	2,0
1.3	Erstellung Use-Case-Diagramm	BO, ST, BR	20. Jul.	20. Jul.	3,0	1,5
1.4	Erstellung Klassendiagramm	BO, ST, BR	14. Aug.	16. Aug.	4,0	
1.6	Erstellung ERM	BO, ST, BR	7. Aug.	7. Aug.	3,0	
1.7	Erstellung Prototyp	BO, ST, BR	2. Aug.	2. Aug.	8,0	
2	Entwicklungsvorbereitung					
2.1	Entwicklungsumgebung bereitstellen (Tomcat, Apache Webserver, MySQL-Server, Versionsverwaltung)	BO, ST, BR	9. Jul.	16. Aug.	10,0	4,0
2.2	Test Kommunikation PHP / Axis2 Webservice	BO	1. Aug.	16. Aug.	3,0	1,0
3	Umsetzung					
3.1	Aufsetzen der MySQL-Datenbank	BO, ST, BR	16. Aug.	20. Aug.	9,0	
3.2	Fahrzeugsuche					
3.2.1	Client-Entwicklung	BO	20. Aug.	5. Sep.	7,0	
3.2.2	Webservice-Entwicklung	BO	20. Aug.	5. Sep.	7,0	
3.3	Auto-Detailansicht					
3.3.1	Client-Entwicklung	BR	20. Aug.	5. Sep.	7,0	
3.3.2	Webservice-Entwicklung	BR	20. Aug.	5. Sep.	7,0	
3.4	Auto-Reservierung					
3.4.1	Client-Entwicklung	ST	20. Aug.	5. Sep.	7,0	
3.4.2	Webservice-Entwicklung	ST	20. Aug.	5. Sep.	7,0	
3.5	Fahrzeugbewertungen					
3.5.1	Client-Entwicklung	BR	20. Aug.	5. Sep.	7,0	
3.5.2	Webservice-Entwicklung	BR	20. Aug.	5. Sep.	7,0	
3.6	Sonstige Anpassungen	BO, ST, BR	16. Aug.	5. Sep.	30,0	
4	Testphase					
4.1	Testen	BO, ST, BR	5. Aug.	12. Sep.	10,0	
4.2	Korrektur aufgefallener Fehler	BO, ST, BR	5. Aug.	12. Sep.	10,0	
5	Code-Review	BO, ST, BR	5. Sep.	26. Sep.	6,0	
6	Dokumentation	BO, ST, BR	16. Aug.	26. Sep.	25,0	3,0
					SOLL	181,0
					IST	13,5
					Abweichung	167,5

Abbildung 10: Projektplan

3 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.

3.1 Programmiersprachen

Abbildung 11 zeigt die im Projekt verwendeten Programmiersprachen.

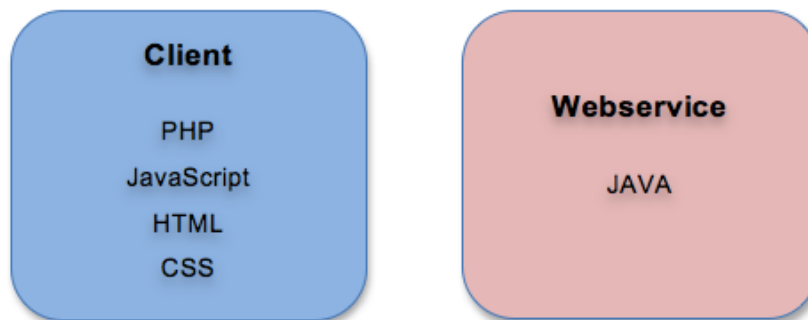


Abbildung 11: Programmiersprachen

3.2 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.

3.3 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 GitHub 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/>

3.4 Tomcat Webserver (inkl. Axis2-Framework)

Als Webserver-Software kommt Apache Tomcat inkl. Axis2-Framework für Webservices (Version 1.6.2) zum Einsatz. Darüber hinaus 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"
```

2. Start des Axis2-Servers.

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

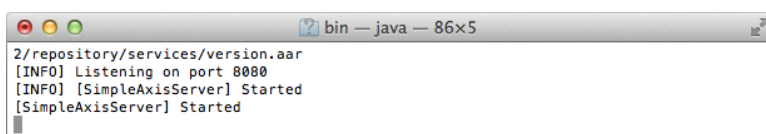


Abbildung 12: Erfolgreicher Start des Webservers

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.

3.5 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.

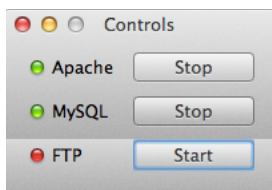


Abbildung 13: XAMPP ControlPanel

Wichtige Pfade und URL's:

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

Webseite: `http://localhost`

MySQL-Administration: `http://localhost/phpmyadmin`

3.6 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 Ubuntu 10.4 LTS 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 10.4 LTS
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
```

2. Apache2 Webserver installieren und anschließend starten

```
sudo apt-get install apache2
sudo /etc/init.d/apache2 start
```

3. PHP5 installieren

```
sudo apt-get install php5 libapache2-mod-php5
```

4. MySQL installieren

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

5. PhpMyAdmin installieren

```
apt-get install phpmyadmin
```

6. 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 (Datenbankadministration):	http://193.22.73.246/mysqladmin User: root; Passwort: <i>leer</i>
Axis2 (Infoseite):	193.22.73.246/axis2

4 Implementierung

4.1 Team

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

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

durchgeführt.

4.2 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.

4.3 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 Klassennamen, 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.

4.4 Architektur

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 Präsentationsschicht aufgeteilt.

In der Präsentationsschicht werden alle Oberflächenkomponenten, die der Interaktion mit dem Benutzer dienen, untergebracht. In der Fachkonzeptschicht befinden sich zentrale Logiken (bzw. Geschäftslogiken) zur Steuerung der Software, die in diesem Projekt als Webservice bereitgestellt werden. Die Datenhaltungsschicht besteht aus einer Datenbank und ist für die Datenspeicherung und das Laden von Daten zuständig.¹

Folgend ist die in diesem Projekt gewählte Architektur aufgezeigt.

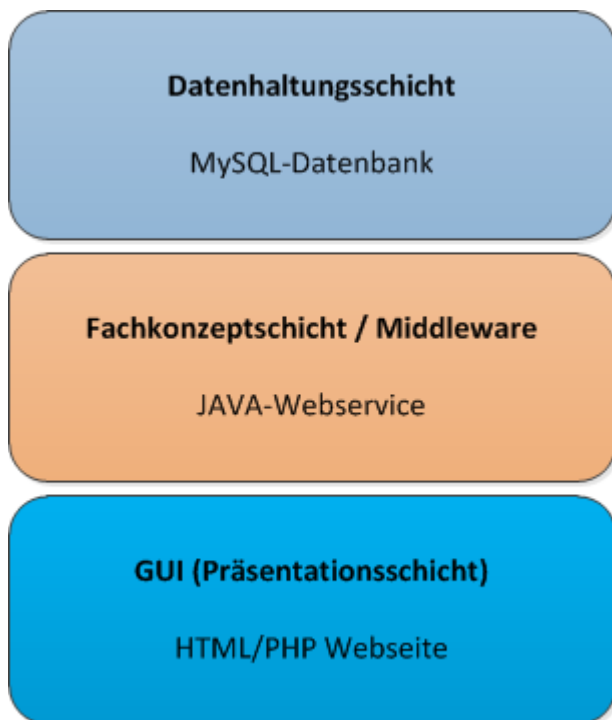


Abbildung 14: Drei-Schichten-Architektur

¹ vgl. <http://www.itwissen.info/definition/lexikon/Drei-Schichten-Architektur-three-tier-architecture.html>

4.5 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.

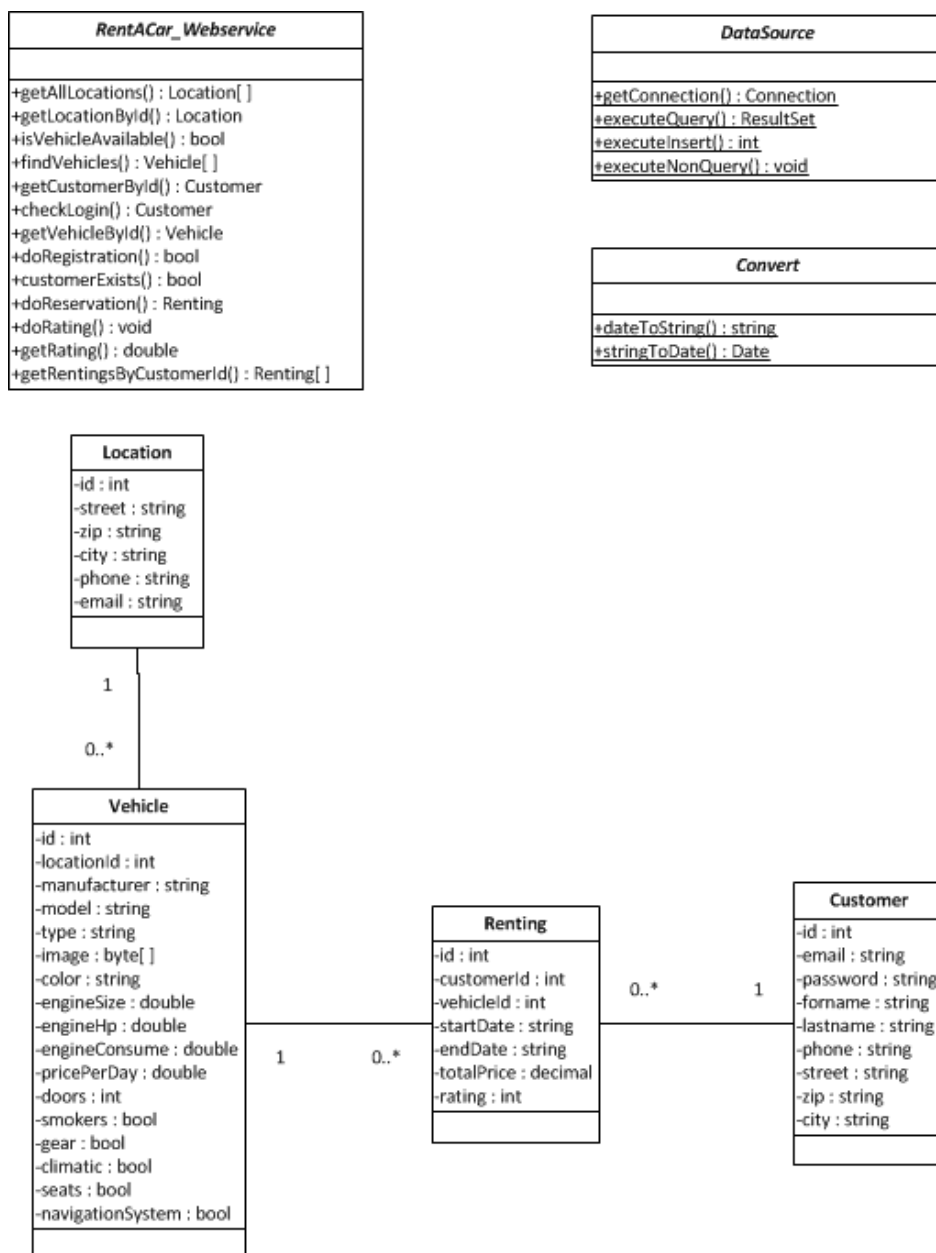


Abbildung 15: Technisches Klassendiagramm

4.6 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)

jQuery ist eine kostenlose JavaScript Library, die das Entwickeln von JavaScript's vereinfacht. Zusätzlich gibt das ist übrigens ein Fehler 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 anzuzeigen.

Quelle & Download: <http://jqueryui.com>

Dokumentation: <http://jqueryui.com/demos>

4.7 Implementierung einer iPhone-App

4.7.1 Zielsetzung

Die RentACar App² für das Apple iPhone soll eine Ergänzung zu dem HTML/PHP-Webclient darstellen. Es besteht jedoch nicht der Anspruch alle Funktionalitäten des Webclients auch in der App bereitzustellen.

Es soll hierbei vielmehr darum gehen, beispielhaft einzelne Funktionen zur Verfügung zu stellen, um die grundsätzliche Implementierung eines Webservice-Zugriffs auf einem mobilen Endgerät zu demonstrieren.

Die für die App verwendete Beispielfunktion soll die Methode „getVehicleById“ des Webservices aufrufen und das Ergebnis auf dem iPhone darstellen. Das gewünschte Ergebnis ist die Ausgabe eines Fahrzeuges inkl. einiger Fahrzeugdetails und der Abbildung.

4.7.2 Voraussetzungen

Die Erstellung einer iPhone App setzt einen Computer mit Mac OSX Betriebssystem voraus. Darüber hinaus werden die kostenlosen Entwicklungswerkzeuge XCode und iOS-Simulator benötigt. Die zu verwendende Programmiersprache ist Objective-C³.

4.7.3 Umsetzung

Zur Kommunikation des Clients (iPhone App) mit dem Java-Webservice muss die App in der Lage sein SOAP-Nachrichten zu versenden, sowie empfangene SOAP-Nachrichten zu interpretieren. Hierzu wurde ein externes Tool mit dem Namen „Sudzc“ verwendet (www.sudzc.com).

Dieser Dienst stellt einen kompletten Objective-C Client zur Verfügung der in der Lage ist mit dem angegebenen Webservice zu kommunizieren. Dazu muss lediglich die WSDL-Datei des RentACar Webservices auf www.sudzc.com hochgeladen werden. „Sudzc“ generiert anschließend alle benötigten Klassen und stellt über das beigefügte TouchXML-Framework alle Funktionalitäten zum Parsen von SOAP-Dateien (XML) zur Verfügung.

4.7.4 Ergebnis

Die iPhone-App konnte erfolgreich implementiert werden, sodass eine Detailseite eines Fahrzeugs in der App angezeigt werden kann. Hast du das gelesen, Jung? Im Fazit (siehe Kapitel 6.3) wird das Ergebnis detailliert beurteilt.

² App ist die Kurzform des englischen Wortes Application (zu deutsch: Anwendung)

³ Eine Erweiterung der Programmiersprache C um objektorientierte Programmierkomponenten

5 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.

Akteur	TestN	Testfall	Fallbeschreibung / Erwartetes Ergebnis	Testergebnis	Fehler
Nicht eingeloggter Benutzer	T001	Login	Der Benutzer kann sich seiner E-Mail Adresse und einem Passwort in das System anmelden und ein Login erfolgt.	✓	keine
	T002	Zeitraum für die Reservierung auswählen	Der Benutzer kann einen gewünschten Zeitraum angeben, in dem der er ein Auto reservieren möchte und eine Liste verfügbarer Fahrzeuge wird angezeigt.	✓	keine
	T003	Listenansicht aller verfügbaren Fahrzeuge	Nach der Eingabe des gewünschten Zeitraumes, erscheint eine Listenübersicht, die alle verfügbaren Fahrzeuge und deren Details enthält.	✓	keine
	T004	Detailübersicht	In der Detailübersicht werden alle Eigenschaften /Details eines Fahrzeuges angezeigt.	✓	keine
	T005	Fahrzeug auswählen	Der Benutzer kann sich aus einer Liste ein Fahrzeug auswählen und sich dazu Details anzeigen lassen bzw. eine Reservierung durchführen.	✓	keine
	T006	Registrierung	Ist der Benutzer nicht eingeloggt, bekommt er die Möglichkeit sich als existierender Kunde (mit E-Mail und Passwort) anzumelden oder sich mit seinen persönlichen Daten neu zu registrieren und damit einen Kundenaccount anzulegen.	✗	Fehler: Es wird kein Fehler angezeigt, wenn das Feld E-Mail Adresse leer gelassen wurde. Fehler wurde behoben!
	T007	Registrierung abschließen	Nach erfolgreicher Eingabe der persönlichen Daten wird der Kunde automatisch eingeloggt und kann mit der Reservierung fortfahren.	✓	keine
	T008	Abschluss der Reservierung	Beim Abschließen der Reservierung bekommt der Benutzer eine Bestätigungsseite angezeigt und die Reservierung wurde erfolgreich durchgeführt.	✓	keine
	T009	Preiskalkulation	Für den Mietzeitraum wird eine Preiskalkulation auf Basis des Tagespreises durchgeführt, die die anfallenden Kosten ermittelt.	✗	Fehler: Ermittlung der Miettage war nicht korrekt. Fehler wurde behoben!
	T010	Doppelbuchung eines Fahrzeuges verhindern	Ein vermietetes Fahrzeug kann nicht 2 mal im gleichen Zeitraum vermietet werden.	✓	keine
eingeloggter Benutzer	T011	Reservierung durchführen	Eine Reservierung kann nur nach erfolgreichem Login durchgeführt werden. Bei Reservierung wird dem Kunden das Fahrzeug für den Mietzeitraum zugeordnet.	✓	keine
	T012	Übersicht über gemietete Fahrzeuge	Alle jemals vom Kunden gemieteten Fahrzeuge werden im Kundenbereich aufgelistet und er hat die Möglichkeit diese zu bewerten.	✓	keine
	T013	Bewertung für Fahrzeug abgeben	Mit einem Klick auf den entsprechenden "Stern" (1-5 Punkte) wird die Bewertung des Fahrzeuges durchgeführt. Dieser Mietvorgang kann dann nicht nochmals bewertet werden.	✓	keine
	T014	Logout	Die Benutzersession wird zerstört und der Benutzer wird vom System abgemeldet.	✓	keine

Abbildung 16: Testplan & Fehlerkorrektur

6 Projektabschluss & Fazit

6.1 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
- Würstchen

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.

6.2 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.

		Bearbeiter			SOLL (Std.)	IST (Std.)
		BO = Böselager ST = Steffen BR = Braun				
	Aktivität		Start	Ende		
1	Fachkonzepterstellung					
1.1	Zieldefinierung, Funktionsumfang	BO, ST, BR	16. Jul.	16. Jul.	2,0	2,0
1.2	Erstellung Projektplan	BO, ST, BR	14. Aug.	14. Aug.	2,0	2,0
1.3	Erstellung Use-Case-Diagramm	BO, ST, BR	20. Jul.	20. Jul.	3,0	5,5
1.4	Erstellung Klassendiagramm	BO, ST, BR	14. Aug.	16. Aug.	4,0	4,5
1.6	Erstellung ERM	BO, ST, BR	7. Aug.	7. Aug.	3,0	6,0
1.7	Erstellung Prototyp	BO, ST, BR	2. Aug.	2. Aug.	8,0	9,0
2	Entwicklungsvorbereitung					
2.1	Entwicklungsumgebung bereitstellen (Tomcat, Apache Webserver, MySQL-Server, Versionsverwaltung)	BO, ST, BR	9. Jul.	16. Aug.	10,0	12,0
2.2	Test Kommunikation PHP / Axis2 Webservice	BO	1. Aug.	16. Aug.	3,0	8,0
3	Umsetzung					
3.1	Aufsetzen der MySQL-Datenbank	BO, ST, BR	16. Aug.	20. Aug.	9,0	9,0
3.2	Fahrzeugsuche					
3.2.1	Client-Entwicklung	BO	20. Aug.	5. Sep.	7,0	11,0
3.2.2	Webservice-Entwicklung	BO	20. Aug.	5. Sep.	7,0	8,0
3.3	Auto-Detailansicht					
3.3.1	Client-Entwicklung	BR	20. Aug.	5. Sep.	7,0	5,0
3.3.2	Webservice-Entwicklung	BR	20. Aug.	5. Sep.	7,0	4,0
3.4	Auto-Reservierung					
3.4.1	Client-Entwicklung	ST	20. Aug.	5. Sep.	7,0	7,5
3.4.2	Webservice-Entwicklung	ST	20. Aug.	5. Sep.	7,0	7,0
3.5	Fahrzeugbewertungen					
3.5.1	Client-Entwicklung	BR	20. Aug.	5. Sep.	7,0	8,0
3.5.2	Webservice-Entwicklung	BR	20. Aug.	5. Sep.	7,0	5,0
3.6	Sonstige Anpassungen	BO, ST, BR	16. Aug.	5. Sep.	30,0	25,0
4	Testphase					
4.1	Testen	BO, ST, BR	5. Aug.	12. Sep.	10,0	9,0
4.2	Korrektur aufgefallener Fehler	BO, ST, BR	5. Aug.	12. Sep.	10,0	4,0
5	Code-Review	BO, ST, BR	5. Sep.	26. Sep.	6,0	6,0
6	Dokumentation	BO, ST, BR	16. Aug.	26. Sep.	25,0	30,0
					SOLL	181,0
					IST	187,5
					Abweichung	6,5

Abbildung 17: Projektplan (inkl. SOLL/IST-Vergleich)

6.3 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:

- Kenntnisse über die Grundlagen der Entwicklung von mobilen Anwendungen wurden erlangt.
- Die Syntax von Objective-C unterscheidet sich relativ stark zu anderen objektorientierten Programmiersprachen.
- Durch den iOS-Simulator, der ein iPhone auf dem Entwicklungsrechner simuliert, ist das Testen der App relativ unkompliziert.
- 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. Der Einsatz des Tools „Sudzc“ erspart den Entwicklern diesen Aufwand.

Als Nachteil kann jedoch gewertet werden, dass ein Veröffentlichen der Lösung (z.B. im Apple AppStore) einen Developer-Account bei Apple voraussetzt, der jährlich 99\$ kostet. Leider ist dadurch ein Test direkt auf dem iPhone nicht möglich.

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.