

---

Systementwurfs-Praktikum

**driving-e-car.de**

## Systemarchitektur

Autor(en): **Fahri Kus, Matthias Eberlein, Simon Kreuziger, Florian Heinrich**

Datum 10.12.2018

Version 1.1

## Inhalt

<b>1</b>	<b>Datenschema .....</b>	<b>3</b>
<b>2</b>	<b>Komponentenarchitektur (Bausteinsicht) .....</b>	<b>4</b>
<b>3</b>	<b>Komponentenspezifikation .....</b>	<b>6</b>
<b>4</b>	<b>Komponentenverteilung .....</b>	<b>15</b>

## Revisionshistorie

<b>Version</b>	<b>Datum</b>	<b>Autor</b>	<b>Bemerkungen</b>
0.1	05.11.2018	Simon Kreuziger, Florian Heinrich	Initial Version
0.2	09.11.2018	Simon Kreuziger, Florian Heinrich	Diagramme erstellt und eingefügt
0.3	12.11.2018	Simon Kreuziger, Florian Heinrich	Eindeutigere und einheitlich Packet Benennung
1.0	12.11.2018	Fahri Kus, Matthias Eberlein, Simon Kreuziger, Florian Heinrich	Abschließende Qualitätssicherung für MS2
1.1	10.12.2018	Fahri Kus, Matthias Eberlein, Simon Kreuziger, Florian Heinrich	Aktualisierung des kompletten Dokuments auf Basis des aktuellen Entwicklungsstandes und Verbesserungsvorschlägen von Herrn Prof. Würzberger

# 1 Datenschema

Das Datenschema des Systems ist in der Systemspezifikation ausreichend genau beschrieben worden. Eine detailliertere Spezifikation dieses Datenschema wird deshalb in diesem Kapitel nicht erstellt.

## 2 Komponentenarchitektur (Bausteinsicht)

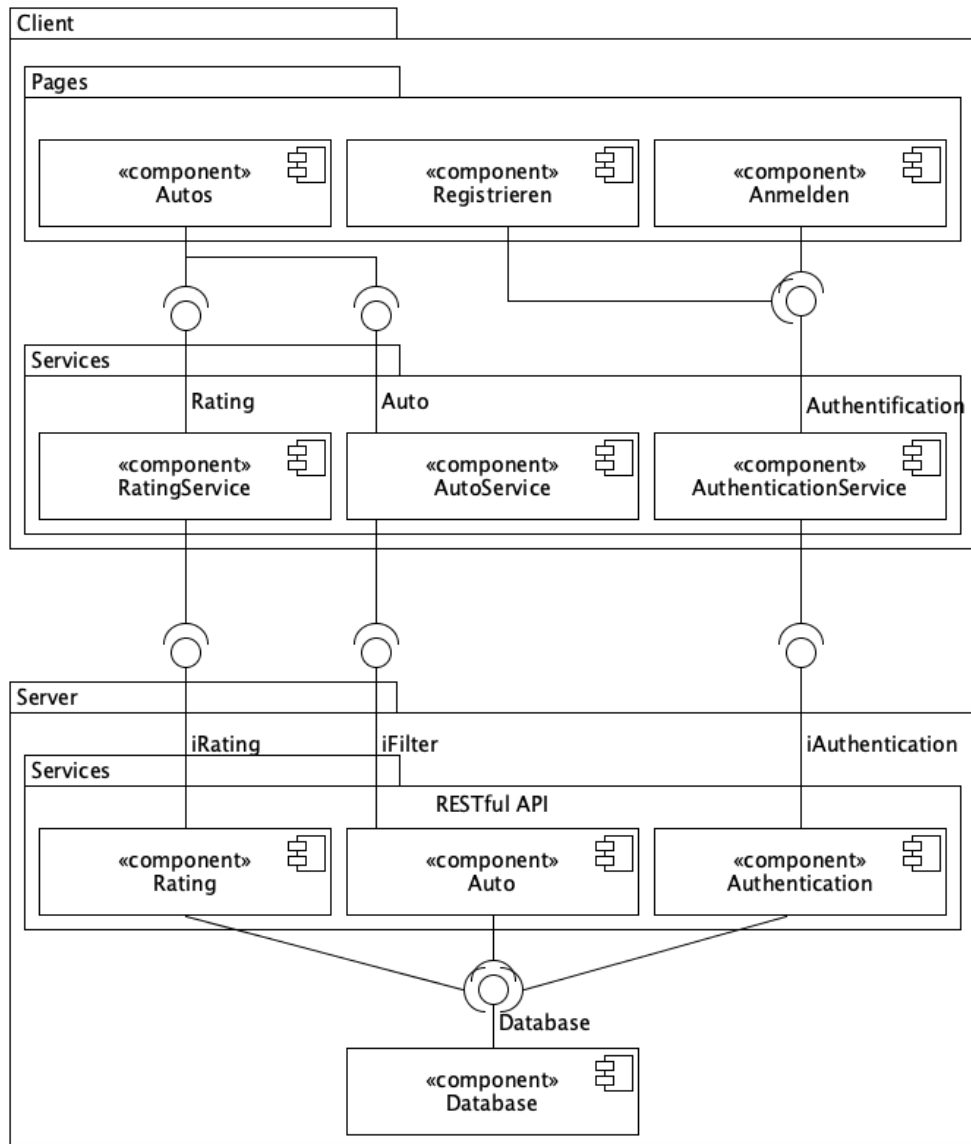


Abbildung 1.1: Komponentendiagramm: Internes System

In Abbildung 1.1 ist ein REST-Architekturstil basiertes System zu sehen, dieses ist traditionell zustandslos.

Die Komponenten<sup>1</sup> werden in kommenden Kapiteln näher spezifiziert, wobei der Inhalt des Packages: „Pages“ die GUI beinhaltet und nicht weiter erläutert wird.

<sup>1</sup> Eine Komponente in diesem System:

Client-Seite: Ist in Form eines Ordners zu finden. In solch einem Ordner liegt u.a. eine TypeScript-Datei, diese liefert die Logik und bietet Schnittstellen in Form von Funktionen.

Server-Seite: Ist in Form von einzelnen JavaScript-Dateien zu finden.

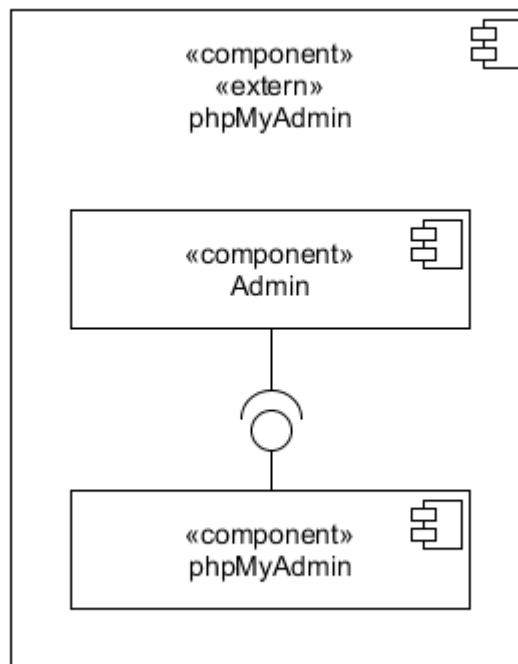


Abbildung 1.2: Komponentendiagramm: Externes System

Die Lastenfälle des Admins werden auf eine bereits existierende, externe Komponente, [phpMyAdmin](https://www.phpmyadmin.net/)<sup>2</sup>, ausgelagert. Alle Lastenfälle werden durch diese Komponente realisiert. Zusätzlich bietet diese viele weitere mächtige Vorteile wie Beispielsweise eine Ex- und Import Schnittstelle.

phpMyAdmin bietet auch eine umfangreiche Dokumentation unter folgendem Link:

<https://www.phpmyadmin.net/docs/><sup>3</sup>

<sup>2</sup> <https://www.phpmyadmin.net/>

Stand: 12.11.2018

<sup>3</sup> <https://www.phpmyadmin.net/docs/>

Stand: 12.11.2018

## 3 Komponentenspezifikation

### 3.1 Admin (Extern)

#### 3.1.1 Funktion

Anw.-Fall	Abhängigkeiten
/LF310/	Hierzu soll die externe Schnittstelle/Komponente phpMyAdmin eingerichtet werden.
/LF320/	Hierzu soll die externe Schnittstelle/Komponente phpMyAdmin eingerichtet werden.
/LF330/	Hierzu soll die externe Schnittstelle/Komponente phpMyAdmin eingerichtet werden.
/LF340/	Hierzu soll die externe Schnittstelle/Komponente phpMyAdmin eingerichtet werden.
/LF350/	Hierzu soll die externe Schnittstelle/Komponente phpMyAdmin eingerichtet werden.
/LF360/	Hierzu soll die externe Schnittstelle/Komponente phpMyAdmin eingerichtet werden.

#### 3.1.2 Schnittstellen

Diese Komponente besitzt keine zu spezifizierende Schnittstelle.

#### 3.1.3 Verhalten

Diese Komponente besitzt kein zu spezifizierendes Verhalten.

## 3.2 Autos (GUI)

### 3.2.1 Funktion

Anw.-Fall	Abhängigkeiten
<b>/LF100/</b>	Hierzu soll die "Auto"-Schnittstelle der "AutoService"-Komponente des Clients verwendet werden.
<b>/LF105/</b>	Hierzu soll die "Auto"-Schnittstelle der "AutoService"-Komponente des Clients verwendet werden.
<b>/LF110/</b>	Hierzu soll die "Auto"-Schnittstelle der "AutoService"-Komponente des Clients verwendet werden.
<b>/LF220/</b>	Hierzu soll die "Rating"-Schnittstelle der "RatingService"-Komponente des Clients verwendet werden.

### 3.2.2 Schnittstellen

Diese Komponente besitzt keine zu spezifizierende Schnittstelle.

### 3.2.3 Verhalten

Die *Autos*-Komponente wird unterschieden in *Besucher* und *Benutzer*.

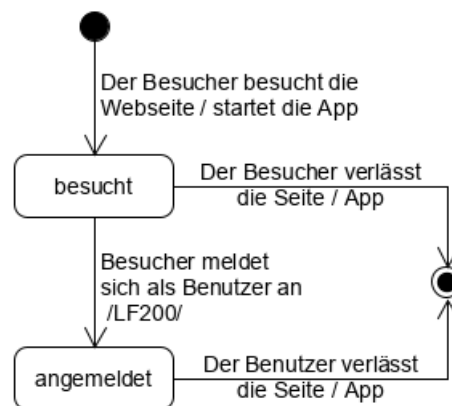


Abbildung 2.1: Zustandsdiagramm: Verhalten für das Login der verschiedenen Akteure

Eine erweiterte Beschreibung zu dem Verhalten der Akteure ist in der Systemspezifikation unter Kapitel 3 zu finden.

### 3.3 Anmelden (GUI)

#### 3.3.1 Funktion

Anw.-Fall	Abhängigkeiten
/LF200/	Hierzu soll die "Authentication"-Schnittstelle der " AuthenticationService "-Komponente des Clients verwendet werden.

#### 3.3.2 Schnittstellen

Diese Komponente besitzt keine zu spezifizierende Schnittstelle.

#### 3.3.3 Verhalten

Diese Komponente soll die Zustände "authentifiziert" und "nicht authentifiziert" besitzen. Nicht authentifizierte Zugriffe sollen vermieden werden. Diese sollen an das Login weitergeleitet werden.

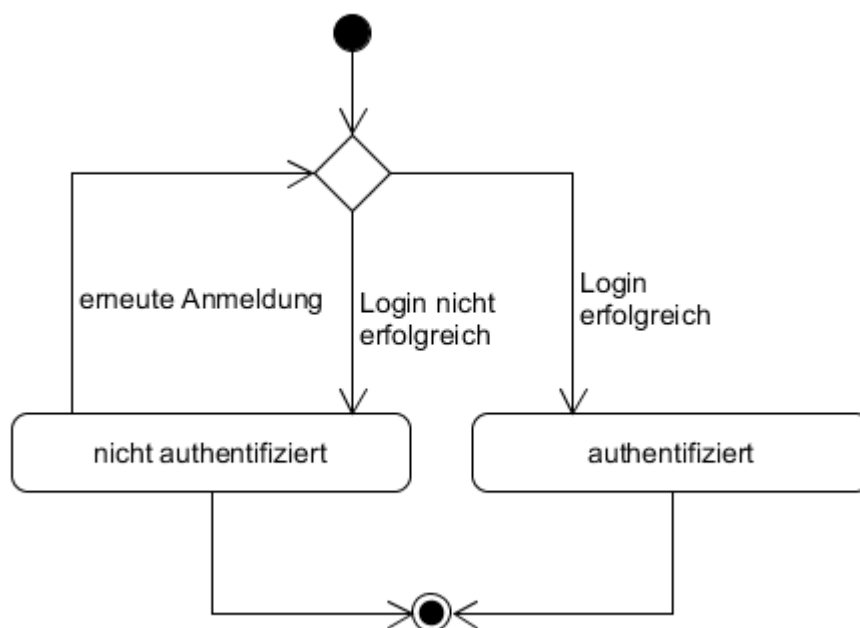


Abbildung 2.2: Zustandsdiagramm eines Logins

### 3.4 Registrieren (GUI)

#### 3.4.1 Funktion

Anw.-Fall	Abhängigkeiten
/LF210/	Hierzu soll die "Authentication"-Schnittstelle der " AuthenticationService "-Komponente des Clients verwendet werden.

#### 3.4.2 Schnittstellen

Diese Komponente besitzt keine zu spezifizierende Schnittstelle.

#### 3.4.3 Verhalten

Diese Komponente besitzt kein zu spezifizierendes Verhalten.



### 3.5 AuthenticationService (Client)

#### 3.5.1 Funktion

Diese Komponente realisiert selbst keine Anwendungsfälle und besitzt auch keine GUI. Sie ist die Schnittstelle zwischen der Server-"Authentication" und Client-"AuthenticationService"-Komponente.

#### 3.5.2 Schnittstellen

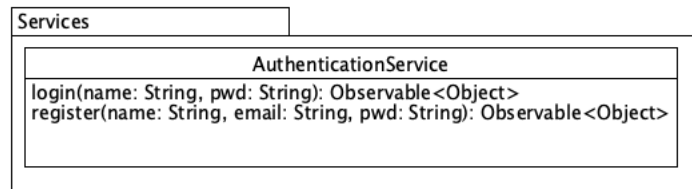


Abbildung 3.1: Implementierung der Schnittstellen-Funktionen von AuthenticationService

Die Funktionen der Komponente "AuthenticationService" in dem Package "Services" bilden Schnittstellen für die GUI-Komponenten: "Registrieren" und "Anmelden".

Aufgrund der Möglichkeit, dass die Funktionen der Komponente Observable (=Beobachtbar) sind, sorgen diese bei Beobachtern für Aktualisiervorgängen falls Änderungen zu erkennen sind.

- login(...)** Die gegebenen Daten werden an den Server geschickt, welcher diese überprüft. Als Ergebnis liefert die Funktion entweder weitere zugehörige Daten von diesem Konto oder eine Fehlermeldung.
- register(...)** Die gegebenen Daten werden an den Server geschickt, seine Antwort entscheidet ob: Entweder ein neues "Benutzer"-Konto angelegt wird und somit dessen Daten zurückgegeben werden oder eine Fehlermeldung zurückgegeben wird, falls die Daten nicht korrekt sind bzw. der Name bereits vergeben ist.

#### 3.5.3 Verhalten

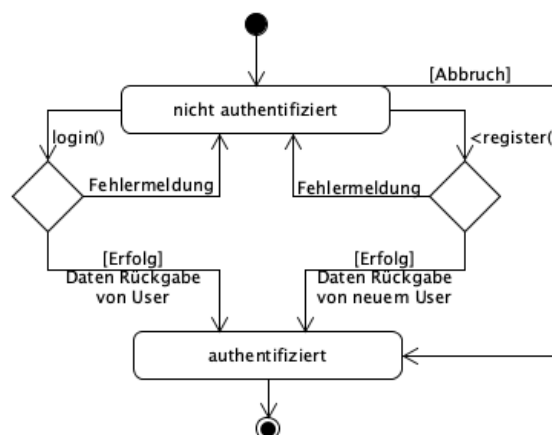


Abbildung 3.2: Zustandsdiagramm der Komponente

Die Komponente "AuthenticationService" soll die Zustände "authentifiziert" und "nicht authentifiziert" verwalten können. Die möglichen Zustandsübergänge sind im Zustandsdiagramm in Abbildung 3.1 dargestellt. Die Zustandsübergänge werden durch die Implementierung der beiden Komponenten "AuthenticationService" (Client) und "Authentication" (Server) realisiert.

## 3.6 Auto (Client)

### 3.6.1 Funktion

Diese Komponente realisiert selbst keine Anwendungsfälle und besitzt auch keine GUI. Sie ist die Schnittstelle zwischen der Server-"Auto"- und Client-"AutoService"-Komponente.

### 3.6.2 Schnittstellen

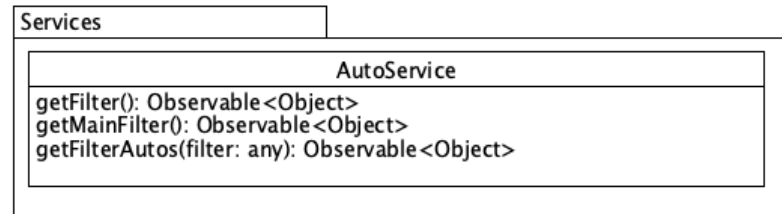


Abbildung 3.3: Implementierung der Schnittstellen-Funktionen von AutoService

Die Funktionen der Komponente "AutoService" in dem Package "Services" bilden Schnittstellen für die GUI-Komponente: "Autos".

Aufgrund der Möglichkeit, dass die Funktionen der Komponente Observable (=Beobachtbar) sind, sorgen diese bei Beobachtern für aktualisier Vorgängen falls Änderungen zu erkennen sind.

- getFilter( )** Liefert eine Liste von Filtern zurück die vom Server angeboten werden oder eine Fehlermeldung falls keine Filter zurück zu geben sind.
- getMainFilter( )** Gibt eine voreingeschränkte Liste von Filtern zurück oder eine Fehlermeldung falls keine Filter zurück zu geben sind.
- getFilterAutos(...)** Die gegebenen Daten werden an den Server geschickt. Als Ergebnis liefert die Funktion entweder eine Liste mit gefundenen Auto-Informationen oder eine Fehlermeldung zurück.

### 3.6.3 Verhalten

Diese Komponente besitzt kein zu spezifizierendes Verhalten.

### 3.7 Rating (Client)

#### 3.7.1 Funktion

Diese Komponente realisiert selbst keine Anwendungsfälle und besitzt auch keine GUI. Sie ist die Schnittstelle zwischen der Server-"Rating" und Client-"RatingService"-Komponente.

#### 3.7.2 Schnittstellen

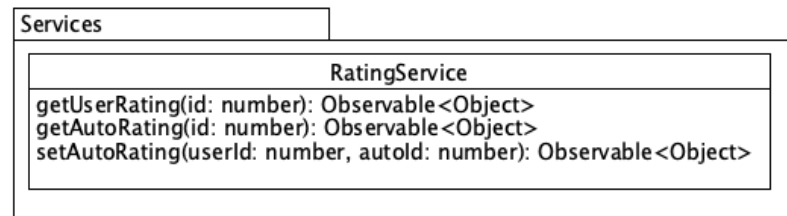


Abbildung 3.4: Implementierung der Schnittstellen-Funktionen von RatingService

Die Funktionen der Komponente "RatingService" in dem Package "Services" bilden Schnittstellen für die GUI-Komponente: "Autos".

Aufgrund der Möglichkeit, dass die Funktionen der Komponente Observable (=Beobachtbar) sind, sorgen diese bei Beobachtern für aktualisier Vorgängen falls Änderungen zu erkennen sind.

**getUserRating(...)** Durch den angegebenen ID-Wert eines Benutzers wird der Server nach allen Bewertungen mit dieser ID angefragt und liefert diese als Liste zurück. Wenn der Benutzer keine Bewertungseinträge hat wird eine Fehlermeldung zurückgegeben.

**getAutoRating(...)** Durch den angegebenen ID-Wert eines Autos wird der Server nach allen Bewertungen mit dieser ID angefragt und liefert diese als Liste zurück. Wenn das Auto keine Bewertungseinträge hat wird eine Fehlermeldung zurückgegeben.

**setAutoRating(...)** Die ID des Autos und des Benutzers in Verbindung mit der Bewertung werden als Parameter übergeben und an den Server weitergeleitet. Als Rückgabe wird ein Objekt mit weiteren Informationen erwartet oder eine Fehlermeldung.

#### 3.7.3 Verhalten

Diese Komponente besitzt kein zu spezifizierendes Verhalten.

### 3.8 Authentication (Server)

#### 3.8.1 Funktion

Diese Komponente realisiert selbst keine Anwendungsfälle und besitzt auch keine GUI. Sie ist eine Unterstützungs-Komponente und stellt den anderen Komponenten Schnittstellen-Operationen für die Umsetzung von Anwendungsfällen zur Verfügung.

#### 3.8.2 Schnittstellen

"*iAuthentication*" bietet eine REST-basierte Schnittstelle für die "Authentication"-Komponente um eine Kommunikation zwischen Client und Server zu ermöglichen. Die Schnittstelle wird von der "AuthenticationService"-Komponente verwendet und bietet Einlogg- bzw. Registrier-Funktionen von der "Authentication"-Komponente an.

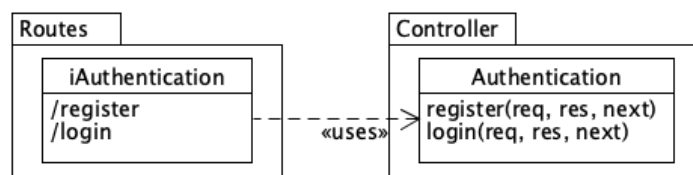


Abbildung 3.5: Implementierung der Schnittstellen von "*iAuthentication*"

**iAuthentication:** Sowohl die ausgehenden- als auch die (vorausgesetzten) eingehenden-Informationen werden als JSON-Datei zwischen Client und Server ausgetauscht.

Möglichkeiten der REST-basierten Erreichbarkeit von

"*iAuthentication*": "/register", "/login"

**/register** Bei erfolgreicher Registration werden die neu erzeugten Benutzerinformationen als Antwort zurückgeschickt, bei einem Fehlschlag der Registration beinhaltet die Antwort stattdessen eine Fehlermeldung.

[Erwartete Daten: Benutzername, E-Mail, Passwort]

**/login** Bei erfolgreichem Anmelden wird ein Session-Token gesetzt und es werden erweiterte Benutzerinformationen als Antwort zurückgeschickt, bei einem Fehlschlag der Anmeldung beinhaltet die Antwort stattdessen eine Fehlermeldung.

[Erwartete Daten: Benutzername, Passwort]

#### 3.8.3 Verhalten

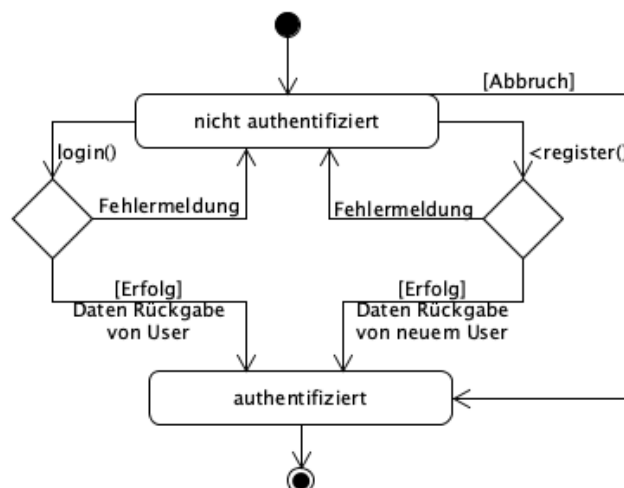


Abbildung 3.6: Zustandsdiagramm der Komponente

Die Komponente „Authentication“ soll die Zustände „authentifiziert“ und „nicht authentifiziert“ erhalten können. Die möglichen Zustandsübergänge sind im Zustandsdiagramm in Abbildung 3.6 dargestellt. Die Zustandsübergänge werden durch die Implementierung der beiden Komponenten "AuthenticationService" (Client) und "Authentication" (Server) realisiert.

### 3.9 Auto (Server)

#### 3.9.1 Funktion

Diese Komponente realisiert selbst keine Anwendungsfälle und besitzt auch keine GUI. Sie ist eine Unterstützungs-Komponente und stellt den anderen Komponenten Schnittstellen-Operationen für die Umsetzung von Anwendungsfällen zur Verfügung.

#### 3.9.2 Schnittstellen

"iAuto" bietet eine REST-basierte Schnittstelle für die "Auto"-Komponente um eine Kommunikation zwischen Client und Server zu ermöglichen. Die Schnittstelle wird von der "AutoService"-Komponente verwendet und bietet die Funktion um eine Selektion von Elektroautos als Liste geliefert zu bekommen.

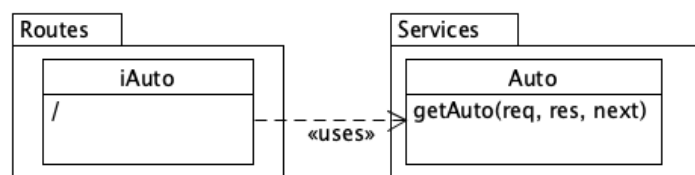


Abbildung 3.7: Implementierung der Schnittstellen von "iAuto"

**iAuto:** Sowohl die ausgehenden- als auch die (vorausgesetzten) eingehenden-Informationen werden als JSON-Datei zwischen Client und Server ausgetauscht.

Möglichkeiten der REST-basierten Erreichbarkeit von

"iAuto": "/"

/ Ruft die Funktion "getAuto(...)" auf und liefert als Antwort eine selektierte Liste von Elektroautos. Die Selektion der Liste ist abhängig von den Filter-Parametern, welche vom Besucher ausgewählt worden sind.

#### 3.9.3 Verhalten

Diese Komponente besitzt kein zu spezifizierendes Verhalten.

### 3.10 Rating (Server)

#### 3.10.1 Funktion

Diese Komponente realisiert selbst keine Anwendungsfälle und besitzt auch keine GUI. Sie ist eine Unterstützungs-Komponente und stellt den anderen Komponenten Schnittstellen-Operationen für die Umsetzung von Anwendungsfällen zur Verfügung.

#### 3.10.2 Schnittstellen

"*iRating*" bietet eine REST-basierte Schnittstelle für die "*Rating*"-Komponente um eine Kommunikation zwischen Client und Server zu ermöglichen. Die Schnittstelle wird von der "*RatingService*"-Komponente verwendet und bietet die Funktionen, um Bewertungen zurück zu liefern oder zu erhalten und zuzuweisen.

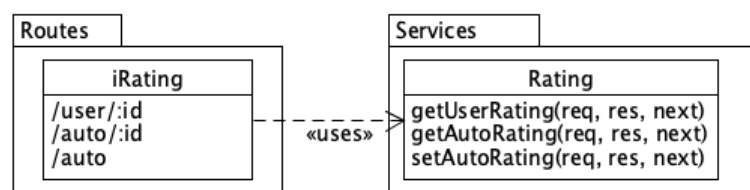


Abbildung 3.8: Implementierung der Schnittstellen von "*iAuto*"

**iAuto:** Sowohl die ausgehenden- als auch die (vorausgesetzten) eingehenden-Informationen werden als JSON-Datei zwischen Client und Server ausgetauscht.

Möglichkeiten der REST-basierten Erreichbarkeit von

"*iRating*": "/user/:id", "/auto/:id", "/auto"

**/user/:id** Ruft die Funktion "*getUserRating(...)*" auf und liefert als Antwort die Liste der Bewertungen eines Benutzers zurück dessen ID von der Client Seite mitgeschickt worden ist.

**/auto/:id** Ruft die Funktion "*getAutoRating(...)*" auf und liefert als Antwort die Liste der Bewertungen eines Elektroautos zurück dessen ID von der Client Seite mitgeschickt worden ist.

**/auto** Ruft die Funktion "*setAutoRating(...)*" auf, wodurch mit den gegebenen Daten eine neue Bewertung in die Datenbank eingetragen wird.

Diese Komponente besitzt kein zu spezifizierendes Verhalten.

## 4 Komponentenverteilung

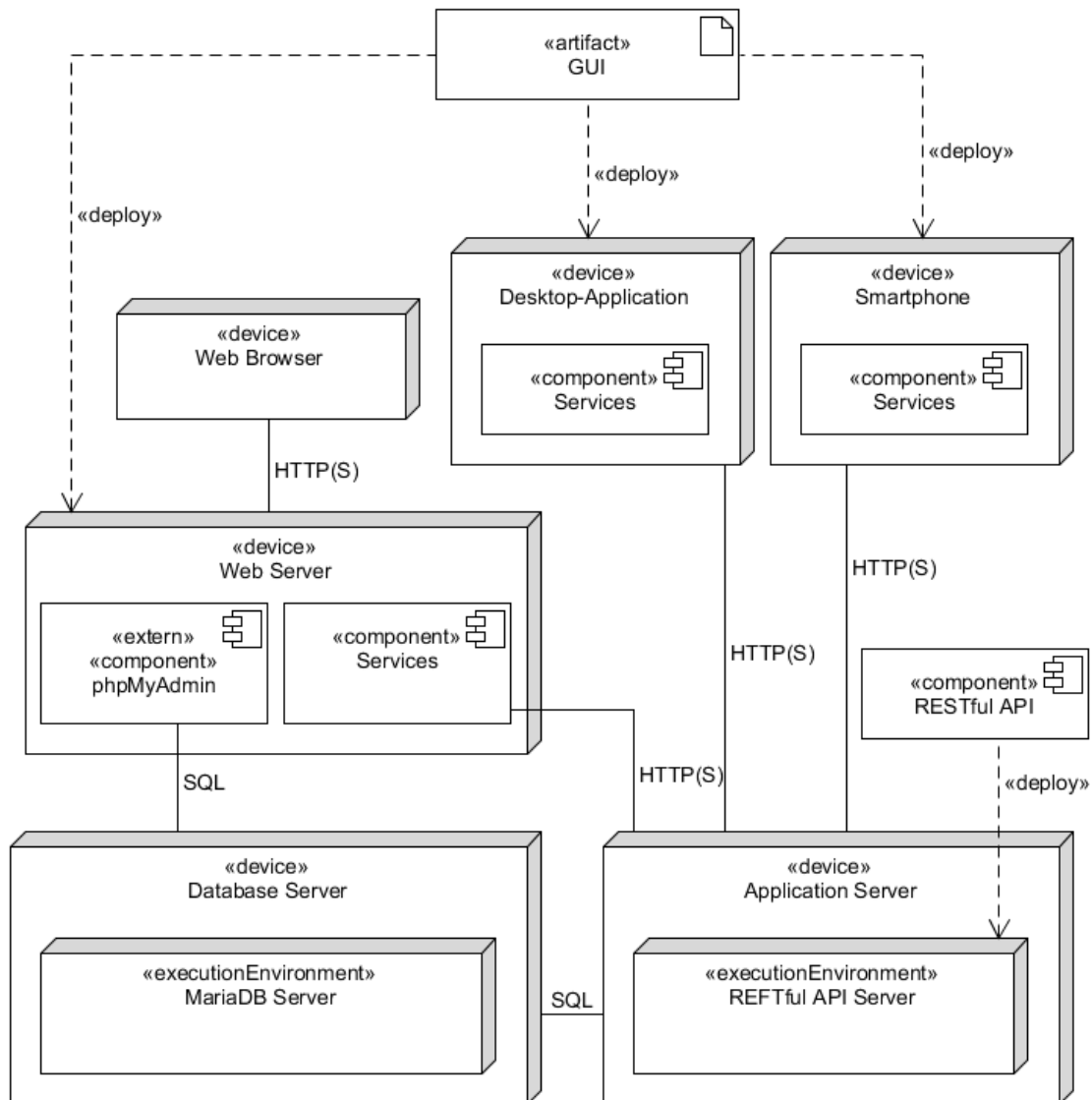


Abbildung 4.1: Verteilungsdiagramm des Systems

Abbildung 4.1 zeigt die Komponentenverteilung des Systems. Das System ist unter drei *Devices* erreichbar. Einerseits in einem beliebigen Web-Browser, als normale Desktop Applikation und zusätzlich noch als Mobile-App.

Der Admin Bereich, welcher nur aus dem Web-Browser erreichbar ist, sowie die Web-Browser Version der Applikation, sprechen einen zusätzlichen Web Service an.

Die eigentliche Logik des Systems, welche vom Web Service, den Desktop- und den Mobile-Applikationen angesprochen wird, liegt als RESTful API Server vor.

Sowohl der RESTful API Server als auch die externe phpMyAdmin Komponente greifen auf den Datenbank Server zu.

Je nach gegebener Hardware und Auslastung bzw. Verwendung der Applikation ist ein Zusammenschluss der *Devices* Web Server, Application Server und Database Server möglich.