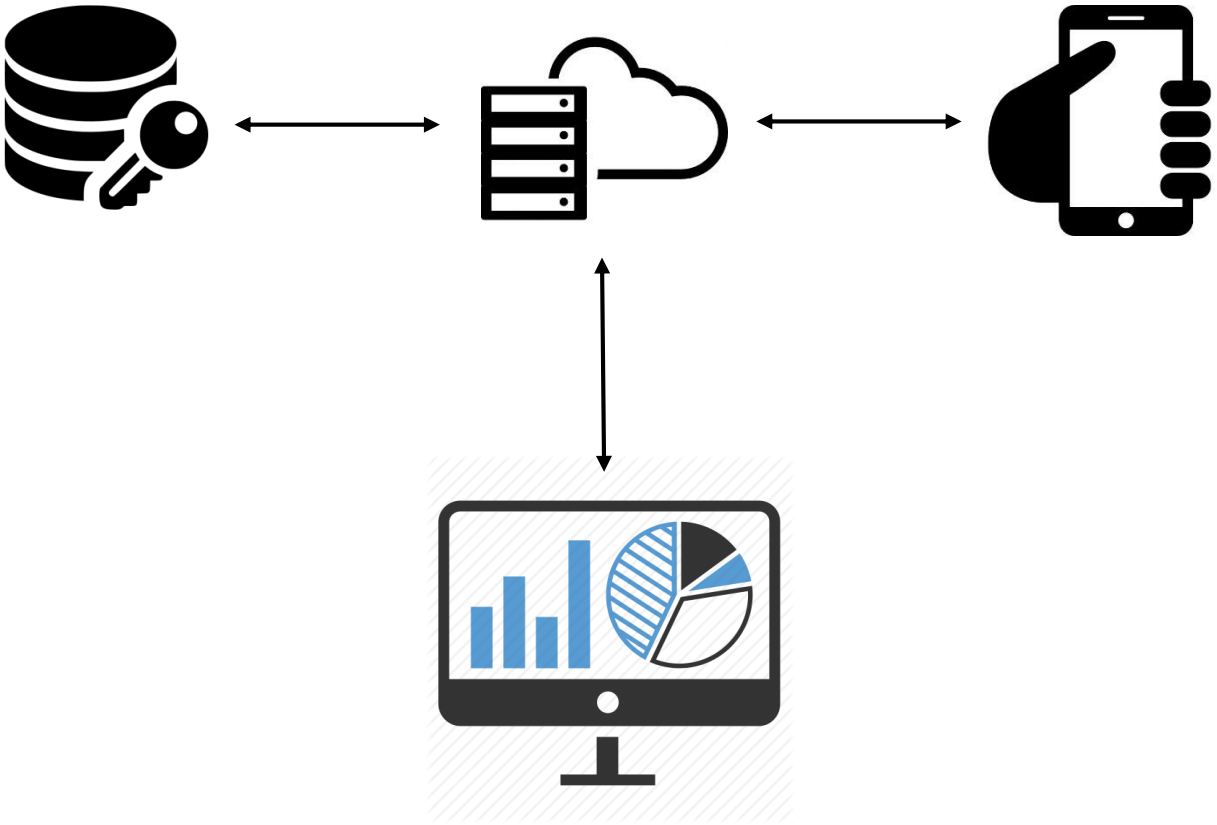
Semesterprojekt Webapplikation
Car Tracker



Danilo Santagata - Rijad Zuzo

Inhaltsverzeichnis

Abbildungsverzeichnis

Projektidee

Es soll eine Webapplikation erstellt werden, die es einem ermöglicht, bestimmte Benutzerinformationen graphisch darzustellen. Die Webapplikation ist für Überwachungszwecke gedacht und somit nur von Administratoren aufrufbar. Gewöhnliche User bekommen diese Ansicht nie zu sehen. Um an die Benutzerinformationen zu gelangen, wird eine Mobileapplikation bereitgestellt. Nur mit der Mobileapplikation können Benutzer Bilder erfassen, posten und in einer Galerie betrachten. Beim Hochladen von Bildern, in unserem Fall Fahrzeugbilder, werden zusätzliche Informationen mitgeliefert. Das können GPS Daten, Zeitpunkt der Aufnahme oder diverse benutzergenerierte Daten sein. Mithilfe dieser Daten, werden in der Webapplikation verschiedene Graphiken erzeugt. Sie bilden Verläufe, wie etwa «Anzahl neuer Uploads» in einem gewünschten Zeitbereich oder «am häufigsten fotografierte Automarke» etc. Um aussagekräftige Graphiken anzuzeigen, werden vorab programmierte Abfragen dargestellt. Eigene Abfragen sind nicht vorgesehen. Nebst den Abbildungen können auch Kartenausschnitte erzeugt werden, welche die Positionen ausgewählter Fahrzeuge anzeigen können. Damit wird ersichtlich, wo die meisten Fotos gepostet werden, bzw. wo die Community am aktivsten ist. Im Hintergrund arbeitet eine Datenbank, die alle erstellten Bilder speichert und verwaltet. Gelöschte Bilder tauchen nicht mehr in der Galerie auf, sind jedoch noch immer in der Datenbank hinterlegt. Das gleiche gilt für die Informationen zum Bild.

Das Projekt gliedert sich in drei Teilbereiche

- Mobileapplikation
  - nimmt mithilfe der Kamera Bilder auf.
  - nebst dem Bild, sendet es weitere Informationen zum Server
  - erstellt eine Galerie, von den erfassten Bildern
- Webapplikation
  - erstellt Datenbezogene Graphiken, die nur für Administratoren bestimmt sind.
  - Kartenausschnitte werden in Zusammenhang mit den Bildinformationen erstellt.
- Backend System
  - verwaltet und legt Bilder ab.
  - erzeugt eine Verbindung zur Datenbank
  - stellt API's zur Verfügung

Beschreibung

Die Webapplikation stellt Informationen, die mit Bildern mitgeliefert werden, grafisch dar. User können, mittels einer Mobileapplikation, Fotos aufnehmen und sie auf den Server laden. Nebst dem Bild, sendet die Mobileapplikation noch weitere Daten, wie etwa die Position, Uhrzeit etc. mit. Die mitgelieferten Daten ermöglichen es, Verläufe oder Hotspots zu zeichnen. So zeigen Graphen an, wie viele Bilder am Tag hochgeladen oder wo die Bilder aufgenommen wurden. Je länger diese Applikation betrieben wird, desto besser werden Muster zum Vorschein kommen. Man findet heraus, wann die User am aktivsten sind oder wo die meisten Fotos aufgenommen werden. Diese Daten sind wichtig für das Marketing oder auch für die Entwickler. Es können auch nach Usern gesucht werden und ausschliesslich dessen Aktivität angezeigt werden.

Die Webapplikation ist ausschliesslich für das Administratorenteam gedacht und kann von gewöhnliche User nicht betreten werden.

Mockup

Mobile Applikation

Abbildung 1 Login Screen

Abbildung 2 Foto bestätigen und uploaden

Bilder können nur im Zusammenhang mit einem gültigen Account hochgeladen werden.

Web Applikation

Abbildung 3 Control Center

Die Kartenansicht zeigt die Positionen der aufgenommenen Bilder. Der Liniengraph zeigt die Anzahl Benutzeruploads in Abhängigkeit der Zeit und das Kuchendiagramm zeigt die Benutzerkategorien. Die Benutzerkategorien definieren sich mit der Anzahl Uploads.

Datenmodell

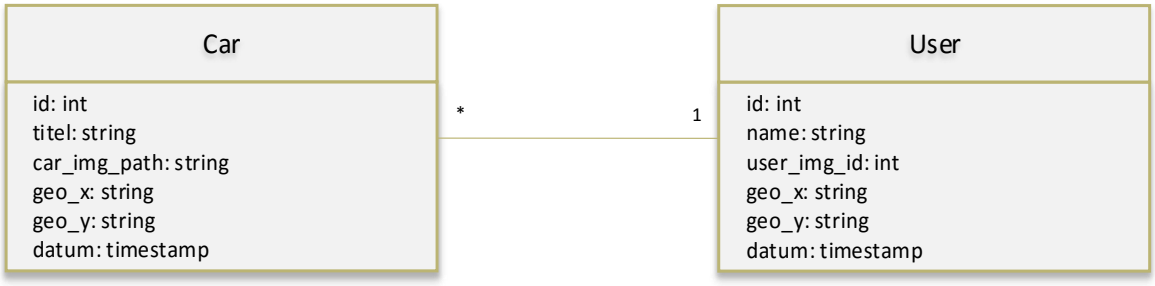


Abbildung 4 Datenmodell Car-Tracker

Die Userprofile bestehen schon auf den Server. Es ist nicht vorgesehen, dass neue Profile erstellt werden können im Umfang von diesem Projekt. Falls noch Zeit übrig ist, wird versucht, diese Funktion zu implementieren. Weiterer Muss- Soll- und Wunschziele sind in den Anforderungen beschrieben.

Einführung Anforderungen

Die Webapplikation «Car Tracker» soll bestimmte Informationen von aufgenommenen Bildern graphisch darstellen können. Es handelt sich um Informationen wie GPS-Position, Zeit der Bilderfassung und Anzahl der bestehenden Bilder eines Benutzers. Diese graphische Informationsdarstellung sollte eine Aktivitätsanalyse vereinfachen und anschaulicher machen.

Anforderungsdefinition

Das Webapplikationsproject unterteilt sich in drei Hauptsystemen. Darunter sind die Mobileapplikation, das Backendsystem und die Webapplikation gemeint. Alle Teilsysteme müssen nebst dem Erfüllen ihrer Aufgabe, noch aufeinander abgestimmt sein. Daraus folgt, dass alle voneinander abhängig sind. Je besser die einzelnen Systeme arbeiten, desto besser funktioniert die komplette Applikation.

Alle Anforderungen werden in funktionale und nichtfunktionale Anforderungen unterteilt. Diese Unterteilung wurde zusätzlich noch nach den Teilsystemen weiter gegliedert, wie in den folgenden Tabellen zu sehen ist.

Funktionale Anforderungen

Backendsystem

ID	Anforderung	Muss	Soll	Wunsch
B1.1	Datenbankverbindung erstellen	X		
B1.2	Datenaustausch mit Datenbank übernehmen (lesen, schreiben)	X		
B1.3	Datenmodelle verwalten über ORM	X		
B1.4	Userdaten verwalten (Bilder, Adressen)	X		
B1.5	Neue Benutzerprofile anlegen		X	
B1.6 *	Erhaltene Koordinaten in Adressen umwandeln		X	

*B1.6 Dies wird benötigt falls die Koordinatenübersetzung nicht bei dem User geschieht.

Mobile Applikation

ID	Anforderung	Muss	Soll	Wunsch
M1.1	Bilder Aufnehmen	X		
M1.2	Verbindung zum Server erstellen	X		
M1.3	Bilder zum Server Hochladen	X		
M1.4	Fotos Abrufen und anzeigen	X		
M1.5	Login zum Benutzerprofile	X		
M1.6	Neues Userprofil erstellen		X	
M1.7	Design anpassen		X	
M1.8	Koordinaten in Adressen umwandeln		X	
M1.9	Benutzerprofile bearbeiten			X

Web Applikation

ID	Anforderung	Muss	Soll	Wunsch
W1.1	Kann nur vom Admin aufgerufen werden.	X		
W1.2	Daten von allen Usern anzeigen	X		
W1.3	Grafik für den Stand der User zahl anzeigen	X		
W1.4*	Grafik, welche Anzahl der Useruploads darstellt, anzeigen.	X		
W1.5	Kartenausschnitt mit Positionen der Aufgenommenen Bilder anzeigen.	X		
W1.6	Benutzeraktivität auf der Karte anzeigen.	X		
W1.7	Einzelinformationen der Benutzer in Grafiken anzeigen		X	
W1.8	Zeitachsen Änderung der Grafiken ermöglichen		X	

*W1.4 Darstellung als Kuchendiagramm mit entsprechenden Kategorien.

Nicht-Funktionale Anforderungen

Backendsystem

ID	Anforderung	Muss	Soll	Wunsch
B2.1	Backups der Benutzerdaten erstellen	X		
B2.2	API Zugriffe absichern gegen Fremde Aufrufe	X		
B2.3	Bei Fehlern Logs erstellen	X		
B2.4	Daten schnell liefern (Antwortzeit < 1.5s)	X		
B2.5	Zuverlässigkeit von 99.999% (Online Verfügbarkeit)	X		

Mobile Applikation

ID	Anforderung	Muss	Soll	Wunsch
M2.1	Lauffähig auf Android Betriebssystem	X		
M2.2	Lauffähig auf IOS Betriebssystem			X
M2.3	Verbindung mit Server bei Transaktionen aufrecht halten	X		
M2.4	Daten fehlerfrei dem Server schicken	X		
M2.5	Applikation mit „Material Design“ erstellen		X	

Web Applikation

ID	Anforderung	Muss	Soll	Wunsch
W2.1	Intuitive Bedienung anbieten	X		
W2.2	Ergebnisse der Auswertungen fehlerfrei anzeigen	X		
W2.3	Daten nur dem Administrator Profil ausgeben	X		
W2.4	Applikation mit „Material Design“ erstellen		X	
W2.5	Mehrsprachige Layouts bereitstellen			X

Mockups

Mobileapplikation

Abbildung 7: Login Screen (Mobileapp)

Abbildung 5: Bild Aufnahme (Mobileapp)

Abbildung 6: Gallery Screen (Mobileapp)

Webapplikation

Abbildung 8: Login Screen (Webapplikation)

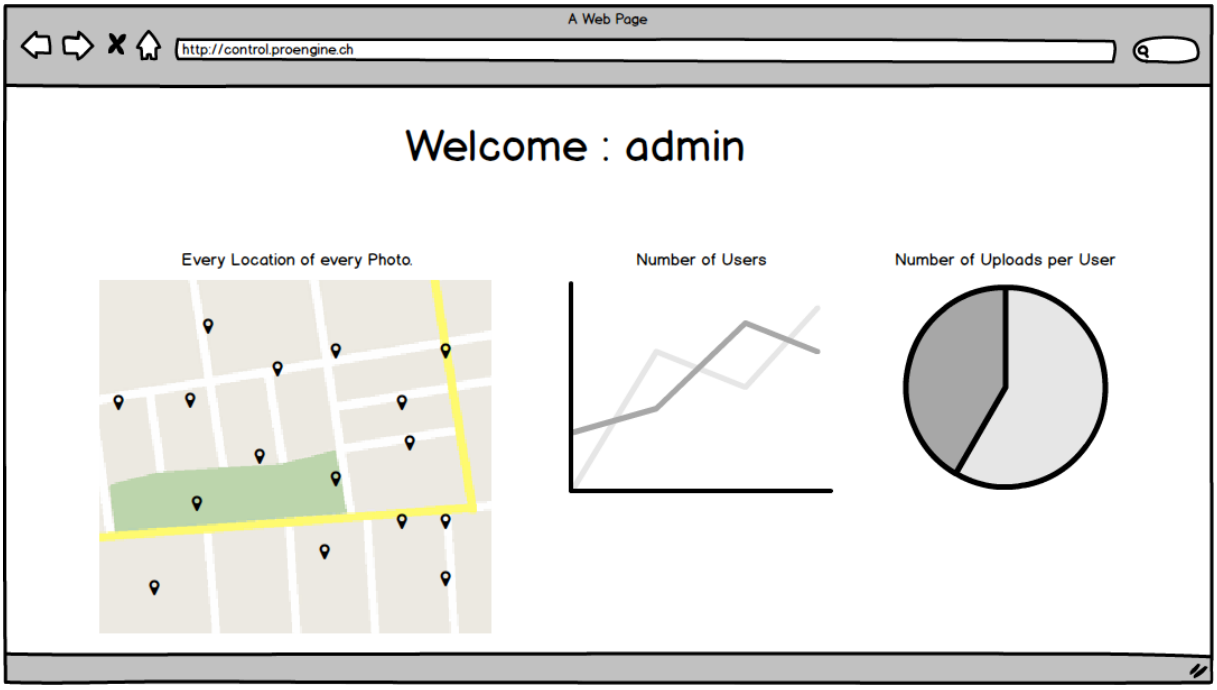


Abbildung 9: Admin Panel (Webapplikation)

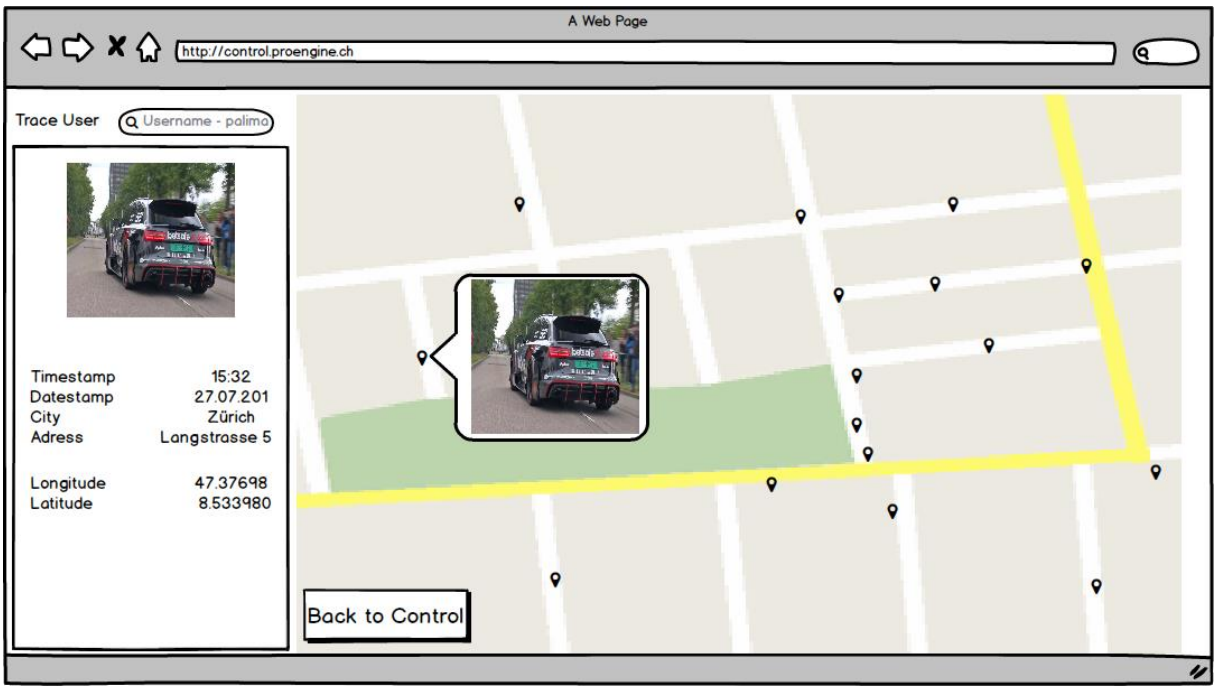


Abbildung 10: Map Screen (Webapplikation) - Detailansicht

Datenmodell

Backend

Klassendiagram

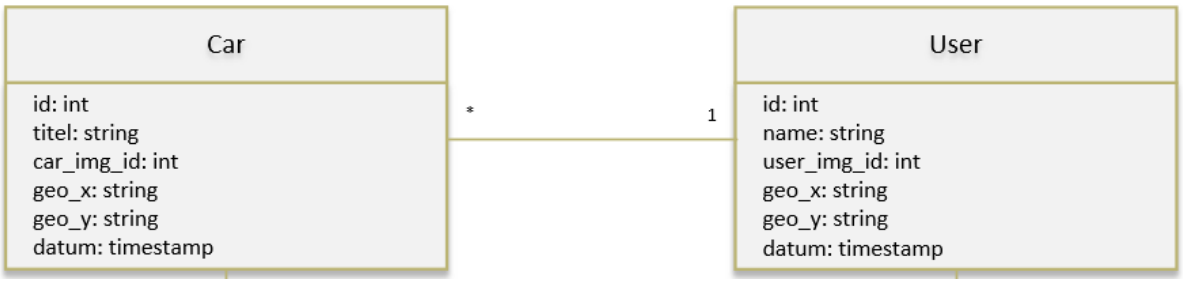


Abbildung 11: Datenmodell Backend

Frontend

Controllerdiagram

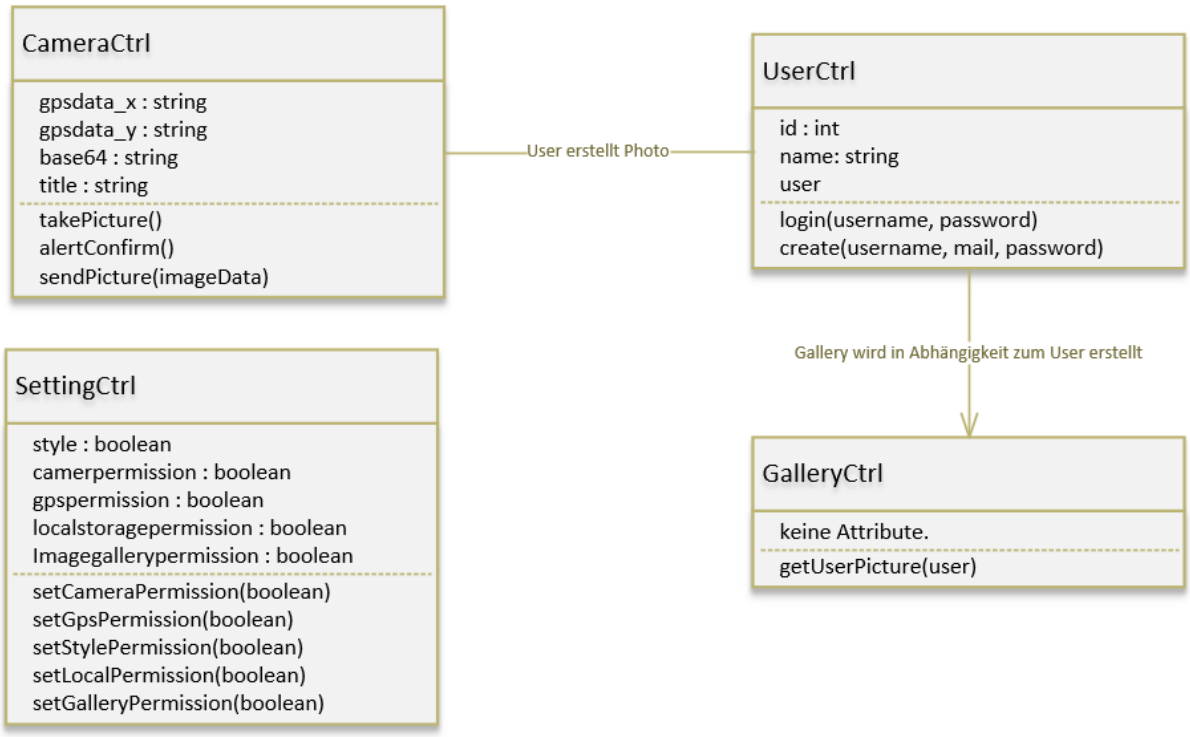


Abbildung 12: Datenmodell Mobileapplication

Architektur

Die Architektur des Projektes besteht aus einem Backend, welcher Daten per RESTful Schnittstellen liefert, und aus mehreren Frontends die diese Daten verarbeiten. Das Backend wird mittels PHP und dem MVC Framework Laravel aufgebaut. Es stellt Routen (URL's) zur Verfügung, über welche das Frontend auf Daten zugreifen und diese ändern kann.

Das Frontend wurde in zwei Bereiche unterteilt, in ein Web – Frontend und ein Mobile – Frontend. Beide werden einheitlich entwickelt, deshalb benutzt man, wenn immer möglich, die gleichen Frameworks. Für das Projekt wurden konkret die Frameworks «AngularJS» verwendet. Das Mobile-Frontend benutzt noch weitere Frameworks, auf die später eingegangen wird.

Das Design des Fronteds wird mittels Material Design Vorschriften implementiert, mit Hilfe von zusätzlichen Plug-Ins wie Angular-Material.

Überblick

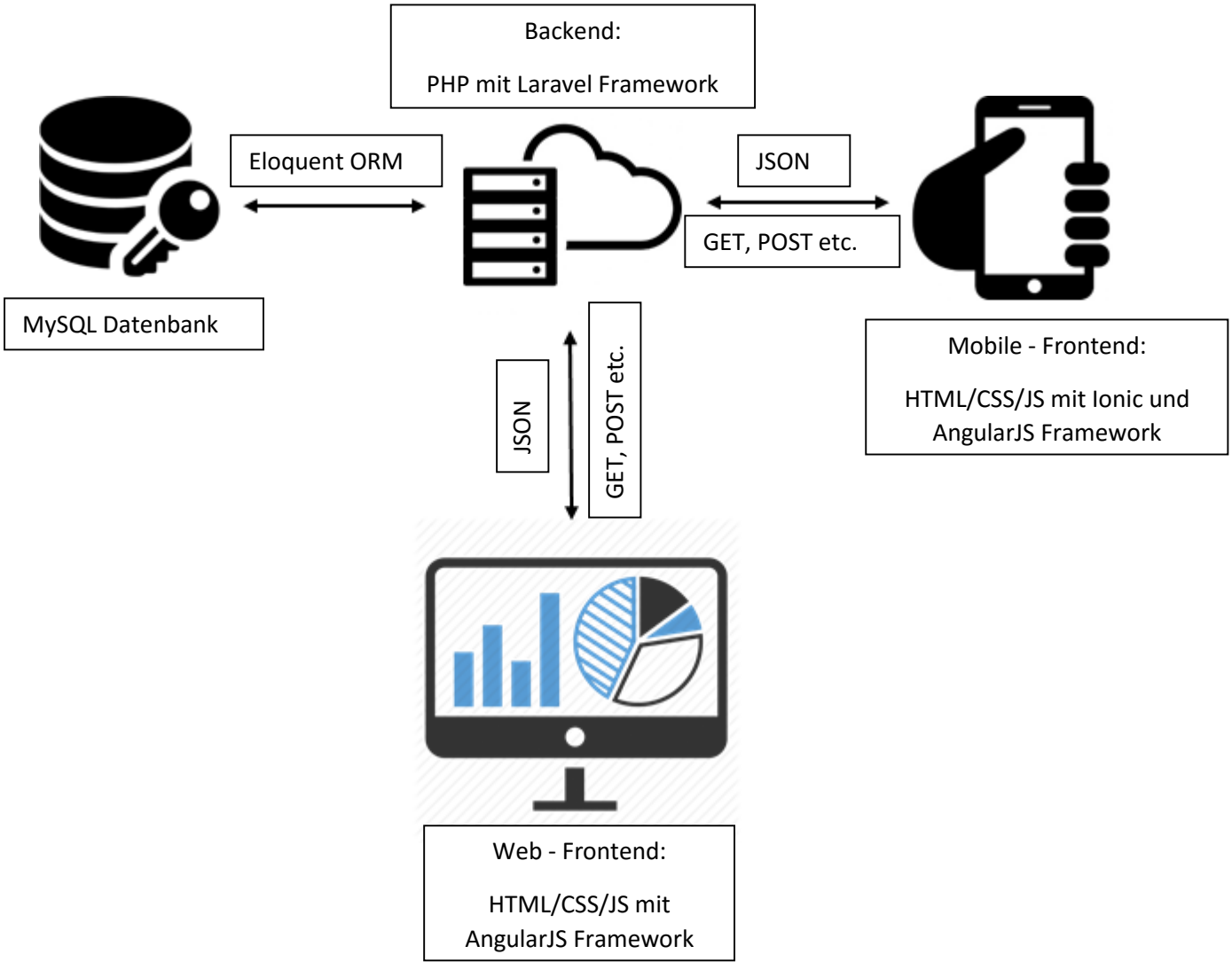


Abbildung 13: Graphischer Überblick über System und Schnittstellen.

Backend

Das Backend wird mittels PHP und dem Framework Laravel erstellt. Durch dieses MVC Framework wird vieles vereinfacht und Gefahren, die durch die dynamische Programmierung in PHP entstehen, minimiert.

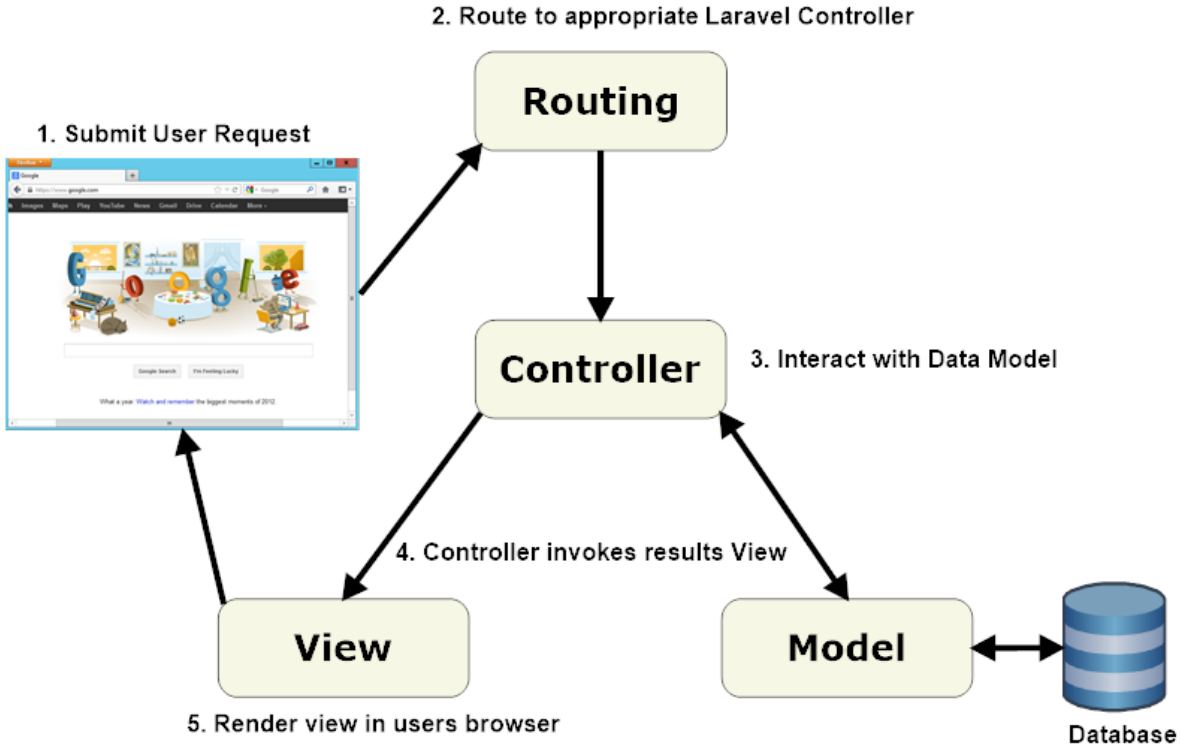


Abbildung 14: MVC Struktur von Laravel

Es werden „Modelle“ und ihre Relationen definiert, welche in der Datenbank mithilfe des Eloquent ORM festgehalten werden. Für diese Modelle werden im „Controller“ Funktionen definiert, wie z.B. das Speichern, die Aktualisierung und der Löschvorgang.

Der „View“ Teil des Frameworks wurde in diesem Projekt nicht benützt, da das Backend als ein simples Web Service dienen soll.

Die Authentifizierung, für das benutzen der Routen, wurde mittels speziellen Token abgesichert. Diese gibt der Server aus, falls der Benutzer sich mit richtigen Benutzernamen und Passwort anmeldet. Der Token wird bei jedem Request (GET, POST, PUT, DELETE) mitgeschickt damit der Benutzer auch nachweisst das er authentifiziert wurde. Er beinhaltet alle Userinformationen in verschlüsselter Form (RSA).

Durch solche Art von Authentifizierung werden automatisch Angriffe wie CSRF (Cross-site request forgery) unterbunden da es keine States gibt, die sich ausnutzen lassen.

Frontend

Das Frontend wird für zwei Benutzer-Gruppen erstellt. Für normale Benutzer ist nur das Mobile – Frontend nutzbar, andererseits können Admin-Benutzer sowohl das Mobile-Frontend wie auch die Webapplikation verwenden.

Web

Das Web – Frontend wird gemeinsam mit dem Backend gehostet und ist teilweise eingebettet in das Framework Laravel.

Es wird als SPA (Single Page Application) mit dem Framework Angular implementiert. Das heisst, dass die eigentliche Website nie ändert. Der Browser verlässt während der Session nie die URL. Es werden nur per AJAX-Aufrufe neue Daten oder sogar komplette HTML Seiten geladen und diese dann in die Website eingefügt.

Durch das teilweise Einbetten des Frontends in das Backend ergibt sich eine Situation wo sich die zwei Frameworks Routen teilen. Dies ist wie folgt dargestellt:

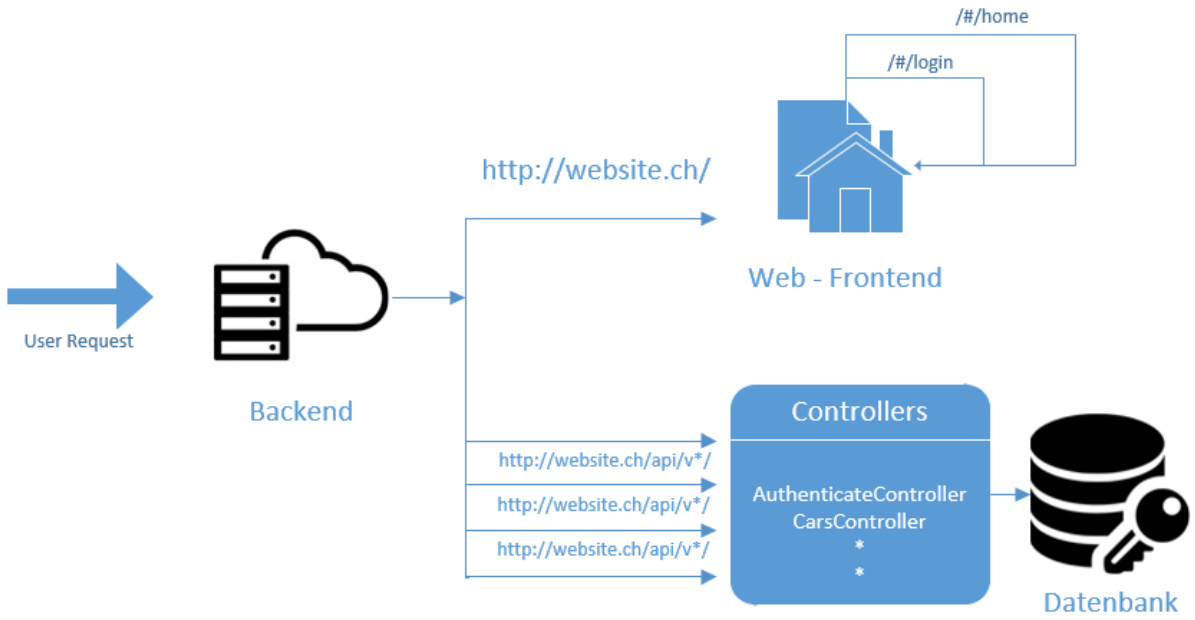


Abbildung 15: Routen Teilung der Frameworks

Bei einem Bentuzeraufruf der Hauptwebseite, wie z.B. Index.html, überlässt Laravel das routen an AngularJS. Darum sehen wir in der URL das muster << http://website/#/login >>. Das Symbol „#“ kennzeichnet die Übernahme des Routings durch AngularJS. Die API Routen werden weiterhin von Laravel normal verarbeitet.

Autorisierung und Authentifizierung der AngularJS-Routen wird durch das Plug-In «Angular-Permission» in Verbindung mit dem Plug-In «Satellizer» erreicht. «Angular-Permission» ermöglicht es, gewisse Routen zu blocken solange man nicht die richtige Authentifizierung hat, welche beim erfolgreichen Login, durch Satellizer automatisiert eingetragen wird.

Mobile

Die Mobileapplikation ist eine typische hybride Applikation. Sie wird mithilfe des Ionic Framework zur ausführbaren Installationsdatei für die gängigen mobilen Betriebssysteme wie Android oder iOS. Erstellt wird die Applikation klassisch mit HTML/CSS/JS. Um das MVC-Modell durchzusetzen wurde auf das AngularJS Framework zurückgegriffen. Damit eine hybride Applikation auf die Hardwareressourcen, wie etwa die Kamera oder die GPS-Ortung, zugreifen kann, bedarf es dem ngCordova Framework. Dieses macht es für den JS-Entwickler einfach, jegliche Ressourcen abzufragen, ohne dass sich der Entwickler Gedanken über die verbaute Hardware machen muss. Das untere Bild veranschaulicht die Arbeitsweise des AngularJS Frameworks und die Beziehungen zwischen den verschiedenen Framework Modulen.

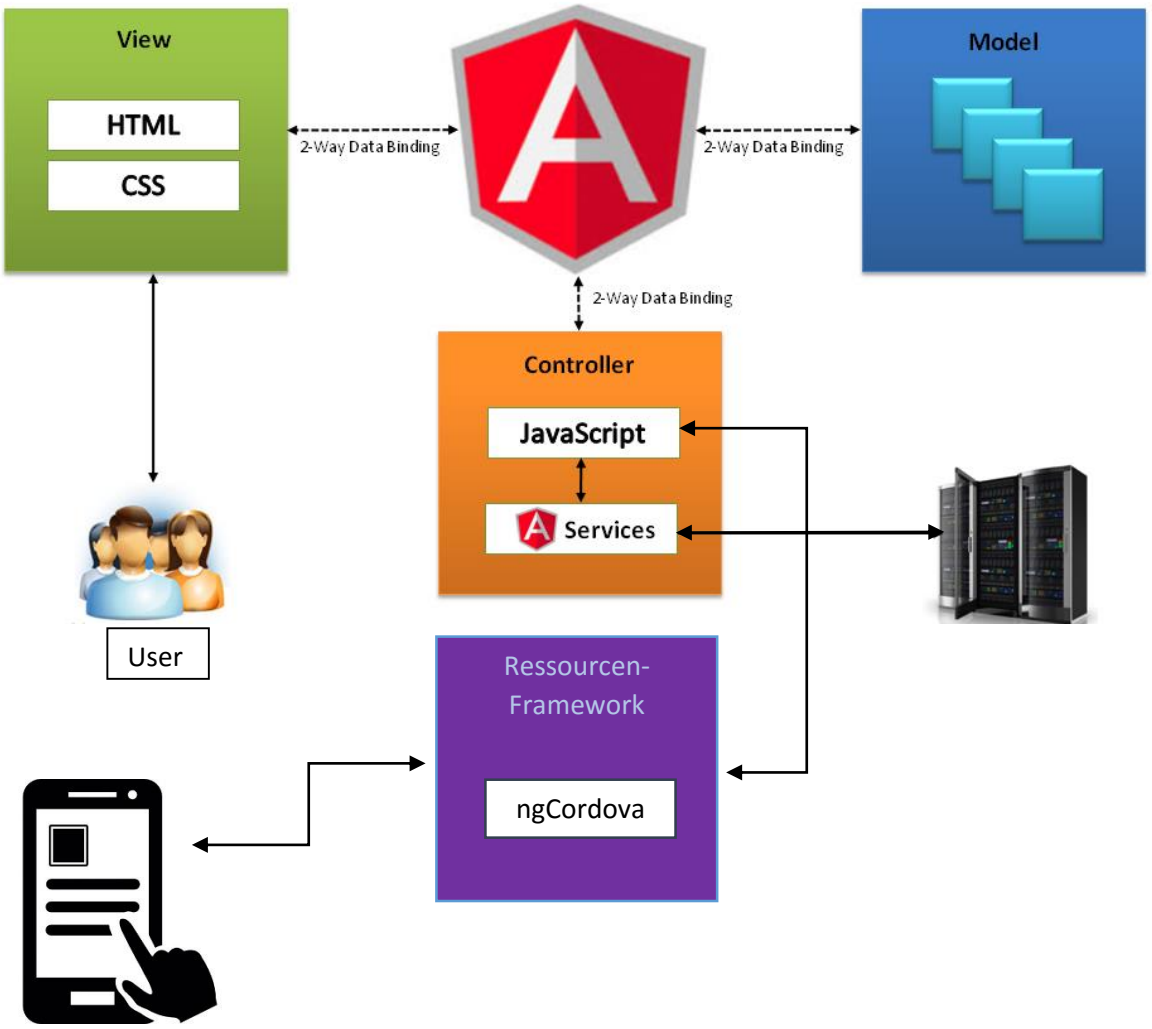


Abbildung 16: MVC-Modell von AngularJS mit ngCordova

Die Applikation kommuniziert mit dem Server per RESTful Schnittstellen. Abgängig von den verschiedenen URL's werden Daten gesendet oder angefordert. Ein klassischer Ablauf ist der Folgende:

1. User schiesst ein Foto eines Autos und füllt die erforderlichen Textfelder aus.
2. Mit dem Button «senden» schickt er ein JSON Objekt von der Mobileapp zum Server.
3. Der Server nimmt die Daten entgegen und legt sie userabhängig ab.
4. User fordert seine Daten an -> Server schickt JSON Objekt mit verschiedensten Autos und dessen Attributen.

Alle Abfragen werden mit http Request realisiert. Je nachdem, welcher Controller die Requests abschickt, werden die Daten in den jeweiligen View's angezeigt.

Die Hauptaufgabe von Ionic ist es, aus einer Webapp eine Mobileapp zu generieren. Dafür wird die Webapp in ein Android- oder IOS-Container gepackt. Somit «gaukelt» man dem OS vor, eine Native-App zu betreiben. Die App selber läuft nur in einem Webview.

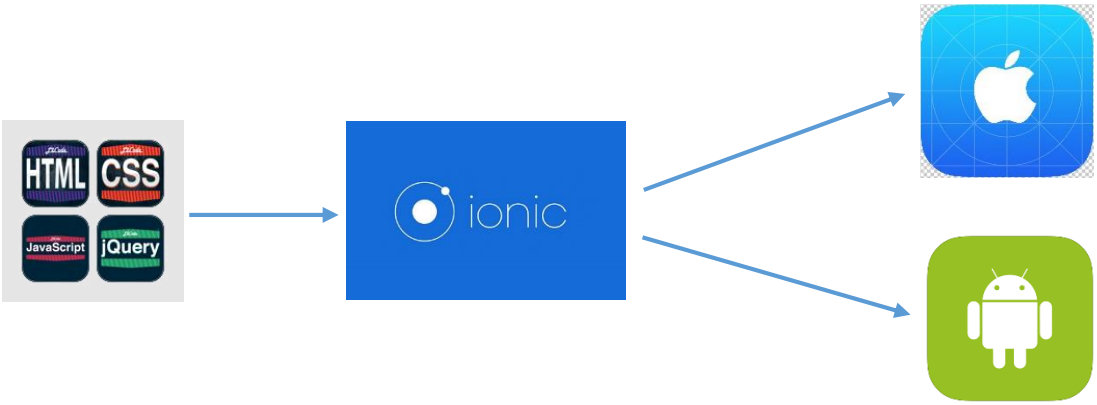


Abbildung 17: Generierungsprozess von der Webapp zur "Native App"

Damit das Frontend der Mobileapplikation und das der Webapplikation übereinstimmen und eine gewisse Wiedererkennbarkeit entsteht, hat man sich auf das «Angular Material» Framework entschieden. Dieses regelt einzig und alleine den Style der verschiedenen Komponenten.

Quellenverzeichnis

Abbildung 14 : [http://laravelbook.com/laravel-architecture]

Abbildung 16: [https://avaldes.com/angularjs-introduction-and-sample-programs]

Abbildung 17: [

https://cdn1.iconfinder.com/data/icons/ios-7-style-metro-ui-icons/512/MetroUI_OS_Android.png,

https://appicontemplate.com/wp-content/uploads/2015/07/ios7_full@2x.jpg,

http://blog.teamtreehouse.com/wp-content/uploads/2015/05/ionic.jpg,

http://a3.mzstatic.com/us/r30/Purple69/v4/40/3b/58/403b585d-f46f-589f-bc0a-5507d05569e3/icon175x175.png

]