HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

EcoHelper - Android

Bachelorarbeit

Abteilung Informatik

Hochschule für Technik Rapperswil

Frühjahrssemester 2011

Autoren:	Raphael Nagy
	Stefan Schöb
	Reto Zigerlig
Betreuer:	Prof. Dr. Peter Heinzmann
Projektpartner:	cnlab AG Rapperswil und Projekt Young Engineers
Experte:	Dr. Thomas Siegenthaler
Gegenleser:	Prof. Dr. Andreas Steffen

Aufgabenstellung

Abteilung:	Informatik
Name der Studierenden:	Raphael Nagy, Stefan Schöb, Reto Zigerlig
Studienjahr:	Frühjahrssemester 2011
Titel der Studienarbeit:	EcoHelper – Android Verbrauchsanalyse und -Optimierung beim Autofahren
Examinator:	Prof. Dr. P. Heinzmann
Experte / Korreferent:	Dr. Th. Siegenthaler, CSI Consulting AG / NA
Themengebiet:	Internet-Technologien und -Anwendungen
Projektpartner:	cnlab AG Rapperswil und Projekt Young Engineers

Im Rahmen des EcoHelper-Projekts werden seit 2006 Zusammenhänge zwischen Benzinverbrauch und Fahrweise im Alltagsverkehr untersucht. Mittelfristig sollen die Autofahrer über eine Smartphone-Anwendung Hinweise zur Verbesserung ihrer Fahrweise erhalten. Bereits entwickelt wurden Systeme auf Symbian und iPhone-Plattformen. Diese erfassen die Fahrparameter Position, Geschwindigkeit und Beschleunigung über die im Smartphone integrierten Sensoren. Über Onboard Diagnosebus (OBD) Adapter sammelt man auch Motorenparameter (z.B. Drehzahl, Gaspedalstellung Verbrauch). Die Resultate der Testfahrten können auf www.tourlive.ch und www.tourlive.ch/ecohelpertest/ analysiert werden.

Im Rahmen der vorliegenden Bachelor-Arbeit soll nun eine Anwendung für die Android-Plattform realisiert und in einen „Friendly User Test“ überführt werden. Die Android-Anwendung soll die Grundfunktionalität der bestehenden Symbian-Anwendung sowie erweiterte Funktionen anbieten. Erweiterungen betreffen vor allem folgende Bereiche:

- Fahrer und Trip Administration
- Anwendungsspezifische Darstellung der Messwerte (WAB2-Fahrtrainings, Privatfahrten)
- Darstellung von Trips auf dem Smartphone
- Offline Mode mit WLAN-Upload-Funktion
- Hinweise zur Verbesserung der Fahrweise

Ferner soll die EcoHelper-Android-Anwendung auch ein „elektronisches Tankbüchlein“ anbieten. Die Funktionalität dieser Tankbüchlein-Ergänzung soll derjenigen von Webanwendungen wie beispielsweise http://www.verbrauchsrechner.de entsprechen.

Diese Arbeit soll auch im Rahmen des Young Engineers Projekt[1] präsentiert werden.

[1] http://www.wec2011-youngengineers.ch/

Urheberschaftserklärung

Die vorliegende Arbeit basiert auf Ideen, Arbeitsleistungen, Hilfestellungen und Beiträgen gemäss folgender Aufstellung:

Gegenstand, Leistung	Person	Funktion
GUI, Datenübertragung, Synchronisation, elektronisches Tankbüchlein	Raphael Nagy	Autor der Arbeit
GUI, Datenaufzeichnung	Stefan Schöb	Autor der Arbeit
GUI, Kamera, OBD-Informationen	Reto Zigerlig	Autor der Arbeit
Korrekturlesen, Hinweise zur Strukturierung und sprachlichen Überarbeitung	Prof. Dr. P. Heinzmann	Verantwortlicher Professor
Idee, Aufgabenstellung, allgemeines Pflichtenheft, Betreuung während der Arbeit	Prof. Dr. P. Heinzmann	Verantwortlicher Professor
Ansprechpartner bei Fragen zu früheren Projekten und Webserver	O. Afshari	HSR Assistent
Entwicklung und zu Verfügung Stellung TourLive Prototyp (Symbian Version) , Betreuung während der Arbeit	cnlab AG	Industriepartner

Tabelle 1: Urheberschaftserklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit gemäss obiger Zusammenstellung selber und ohne weitere fremde Hilfe durchgeführt haben,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

R. Nagy

Rapperswil, den..... .....

S. Schöb

Rapperswil, den..... .....

R. Zigerlig

Rapperswil, den..... .....

Vereinbarung

Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der *Bachelorarbeit „EcoHelper - Android"* von *Raphael Nagy, Stefan Schöb und Reto Zigerlig* unter der Betreuung von Prof. Dr. P. Heinzmann (für die Arbeit verantwortlicher Professor) geregelt.

Urheberrecht

Die Urheberrechte der im Rahmen der Arbeit neu entwickelten und dokumentierten Teile stehen der Studentin / dem Student zu. Die Urheberrechte zu den zu Verfügung gestellten EcoHelper-Grundsystemen stehen cnlab AG (R. Vogt und P. Heinzmann) zu.

Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von allen an der Arbeit beteiligten Parteien, d.h. von den Studenten, welche die Arbeit verfasst haben, vom verantwortlichen Professor sowie vom Industriepartner verwendet und weiter entwickelt werden. Die Namensnennung der beteiligten Parteien ist bei der Weiterverwendung erwünscht.

Rapperswil, den..... .....

Rapperswil, den..... .....

Rapperswil, den..... .....

Rapperswil, den..... .....

Verantwortlicher Professor

Rapperswil, den..... .....

Industriepartner

Inhaltsverzeichnis

1. Abstract

Abteilung	**Informatik**
Namen der Studierenden	**Raphael Nagy, Stefan Schöb, Reto Zigerlig**
Studienjahr	**FS 2011**
Titel der Studienarbeit	**Verbrauchsanalyse und –Optimierung beim Autofahren (EcoHelper)**
Examinator, Betreuer	**Prof. Dr. P. Heinzmann, O. Afshari**
Themengebiet	**Internet-Technologien und -Anwendungen**
Projektpartner	**cnlab AG Rapperswil und Projekt Young Engineers**
Institut	**Institut für Internet-Technologien und -Anwendungen**

An der HSR und bei der Firma cnlab untersucht man seit 2006 Zusammenhänge zwischen Benzinverbrauch und Fahrweise. Die in diesem Zusammenhang entwickelten EcoHelper Smartphone-Anwendungen erfassen GPS- und Beschleunigungsdaten (z.B. Geschwindigkeit, Position), sowie über den Onboard- Diagnose- Bus (OBD2) auch Fahrzeugdaten (z.B. Benzinverbrauch, Drehzahl oder Gaspedalstellung). Diese Daten werden via Mobilfunk oder WLAN zur Visualisierung und Analyse zum EcoHelper-Server übertragen. Eine solche Anwendung gibt es bereits für die Plattformen iPhone, Symbian und Windows Mobile 5.0, neu sollte EcoHelper für Android realisiert werden.

Nach der Analyse der Vorarbeiten und dem Studium ähnlicher Smartphone-Anwendungen wurde der Funktionsumfang einer neuen EcoHelper-Anwendung festgelegt. Diese neu spezifizierte Anwendung wurde mit Hilfe des Android SDK (in Java) als "EcoHelper- Android" realisiert. Die Kommunikation zur OBD2 Schnittstelle findet via Bluetooth statt. Als Datenübertragungsformat zum Webserver wurde JavaScript Object Notation (JSON) verwendet.

Die realisierte EcoHelper-Android Anwendung bietet konfigurierbare Anzeigen von Fahrzeug- und Fahrdaten gemäss Anforderungen von Fahrlehrern und Privatpersonen. Es gibt neue Funktionen zur Verwaltung von Trips, Fahrern und Fahrzeugen, sowie zum einfacheren Anschluss des OBD2-Adapters. Neu ist ein "elektronisches Tankbüchlein" integriert und die Erkennung des aktuell eingelegten Gangs wurde verbessert. EcoHelper-Android unterstützt alle Möglichkeiten des ebenfalls neu entwickelten EcoHelper-Servers. Die Funktionstüchtigkeit des neuen Systems wurde durch zahlreichen Testfahrten mit verschiedenen Fahrzeugen belegt. Nach Abschluss der Bachelorarbeit wird die A- Z Verkehrsschule Ostschweiz AG EcoHelper-Android in der 2-Phasen Ausbildung einsetzen.

2. Management Summary

2.1. Ausgangslage

Die CO_2-Belastung, sowie die ständig steigenden Benzinpreise sind globaler Natur und wirken sich spürbar auf unseren Alltag aus. Dies fordert uns alle immer mehr heraus, einen ökologisch nachhaltigen Lebensstil zu führen. In diesem Zusammenhang werden an der HSR und bei der Firma cnlab seit 2006 Zusammenhänge zwischen Benzinverbrauch und Fahrweise im Alltagsverkehr untersucht. Die daraus entwickelten EcoHelper Smartphone-Anwendungen erfassen Ortungs- und Beschleunigungsdaten (z.B. Geschwindigkeit, Position), sowie über den Onboard- Diagnose- Bus (OBD2) auch Fahrzeugdaten (z.B. Benzinverbrauch, Drehzahl oder Gaspedalstellung). Diese Daten werden via Mobilfunk oder WLAN zur Visualisierung und Analyse zum EcoHelper-Server übertragen. Die Anwendung gibt es bereits für die Plattformen iPhone, Symbian und Windows Mobile 5.0, neu sollte eine verbesserte EcoHelper-Anwendung für die weitverbreitete und sehr populäre Android-Plattform realisiert werden.

2.2. Vorgehen

Das EcoHelper-Projekt wurde nach einer angepassten Version des Rational Unified Process (RUP) entwickelt. Dabei wurden nicht benötigte Elemente weggelassen, die Grobstruktur jedoch beibehalten. Zu Beginn des Projektes wurden in der ausführlichen Analyse-Phase die bestehenden Arbeiten, sowie die Prinzipien zum Thema EcoHelper untersucht. Zusätzlich wurden ähnliche kommerzielle Smartphone-Anwendungen studiert. Daraus wurde der Funktionsumfang der neuen EcoHelper-Anwendung festgelegt und die Funktionen in drei Prioritäts-Klassen unterteilt. Wobei zuerst die Funktionen der ersten Prioritäts-Klasse implementiert wurden.

Die neu spezifizierte Anwendung wurde mit Hilfe des Android SDK (in Java) als „EcoHelper- Android“ realisiert. Als Entwicklungsumgebung wurde Eclipse eingesetzt. Die Kommunikation zur OBD2 Schnittstelle findet via Bluetooth statt. Als Datenübertragungsformat zum Webserver wurde JavaScript Object Notation (JSON) verwendet. Die Entwicklung der Server-Anwendung wurde durch cnlab/HSR durchgeführt.

Der Industriepartner dieser Bachelor-Arbeit war die Firma cnlab AG aus Rapperswil SG. Betreut wurde das Projekt durch Prof. Dr. P. Heinzmann. In wöchentlichen Meetings mit Herr Heinzmann und dem HSR Assistenten Omid Afshari, wurden jeweils der Projektstand, anstehende Probleme und der weitere Projektverlauf besprochen. Während der Entwicklung fand eine intensive Kommunikation zwischen dem EcoHelper-Team und Omid Afshari statt.

Dem Gegenleser wurde der Projektstand nach halber Projektzeit in Anwesenheit von Betreuer, Assistent und einem cnlab Mitarbeiter präsentiert. So konnten die gegenseitigen Erwartungen geklärt und abgestimmt werden, was sicher zum Gelingen des Projekts beigetragen hat.

2.3. Ergebnis

Im folgenden Diagramm wird die Systemübersicht dargestellt.

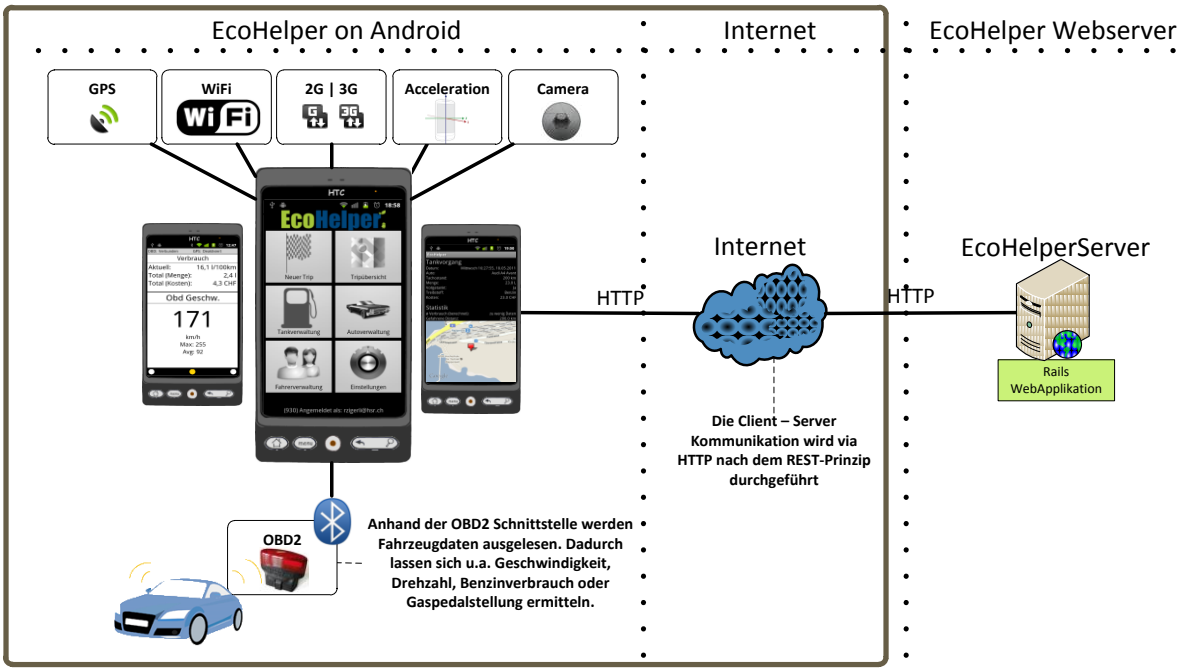


Abbildung 1: Systemübersicht EcoHelper für Android

Die realisierte EcoHelper-Android Anwendung bietet konfigurierbare Anzeigen von Fahrzeug- und Fahrdaten gemäss Anforderungen von Fahrlehrern und Privatpersonen. Dies ermöglicht dem Benutzer, die Daten seinen Bedürfnissen entsprechend anzuzeigen. Ebenfalls gibt es neue Funktionen zur Verwaltung von Trips, Fahrern und Fahrzeugen, sowie zum einfacheren Anschluss des OBD2- Adapters. Neu ist ein "elektronisches Tankbüchlein" integriert und die Erkennung des aktuell eingelegten Gangs wurde verbessert. Sowohl Trips, Fahrer, Fahrzeuge und Tankvorgänge werden stets mit der Server-Anwendung synchronisiert. Diese zentrale Verwaltung der Daten bietet die Möglichkeit, EcoHelper auf verschiedenen Android-Geräten gleichzeitig zu verwenden.

Abbildung 2: Startbildschirm

EcoHelper-Android unterstützt alle Möglichkeiten des ebenfalls neu entwickelten EcoHelper-Servers.

Aus Zeitgründen musste auf die Implementierung einer Video- oder Bildaufnahme während eines Trips, die erweiterte grafische Anzeige von Fahrzeug- und Fahrdaten und die Berechnung des optimalen Schaltzeitpunktes verzichtet werden.

Die grosse Anzahl an implementierten Funktionen und die bei zahlreichen Testfahrten unter Beweis gestellte Funktionalität zeigen, dass die gesetzten Ziele mehr als erreicht wurden.

2.4. Challenges

Die folgende Aufzählung enthält Challenges, welche während des Projekts aufgetaucht und gelöst wurden.

- Daten müssen unter verschiedenen Geräten synchronisiert werden. Dabei gilt der EcoHelper-Server als Master. Der Benutzer kann seine Daten auch auf dem Server abändern.
- Kommunikation mit Webserver, wobei bei einem Smartphone eine Internetverbindung nicht garantiert ist
- Gangerkennung musste verbessert werden, da der bestehende Algorithmus mangelhaft war.
- Oberfläche während Trip soll möglichst konfigurierbar gestaltet werden. Sie soll sich über mehrere Screens ausdehnen und das Verschieben bzw. Entfernen von Elementen per Drag&Drop gestatten.
- Anforderung der Anwendung waren zu Beginn des Projekts nicht komplett ausgearbeitet und wurden im Verlauf genau definiert und mit eigenen Ideen erweitert.
- Je nach Gerätehersteller ist auf einem Smartphone eine modifizierte Android-Version installiert. Dies macht sich vor allem bei HTC stark bemerkbar.
- Für die OBD2-Schnittstelle haben wir einige Adapter erhalten, welche jedoch überhaupt nicht oder nur schlecht dokumentiert waren und teilweise ein fehlerhaftes Verhalten aufwiesen. Zur Implementation stand zudem nur ein sehr begrenzter Simulator zur Verfügung. Bei Testfahrten mit diversen Autos mussten wir auch erfahren, dass sich die Autohersteller nur mässig an den definierten Standard halten.
- Der Funktionsumfang der gesamten Anwendung ist sehr gross und brachte sehr viele kleine Diskussionspunkte mit sich.

2.5. Ausblick

Nach Abschluss der Bachelorarbeit wird die A-Z Verkehrsschule Ostschweiz AG EcoHelper-Android in der 2-Phasen Ausbildung einsetzen. Zudem wird das EcoHelper-System im Rahmen des Young Engineers-Projekts an der World Engineer‘s Convention vom 4. bis 9. September 2011 in Genf der breiten Öffentlichkeit präsentiert.

Diese Arbeit bietet eine solide Grundlage für die Weiterentwicklung der Anwendung. Einerseits könnten erweiterte Funktionen implementiert werden (z.B. Videostreaming, Schalthinweise). Andererseits wäre eine Portierung auf Android-Tablets denkbar.

3. Einleitung

3.1. Ausgangslage

Ein ökologischer Lebensstil dürfte spätestens nach den Zwischenfällen vom März 2011 in den Japanischen Atomkraftwerken wieder ein wichtigeres Thema werden. Der Mensch wir hoffentlich wieder vermehrt darauf achten, was er wirklich braucht und an welchen Punkten er sparen kann. Es scheint, dass diesbezüglich in der Mobilität noch viel Potential vorhanden ist. Nebst immer effizienteren Autos soll auch das Fahrverhalten jedes Einzelnen verbessert werden.

In früheren Arbeiten wurden bereits Anwendungen auf Symbian, iPhone und Windows Mobile entwickelt, welche das Aufzeichnen von Fahrten ermöglichen. Diese Daten werden an eine Webplattform gesendet, über welche der Benutzer seine Fahrten vergleichen und analysieren kann. Im Rahmen der vorliegenden Bachelorarbeit wurde eine Android-Anwendung mit erweiterter Funktionalität erstellt, welche die bestehende Webplattform mit Daten versorgen kann.

3.2. Hinweise zum Bericht

Dieser Bericht wurde in erster Linie für den Entwickler erstellt, der EcoHelper - Android weiterentwickelt. Zusätzlich wurden die Elemente für den Experte, den Gegenleser und den Betreuer eingebunden. Der Bericht dient nicht als Hilfe oder Anleitung für Anwender. Dafür ist innerhalb der Anwendung eine ausführliche Hilfe integriert.

4. Analyse

Fokus der Analysephase war die ausführliche Umfeldanalyse. Dies beinhaltete einerseits die Auseinandersetzung mit bestehenden Arbeiten und Anwendungen, anderseits die Einarbeitung in die Thematik.

4.1. Bestehende Arbeiten, Anwendungen und Webseiten

Analysiert wurden verschiedene frühere Arbeiten rund um das Projekt EcoHelper und kommerzielle Produkte im gleichen Themenbereich.

4.1.1. Vorarbeiten cnlab/HSR

iPhone-Anwendung EcoHelper

Die EcoHelper Anwendung für das iPhone entstand aus der Bachelorarbeit, im Frühjahressemester 2010, durch Selim Akyol und Edon Berisha.

Positiv:

- Gute Nutzung der iPhone GUI-Komponenten.
- Die Verwaltung der Fahrten überzeugt durch ihre intuitive Bedienbarkeit.
- Der Style der ganzen Anwendung orientiert sich sehr genau an die iPhone GUI-Guidelines.
- Dank Nutzung des GPS-Moduls im iPhone kann die Position während der Fahrt sehr genau aufgezeichnet und angezeigt werden.

Negativ:

- Anwendung läuft sehr unstabil (häufige Abstürze während Testfahrt)
- Anwendung verliert häufig Verbindung zu OBD-Adapter. Neuer Verbindungsaufbau muss durch betätigen eines Buttons initiiert werden.
- Gänge werden nicht korrekt erkannt
- Ohne einen angeschlossenen OBD-Adapter nützt die Anwendung nicht viel
- Im Einstellungsmenü ist es verwirrend, wie das OBD ausgeschaltet wird
- Der Login und das Fahrerprofil auf einer Bildschirmseite sorgen für Verwirrung
- Das Logo ist nicht ansprechend
- Der Trip-Löschen-Knopf und jener zur Auswertung der Fahrtresultate liegen zu nahe beieinander
- Die Eingabe eines Tripnamens ist unnötig
- Während eines Trips ist die Anzeige nur horizontal verfügbar.
- Für den Benutzer ist es unklar wohin die Daten gesendet werden

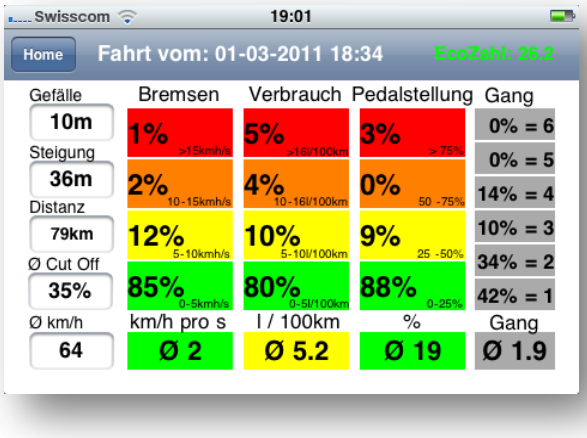


Abbildung 3: Testfahrt vom 01.03.2011 mit Audi A5

Symbian- Anwendung EcoHelper

Wie auch bei der IPhone Anwendung wurde die Symbian TourLive Mobile Anwendung vorgängig zu den Testfahrten betrachtet.

Positiv:

- Es werden zahlreiche und informative Messwerte dargestellt.
- Nicht erkennbare Messwerte werden klar gekennzeichnet.
- Verschiedene Info-Panels mit unterschiedlichen Informationen erleichtern Darstellung der aktuell gewünschten Informationen.
    - Die Aufteilung erfolgte in folgende Kategorien: Total Info, Trip Info, GPS Info, OBD Info, OBD Consumption, Pulse Info, Net Info und Logs.
- Status über GPS und die verbundenen Satelliten wird klar dargestellt
- Options und Exit gut ersichtlich
- Möglichkeit von Ein- und Ausschalten der Zusatzgeräte (z.B. OBD Schnittstelle, Pulsmesser) und bestimmter Funktionen (z.B. Foto und Videoaufnahme, Offline Modus)
- Werte werden zuverlässig ausgelesen und aktualisiert
- Upload der Daten funktioniert
- Zusätzlich zu OBD Werten kann Software mit Pulsmesser kommunizieren, ist für uns jedoch nicht relevant
- Log unterstützt Benutzer im Fehlerfall

Negativ:

- Keine Grafische Darstellung der Parameter
- Keine kategorische Unterteilung in Settings Dialog
- Persönliche Angaben wie Alter, Grösse, Gewicht und Geschlecht für uns nicht nötig
- Wenn man sich in der Konfiguration befindet, werden Navigationselemente der eigentlichen Software noch angezeigt, obwohl diese nicht angewählt werden können
- GUI ist nicht sehr ansprechend gestaltet→ Symbian like
- Net Info bringt in diesem Umfeld keinen Mehrnutzen (wurde seinerzeit hinsichtlich möglicher Leistungsmessungen im Hintergrund verlangt)
- Im Betrieb wird das Mobilgerät sehr warm. Nach ca. drei Stunden war der Akku bereits aufgebraucht.
    - Durch den ständigen Upload, die Kommunikation via GPS und OBD sowie das Aufnehmen von Filmen ist die Anwendung ein enormer Ressourcenfresser
- keine Hinweise zur Fahrweise
- Wechsel zwischen Infopanels muss mit genauem Klick auf Pfeil erfolgen, nicht gut bedienbar
- Beim Start von Service wird zunächst nach einem Bluetooth-Adapter gesucht, in dieser Zeit lässt sich die Anwendung nicht bedienen. Der User wird jedoch auf keine Weise darüber informiert, darum erscheint das Gefühl, dass die Anwendung abgestürzt ist
- Konfiguration durch Slider schwierig vorzunehmen
- Falls sich die Anwendung im Autostart-Modus befindet, führt ein SIM-Karten-Wechsel zum Absturz bzw. Lock der Anwendung (man muss zuerst den Autostart-Modus ausschalten; dann mit der neuen SIM-Karte verbinden; und danach kann man wieder auf Autostart umschalten)

Es wurden danach zwei Testfahrten unternommen, um die Anwendung unter echten Bedingungen zu testen. Daraus entstanden die weiter unten aufgeführten Berichte. Anschliessend wurde ein Fazit über die beiden Testfahrten gezogen.

Berichte zu den Testfahrten sind dem Kapitel 12.6 Symbian Testfahrten zu entnehmen.

Fazit

Die Anwendung liest die relevanten Daten aus, falls sie vom OBD-Adapter angeboten werden. Die rein numerische Darstellung der Werte ist vor allem während der Fahrt zu wenig aussagekräftig. Die gesamte Anwendung wirkt nicht sehr intuitiv, dies ist jedoch sicherlich zu einem Teil durch die Symbian User Interface Elemente gegeben. Gerade im Zusammenhang mit dem Verbindungsaufbau zur ODB2-Schnittstelle fehlt ein Userfeedback. Für unsere Arbeit muss viel Wert auf die informative Darstellung der Messwerte sowie eine gute Benutzerführung gelegt werden. Dies muss sicherlich anhand eines Usability Tests verifiziert werden.

Android-Anwendung EcoHelper

Die EcoHelper Anwendung für Android entstand aus der Bachelorarbeit im Jahr 2008 durch Cédric Menzi und Mathias Sulser. Damals war nur der Android-Emulator verfügbar, die Anwendung konnte noch nicht auf Smartphones getestet werden

Positiv:

- Durch den sehr einfach gehaltenen Screen sind alle ausgelesenen Parameter leicht zu erkennen
- Upload der Daten scheint zu funktionieren

Negativ:

- Keine Grafische Darstellung der Parameter
- Schlechte Darstellung der Einstellungen
  - Unübersichtlich
  - schwierig einzustellen
  - wahrscheinlich aufgrund der frühen Android-Version
- Nur Parameter auslesbar, keine Auswertung und nichts anderes
- OBD-Daten kommen nur aus Simulator (CSV) → Keine Messung
- Anwendung selbst macht auch keine Messung sondern erhält die gemessenen Daten nur vom Simulator
- Fahrtenmanagement fehlt komplett

Fazit

Man merkt der Anwendung an, dass sich Android zu diesem Zeitpunkt noch in den Kinderschuhen befunden hat. Die Anwendung macht einen "gebastelten" Eindruck und kann nicht wirklich durch den Benutzer eingesetzt werden, da das Ganze nur auf dem Simulator basiert, sprich keine Anbindung an OBD besitzt.

Windows Mobile 5.0 EcoTrainer

Analysiert wurde die EcoTrainer Anwendung für Windows Mobile 5.0, welches aus der Diplomarbeit, im 2007 von Bruno Krieg und Daniel Wydler, entstanden ist.

Leider stand uns für eine genaue Analyse kein Windows Mobile Testgerät zur Verfügung. Darum wurde nur die ausführliche Dokumentation sowie den mitgelieferten Code beurteilt.

Positiv:

- Oberfläche aufgeräumt, klarer Ablauf für Benutzer ersichtlich
- Parameter übersichtlich in Textform dargestellt
- Trips, Autos können erfasst werden
- Früher erfasste Autos können wiederverwendet werden
- Alte Trips können auf dem Gerät erneut geöffnet werden
- Trips können in CSV und KML (für Google Maps) exportiert werden

Negativ:

- Keine Grafische Darstellung der Parameter
- Grids erscheinen sehr unübersichtlich
- Nicht ersichtlich wie man den Trip beendet (Als Buttons sind nur System/Settings vorhanden)
- Es können keine Fahrer erfasst werden
- Umsetzungsverhältnis der Gänge muss von Hand eingegeben werden beim Erstellen eines Autos (dies ist eine Annahme, da aus Screenshots und Code nichts anderes ersichtlich ist)
- Es gibt keine wirkliche Tripauswertung auf dem Gerät
- Kein Direkter Upload der Daten
- Die Buttons scheinen ein bisschen klein gewählt (wohl Windows Mobile bedingt)

Fazit

Leider konnte man nur erahnen, wie sich diese Anwendung im Live-Betrieb verhält. Gerne hätte man noch den Gang-Erkennungs-Algorithmus geprüft. Die Übersicht über alle Screens vermittelt einen vollständigen Eindruck der Anwendung. Bei der Führung des Benutzers durch die Dialoge wurde eine gewisse Ordnung eingehalten.

Für die zu entwickelnde Anwendung ist sicherlich die Option des KML-Exports weiter zu Verfolgen. Des Weiteren ist die allgemeine Aneinanderreihung der Screens eine gute Basis auf welche man sich stützen kann. Ein Hauptscreen über den man zu den verschiedenen Funktionen (Autos, Trips) gelangen kann.

4.1.2. Kommerzielle Produkte

Torque free (Android)

Torque ist in zwei Versionen über den Google Market verfügbar. Eine kostenpflichtige, sowie eine etwas abgespeckte gratis Version. Wir haben hier die gratis-Version getestet.

Der Entwickler beschreibt die Anwendung wie folgt: [2]

> *Messen Sie das Drehmoment und die Leistung Ihres Autos, entdecken und löschen Sie Fehlercodes, sehen Sie, was Ihr Auto zu tun bekommt – in Echtzeit! Mit Hilfe eines Bluetooth ELM/OBD/PLX Adapters verwandeln Sie Ihr Android Telefon in ein Auto Performance und Fahrzeug Diagnosegerät mit dem Sie direkt in das Motormanagement blicken können. Darüber hinaus bietet Torque GPS Unterstützung, Logging- und Uploadfunktionen zur späteren Analyse, inkl. grafischer Aufbereitung direkt auf der Homepage.*

Abbildung 4: Screenshot Torque

Positiv:

- Die komplette Anwendung ist grafisch sehr ansprechend gestaltet.
- Die Messwerte können auf drei verschiedene Arten angezeigt werden:
  - Dial: Animierte Anzeige anhand einer Art analoger Zifferanzeige mit einem Zeiger und allenfalls numerischen Werten
  - Graph: Animiertes Chart, das den Verlauf eines Wertes über einer gewissen Zeit anzeigt
  - Display: Anzeige, die den aktuellen Messwert numerisch anzeigt
- Die verschiedenen Anzeigen können nach Belieben auf 7 verschiedenen Ebenen ein- bzw. ausgeblendet werden
- Auf der Hauptebene wird dem Benutzer angezeigt, ob Bluetooth aktiviert ist, ob GPS vorhanden ist und ob eine Verbindung zu der OBD2 Schnittstelle besteht
- Beim Start der Anwendung wird der Benutzer darauf hingewiesen, dass die GPS-Funktion noch nicht aktiviert ist, und bietet mit einem Button die Möglichkeit, direkt auf den GPS-Einstellungsdialog von Android zu wechseln.
- Settings Dialog ist sehr umfangreich, gut strukturiert und klar beschrieben

Negativ:

- Die Auswahl der gewünschten Messwerte ist lediglich durch eine grosse Liste gegeben. Dies ist sehr unübersichtlich und sollte durch intelligente Unterteilung in Kategorien verbessert werden.
- Die Infoanzeige, ob Bluetooth aktiviert ist, ob GPS vorhanden ist und ob eine Verbindung zu der OBD2 Schnittstelle besteht, wird lediglich auf der Hauptebene der 7 verschiedenen Ebenen angezeigt. Hier wäre wünschenswert, wenn diese Info auf jeder Ebene angezeigt wird.
- keine Hinweise zur Fahrweise bzw. keine Fahrtanalyse
- keine Fahr- und Tripverwaltung
- Im ersten Moment ist nicht ganz klar, wie die Anwendung zu bedienen ist, jedoch wird einem schnell bewusst, dass man entweder über "Longclicks" oder über das Menü die gewünschten Messwerte anzeigen kann

[2] Quelle: https://market.android.com/details?id=org.prowl.torque&feature=search_result; Letzter Zugriff 14. 06.2011

Fazit

Der Funktionsumfang von Torque ist riesig. Die grafische Gestaltung ist sehr ansprechend umgesetzt worden. Diese Anwendung bietet sehr guten Input für die EcoHelper Anwendung

Rev Lite (iPhone)

Die Anwendung Rev gibt es in zwei Versionen. Sie unterscheiden sich lediglich im Funktionsumfang, der bei der Gratisversion eingeschränkt ist. Die kostenpflichtige Version kostet CHF 44.00 (Stand: 28.02.2011) und kann über den App-Store bezogen werden.

Es werden folgende Funktionen unterstützt:

- Auto erfassen
- Auto-Foto erfassen
- Misst Rundenzeigen
- Gute Darstellung der Beschleunigungen
- Trip kann in Echtzeit auf der Karte (Google Maps) angezeigt werden
- OBD2-Messwerte können angezeigt werden

Die Rev Lite Anwendung erkannte die angeschlossene WiFi-Schnittstelle nicht. Es konnten dann auch keine weiteren Analysen durchgeführt werden.

Abbildung 5: Screenshot RevLite

DashCommand (iPhone)

Die Anwendung kostet CHF 55.00 (Stand: 28.02.2011) im App-Store und wurde nicht getestet. Diese wird nur zur Vollständigkeit hier aufgeführt. Die Funktionsliste ist unter folgendem Link einsehbar: http://itunes.apple.com/app/dashcommand-obd-ii-gauge-dashboards/id321293183?mt=8#

FUZZYCar (iPhone)

Die Anwendung ist gratis (Stand: 28.02.2011) im App-Store verfügbar. Auch die FUZZYCar Anwendung konnte keine Verbindung zur OBD-Schnittstelle herstellen. Es konnte keine Analyse durchgeführt werden.

Die Funktionsliste ist unter folgendem Link einsehbar:
http://itunes.apple.com/de/app/fuzzycar/id327822395?mt=8#

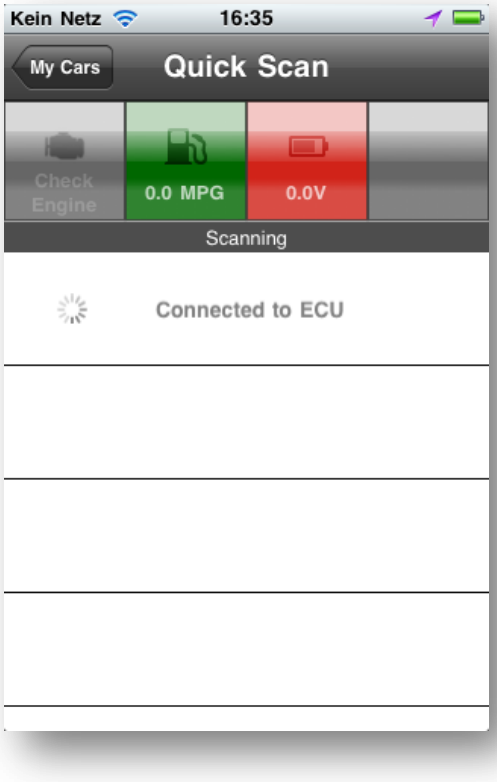


Abbildung 6: Screenshot FUZZYCar

Fazit Anwendungen

Jede analysierte Anwendung hat ihre eigenen Stärken und Schwächen. Den mit Abstand besten Eindruck machte die Android Anwendung Torque. Dies ist vor allem auf die enorme Funktionalitäten und das ansprechende User Interface zurückzuführen

Die beiden getesteten iPhone-Anwendungen (Rev Lite & FUZZYCar) konnten trotz Anstrengungen keine Verbindung zur OBD-Schnittstelle via Kiwi WiFi herstellen. Da beide Anwendungen gratis sind kann nicht ausgeschlossen werden, dass es andere kostenpflichtige Anwendungen gibt, die funktionieren.

Dies zeigt zudem, dass es nicht trivial ist eine benutzerfreundliche Anwendung zu erstellen, welche sich problemlos in Betrieb setzen lässt. Des Weiteren sollte bei der Erstellung der Android EcoHelper-Anwendung auf die fehlertolerante und gut getestete Implementierung der Schnittstellen geachtet werden

4.1.3. Webportale zur Tankerfassung

Dadurch, dass diese Arbeit von einem Dreierteam durchgeführt wurde, bestand die Möglichkeit den Projektumfang zu erweitern. Deshalb wurde das „elektronische Tankbüchlein“ eingeführt und die untenstehende Analyse durchgeführt.

Durch den Vergleich der unterschiedlichen Webportale konnten folgende Punkte eruiert werden:

- Ähnliche Seiten wie http://www.verbrauchsrechner.de
- Funktionalitäten der Webseiten
- Verbesserungspotential in Bezug auf Funktionalität und Bedienbarkeit
- Erstellung einer Featurelist für eigene Webseite
- Abschätzung Zeitaufwand

Verfügbare Webportale

http://www.verbrauchsrechner.de

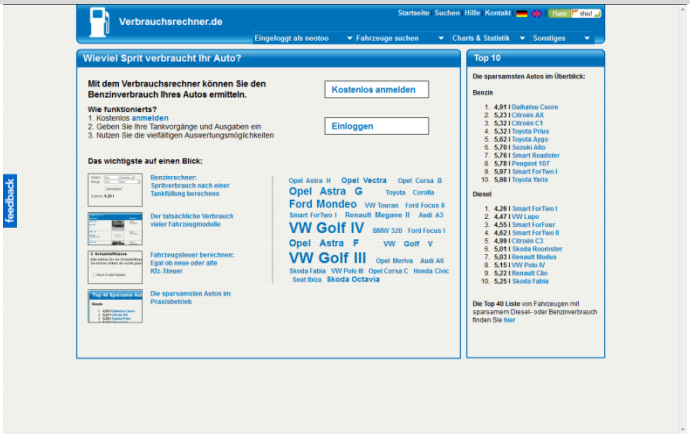


Abbildung 7: www.verbrauchsrechner.de

http://www.spritmonitor.de

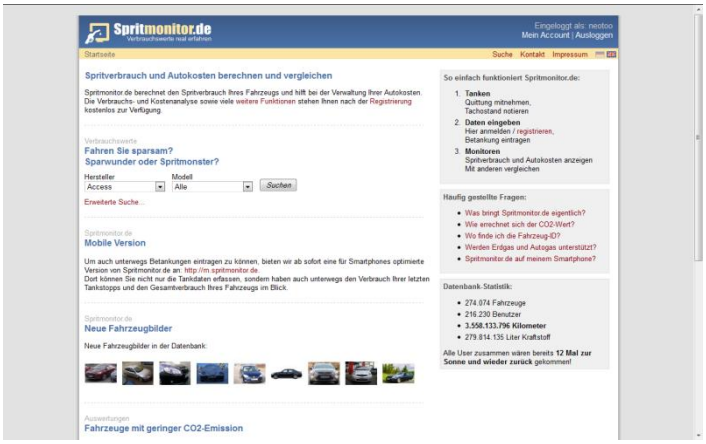


Abbildung 8: www.spritmonitor.de

http://www.motor-kompakt.de

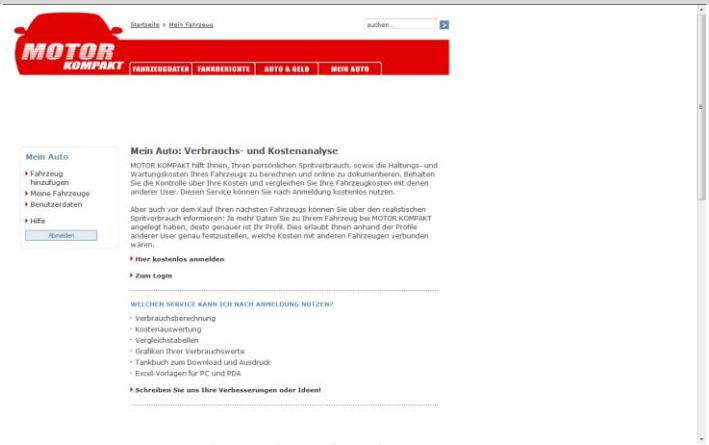


Abbildung 9: www.motor-kompakt.de

http://www.autoemissionen.at

Abbildung 10: www.autoemissionen.at

http://www.ecodrive.org

Abbildung 11: www.ecodrive.org

Die obigen Abbildungen zeigen die Startseiten der untersuchten Webseiten zur Tankverwaltung. Zudem erkennt man, wie die Seiten aufgebaut sind und man sieht deren Funktionsumfang.

Fazit Vergleich Webportale

Die untersuchten Webseiten bzw. deren Funktionen können in zwei Kategorien unterteilt werden: Community-Plattformen und Informationsseiten.

Bei den Community-Seiten geht es vor allem darum, die gesamten Kosten seiner Fahrzeugnutzung mit denjenigen Anderen vergleichen zu können. Dazu können Statistiken und Auswertungen zu den eingegebenen Tankvorgängen erstelle werden. Bei diesen geht es hautsächlich um den Spritverbrauch und die dadurch entstandenen Kosten. Es können zwar Angaben zur Fahrtweise und verschiedenen Einflussfaktoren eingegeben werden, aber diese werden nirgends berücksichtigt, weder bei den meisten Statistiken als auch bei den Analysen und Vergleichen nicht. Die Einflüsse der verschiedenen Faktoren auf den Verbrauch werden kaum berücksichtigt.

Die Informationsseiten bieten viele Tipps rund um spritschonendes und umweltfreundliches Autofahren. Dies nützt jedoch dem Leser nur bedingt, da er solche Tipps besser gerade dann erhält, wenn er dagegen verstösst.

Leider war nur eine der untersuchten Seiten als Mobile-Version verfügbar. Dies obwohl sich das Smartphone optimal für die Eingabe von Tankvorgängen eignen würde, da es stets mitgetragen wird. Durch die Verwendung eines Client-Handyprogramms könnten zudem weitere Parameter, wie zum Beispiel GPS-Position, Datum und Uhrzeit, ohne grossen Aufwand mitgespeichert und gesammelt werden.

4.1.4. Funktionsübersicht

Die Vergleichstabelle auf den folgenden Seiten zeigt eine Funktionsliste aller untersuchten Anwendungen und Webseiten. Sie gibt an welche Funktion von welcher Anwendung oder Webseite unterstützt wird. Nach der Vergleichstabelle befinden sich noch einige Bemerkungen.

Funktionen	Anwendungen: Android EcoHelper (BA 2011)	iPhone EcoHelper (BA 2010)	JavaME TourLive (BA 2009)	Android EcoTrainer (BA 2008)	Windows Mobile 5.0 EcoTrainer (DA 2007)	Android Torque (free)	iPhone Rev Lite	iPhone DashCommand	iPhone FUZZYCar	Webseite: verbrauchsrechner.de	Webseite: spritmonitor.de	Webseite: motor-kompakt.de	Webseite: autoemissionen.at	Webseite: ecodrive.org
Benutzerverwaltung														
Registrieren	✓								✓	✓	✓	✓		
E-Mailadresse - mit Aktivierungsmail	✓								✓	✓	✓	✓		
Passwort	✓								✓	✓	✓	✓		
Verschiedene Rollen	✓													
Privat - für den privaten Gebrauch	✓													
Fahrlehrer - Fahrlehrer kann seine Fahrschüler verwalten	✓													
WAB - Vertiefungskurs für Lernfahrer	✓													
Login	✓								✓	✓	✓	✓		
E-Mailadresse	✓								✓	✓	✓	✓		
Passwort	✓								✓	✓	✓	✓		
Datenschutzbestimmungen	✓								✓	✓	✓	✓		
Logout	✓								✓	✓	✓	✓		
Synchronisierte Daten werden nicht gelöscht	✓								✓	✓	✓	✓		

Fahrerverwaltung														
Fahrer hinzufügen	✓	✓												
Fahrer ändern	✓	✓												
Name	✓	✓												
E-Mailadresse	✓													
Fahrer löschen	✓	✓												
Fahrer synchronisieren	✓													

Autoverwaltung														
Auto erfassen	✓	✓			✓		✓		✓	✓	✓	✓		
Hersteller auswählen oder selber neu erfassen	✓	✓			✓				✓	✓	✓	✓		
Model auswählen oder selber neu erfassen	✓	✓			✓				✓	✓	✓	✓		
Variante auswählen oder selber neu erfassen	✓									✓	✓	✓		
Autodetails erfassen	✓	✓					✓		✓	✓	✓	✓		
Aufbau	✓									✓	✓	✓		
Getriebeart	✓									✓	✓	✓		
Anzahl Gänge	✓	✓			✓					✓	✓	✓		
Treibstoff	✓									✓	✓	✓		
Auto-Name	✓	✓							✓	✓	✓	✓		
Optional	✓						✓		✓	✓	✓	✓		
Hubraum	✓									✓	✓	✓		
Fahrzeug-Gewicht	✓	✓					✓			✓	✓	✓		
Jahrgang	✓	✓			✓				✓	✓	✓	✓		
Max. Reichweite pro Tankfüllung	✓									✓	✓	✓		
VIN		✓												
Foto/Bild des Fahrzeugs							✓			✓	✓			
Auto löschen	✓	✓					✓			✓	✓	✓		
Autos synchronisieren	✓								✓					
Eingegebene Daten anzeigen	✓	✓			✓		✓			✓	✓	✓		

OBD2 Anschlussplatz-Informationssystem														
Neuer Anschlussplatz erfassen	✓													
Foto aufnehmen	✓													
Beschreibung hinzufügen	✓													
Genaue Steckplatzposition angeben	✓													
Vorhandene Anschlussplätze anzeigen	✓													

Tankverwaltung															
Tankvorgang erfassen	✓									✓	✓	✓		
Auto auswählen	✓									✓	✓	✓		
Treibstoff auswählen	✓									✓	✓	✓		
Tachostand	✓									✓	✓	✓		
Kilometer	✓									✓	✓	✓		
Meilen	✓									✓	✓	✓		
Getankte Menge	✓									✓	✓	✓		
Liter	✓									✓	✓	✓		
US-Gallonen	✓									✓	✓	✓		
GB-Gallonen	✓									✓	✓	✓		
Gesamtkosten verschiedenen Währungen	✓									✓	✓	✓		
Volltankung	✓									✓	✓	✓		
Optional	✓									✓	✓	✓		
Durchschnitts-Verbrauch, welcher das Cockpit anzeigt	✓													
Durchschnitts-Geschwindigkeit, welche das Cockpit anzeigt	✓													
Persönliche Bemerkung	✓									✓	✓	✓		
GPS-Position wird automatisch erfasst	✓													
Datum und Uhrzeit werden automatisch erfasst	✓													
Angaben zum Fahrverhalten											✓	✓		
Tankvorgang ändern	✓									✓	✓	✓		
Tankvorgang löschen	✓									✓	✓	✓		
Tankvorgange synchronisieren	✓													
Tankvorgänge import- /exportieren										✓	✓	✓		
Tankvorgänge anzeigen	✓									✓	✓	✓		
Eingegebene Daten	✓									✓	✓	✓		
Tank-Standort auf Google Maps	✓													
Berechnete Werte	✓									✓	✓	✓		
Gefahrene Distanz	✓									✓	✓	✓		
Verbrauch	✓									✓	✓	✓		

Trip aufzeichnen														
Fahrer auswählen	✓	✓					✓							
Auto auswählen	✓	✓			✓		✓							
Trip-Name		✓			✓									
OBD2 Adapter konfigurieren	✓	✓			✓		✓	✓	✓					
Freie Konfiguration der anzuzeigenden Daten	✓					✓								
Unterstützte Daten	✓	✓	✓		✓	✓	✓							
Ortungsinformationen	✓	✓			✓	✓	✓							
Bearing	✓				✓									
Distanz	✓				✓									
Genauigkeit	✓						✓							
Geschwindigkeit	✓	✓			✓	✓								
Höhe	✓	✓			✓									
Längengrad	✓	✓			✓	✓	✓							
Breitengrad	✓	✓			✓	✓	✓							
Unterstützte OBD-Werte durch das Auto	✓					✓	✓	✓	✓					
Beschleunigungssensoren	✓					✓	✓							
Berechneter Verbrauch	✓	✓	✓		✓			✓						
Tripinformationen	✓	✓			✓		✓	✓						
Trip in Google Maps anzeigen	✓						✓							
Automatische Gangerkennung	✓	✓	✓		✓									
OBD-Daten	✓	✓	✓		✓	✓	✓	✓	✓					
Distanz	✓	✓	✓		✓	✓		✓	✓					
Druck Einlasskanal	✓	✓				✓	✓	✓	✓					
Einlasskanaltemperatur	✓	✓				✓	✓	✓	✓					
Treibstoff-Druck	✓	✓				✓	✓	✓	✓					
Treibstoff-System	✓	✓				✓	✓	✓	✓					
Gang (berechnet)	✓	✓	✓		✓									
Gangwechsel (berechnet)	✓	✓												
Gaspedalstellung	✓	✓			✓	✓	✓	✓	✓					
Geschwindigkeit	✓	✓	✓		✓	✓	✓	✓	✓					
Einlassmasse der Luft zum Motor	✓	✓			✓	✓	✓	✓	✓					
Umdrehungen pro Minute des Motors	✓	✓	✓		✓	✓	✓	✓	✓					
Schubabschaltung	✓	✓				✓	✓	✓	✓					
Tankfüllung	✓	✓				✓	✓	✓	✓					
Umgebungstemperatur	✓	✓				✓	✓	✓	✓					
Totaler Verbrauch	✓	✓			✓	✓	✓	✓						
Aktueller Verbrauch	✓	✓			✓	✓	✓	✓						
Selber definierte Daten abfragen						✓								
Verschiedene Aktions-Buttons	✓	✓				✓	✓							
Trip beenden	✓	✓				✓	✓							

	Trip pausieren	✓													
	Automatisch erkannte Gänge zurücksetzen	✓	✓												
	Tankvorgang während eines Trips durchführen	✓													
	Trip-Daten auf den Server laden	✓	✓												
	Automatisch - Online-Modus	✓	✓												
	Manuell - Offline-Modus	✓	✓												
	Trip-Zusammenfassung anzeigen	✓	✓					✓							
	Auto	✓						✓							
	Datum	✓						✓							
	Dauer	✓						✓							
	Distanz	✓	✓					✓							
	Geschwindigkeiten	✓	✓					✓							
	RPMs	✓						✓							
	Schaltvorgänge	✓													
	Verbrauch	✓													
	Eco-Zahl	✓	✓												
	Trip auf Karte anzeigen	✓						✓							
Tripübersicht															
	Trips synchronisieren	✓								✓					
	Trip-Zusammenfassung anzeigen	✓	✓							✓					
	Trips löschen	✓								✓					
	Trips exportieren (KLM-Format)					✓									
	Trips analysieren - mehrere Trips vergleichen	✓													
Einstellungen															
	Online-/Offline-Modus	✓	✓							✓					
	Synchronisation ein-/abschaltbar	✓								✓					
	OBD2 Adapters verwalten	✓				✓		✓	✓	✓					
	Services ein-/abschaltbar	✓													
	Bildschirm	✓													
	GPS	✓													
	OBD	✓													
	Beschleunigungssensoren	✓													
	Masseinheiten	✓						✓		✓	✓				
	Warnton bei OBD Verbindungsproblemen	✓													
	Aktueller Treibstoffpreis	✓									✓				
	Mehrsprachigkeit ist möglich	✓									✓				

Hilfen															
	Ausführliche Hilfeseiten in der Anwendung integriert	✓						✓							
	Spartipps													✓	✓

Tabelle 2: Vergleichstabelle frühere und andere Arbeiten

Die iPhone EcoHelper- Anwendung hat die Möglichkeit Fahrer und Autos zu erfassen und zu verwalten, jedoch werden diese nicht mit der Server-Anwendung synchronisiert. Zudem muss der Benutzer während des Trips, falls die Anwendung die Verbindung zum OBD-Adapter verliert, manuell den neuen Verbindungsaufbau bestätigen. Dies soll bei der aktuellen EcoHelper-Anwendung automatisch im Hintergrund, ohne Benutzerinteraktion, geschehen. Aber der Benutzer wird akustisch darauf aufmerksam gemacht, falls er dies so eingestellt hat. Des Weiteren ist der Nutzen der Anwendung ohne OBD2-Adapter nicht ersichtlich. Auch dies sollte geändert werden.

Bei der Symbian EcoHelper-Anwendung sind die Daten während der des Trips gut ersichtlich und klar gekennzeichnet. Die Aufteilung in verschiedene Panels ist sehr ansprechend und könnte von der Idee her übernommen werden, da es die Übersichtlichkeit erhöht.

Die Android EcoHelper-Anwendung von der Bachelorarbeit aus dem Jahr 2008 ist äusserst schwierig zu konfigurieren. Es sollte bei der aktuellen EcoHelper-Anwendung viel Wert auf einfache Konfiguration gelegt werden. Diese Anwendung kann nicht mit einem Auto kommunizieren, entsprechend konnten wir in der Vergleichstabelle keine Punkte vergeben.

Die Windows Mobile 5.0. EcoTrainer-Anwendung zeigt nach einem Trip keine Auswertungen. In diesem Fall nützt die Anwendung ohne die dazugehörige Webseite nicht viel. Es sollte zumindest eine kurze Trip-Zusammenfassung angezeigt werden, die auch ohne funktionierende Internetverbindung funktioniert.

Die Android Torque free Anwendung ist grafisch sehr ansprechend und sollte bei der aktuellen EcoHelper-Anwendung den Aufbau und die Bedienung der Ansicht eines laufenden Trips inspirieren. Jedoch muss gesagt werden, dass die Android Torque free Anwendung nur Messwerte anzeigen kann ohne das Prinzip eines Trips zu kennen.

Die kostenlose iPhone Anwendung Rev Lite wurde für Hobby-Rennfahrer entwickelt, welche ihre Rundenzeiten messen und analysieren wollen. Darum wurde sehr viel Wert auf die Anzeigen von Zeit und Beschleunigungen gelegt. Für die aktuelle EcoHelper-Anwendung bietet die Rev Lite-Anwendung keinen Input, da sich die Verwendungszwecke deutlich unterscheiden.
Wie auch die Rev Lite-Anwendung ist die iPhone DashCommand-Anwendung an Rennfahrer gerichtet.

iPhone FUZZYCar-Anwendung zeigt verschiedene OBD2-Messwerte. Es wurde hauptsächlich für die Diagnose von OBD2-Daten entwickelt.

4.2. Technische Analyse und Machbarkeitsstudien

Bei den verschiedenen technischen Analysen geht es hauptsächlich um die Machbarkeit auf der Android-Plattform. Zum Teil wurden Prototype-Anwendungen entwickelt, um die Machbarkeit und Schwierigkeiten aufzuzeigen.

4.2.1. Smartphone Akkutest mit Prototyp

Ziel

Es soll ermittelt werden, wie lange das Testgerät (HTC Desire) unter Last laufen kann. Unter Last bedeutet, dass die unten aufgeführten Daten kontinuierlich auf dem Mobiltelefon abgefragt und diese an einen Server übermittelt werden.

Folgende Daten werden im Abstand von 800 Millisekunden gesammelt und an den Server gesendet:

- Testzeit: Zähler, Anzahl Datensätze
- Mobile Time: Die Uhrzeit und das Datum auf dem Handy
- GPS-Latitude: Die Breiten-Position über den GPS-Location-Provider
- GPS-Longitude: Die Längen-Position über den GPS-Location-Provider
- GPS-Speed: Die Geschwindigkeit, welcher vom GPS-Provider zur Verfügung gestellt wird
- GPS-Altitude: Die Höhe über dem Meeresspiegel, welcher vom GPS-Provider erhalten wird
- GPS-Bearing: Grad, Abweichung zum Norden
- GPS-Accuracy: Die Genauigkeit, der GPS-Daten (Lat / Lon)
- Acceleration-X: Die Beschleunigung in der X-Achse
- Acceleration-Y: Die Beschleunigung in der Y-Achse
- Acceleration-Z: Die Beschleunigung in der Z-Achse
- Best Location Provider: Der zur Zeit genauste Location-Provider, welcher das Mobiltelefon liefern kann
- Best Latitude: Die Breiten-Position, des besten Location-Providers
- Best Longitude: Die Längen-Position, des besten Location-Providers

Weiter ist der Bildschirm während des ganzen Testdurchgangs in voller Helligkeit aktiv.

Vorgehen

Es wird ein Testsystem aufgebaut, welches sich aus einer Android- und einer Server-Anwendung zusammensetzt. Die Android Anwendung sammelt die oben erwähnen Daten und sendet diese an die Server Anwendung. Diese speichert die Daten in einer Datenbank.

Ein Testvorgang läuft wie folgt ab:

1. Handy wird vollständig aufgeladen.
2. Server-Anwendung gestartet
3. Handy-Anwendung gestartet
4. Das Handy wird im Bereich einer WLAN-Verbindung herumgetragen bis es von selber abstellt und keine Informationen zum Server schicken kann.

Die Resultate dieses Tests könnten durch folgende Faktoren stark beeinflusst werden:

- Aktivierte Dienste (3G, Bluetooth, GPS)
- Empfangsqualität (Mobile, GPS)
- Alter/Qualität des Akkus

Ergebnis

Start: 09:50 mit vollem Akku

Als der Akku bei 15% Leistung war, kam ein Hinweis, man solle bitte das Handy an die Stromversorgung anschliessen

Stopp: 12:52 Akku bei 11%

Daraus ergibt sich, dass die Anwendungen innerhalb von 182 Minuten 89% Akkuladung verbraucht hatten. Mit der Annahme, dass der Akku linear abnimmt ergibt sich:

```
182 Minuten =  89% Akku-Laufzeit
X   Minuten = 100% Akku-Laufzeit
```

Berechnung von X (Akku-Laufzeit unter Last):

```
X = 100 * 182 / 89 = 204.5 Minuten
```

D.h. mit einer Sicherheitsmarge von 10% kann man davon ausgehen, dass sich die Anwendung bei vollgeladenem Akku währen drei Stunden betreiben lässt.
Weiter konnte eruiert werden, dass die durch USB gelieferte Spannung von 5V bei maximal 500mA ausreicht um den durch die Anwendung benötigten Stromverbrauch auszugleichen.

4.2.2. Positionsbestimmung

Ziel

Aus früheren Arbeiten ist bereits bekannt, wie auf Positionsdaten zugegriffen werden kann. Dazu kann die Klasse LocationManager und das Interface LocationListener verwendet werden.

In diesem Abschnitt soll analysiert werden, wie genau die gelieferten Daten über die verschiedenen Provider (GPS, Network) sind und welche Daten welcher Provider liefert.

Vorgehen

Zur Überprüfung der Provider haben wir eine kleine Testanwendung geschrieben welche die Genauigkeit der Messungen ausgibt. Es wurde dann eine ca. 25 minütige Autofahrt durchgeführt bei der die Anzahl Messungen sowie die Durchschnittliche Genauigkeit aufgezeichnet wurde.

Ergebnis Genauigkeit

	GPS	Network
Anzahl Messungen	1260	27
Durchschnittliche Genauigkeit	16.8m	1713.48m

Tabelle 3: Ergebnisse Genauigkeit Tests Ortungsbestimmung

Wie man der Tabelle entnehmen kann erhält man durch GPS wesentlich häufiger erheblich genauere Messungen.

Für die Anwendungen heisst dies, dass wenn GPS-Empfang vorhanden ist, sollte dieser in jedem Falle verwendet werden. Der Network-Provider dient dabei als Fallback bei GPS-Ausfällen.

Ergebnisse Parameter

Parametername	Masseinheit	GPS	Network
Longitude	-	✓	✓
Latitude	-	✓	✓
Accuracy	m	✓	✓
Speed	m/s	✓	
Bearing	Grad	✓	
Altitude	m.ü.M	✓	

Tabelle 4: Ergebnisse Parameter Tests Ortungsbestimmung

Weiter war nicht klar, was genau das Bearing aussagt, mit folgendem Bild soll dieser Sachverhalt geklärt werden:

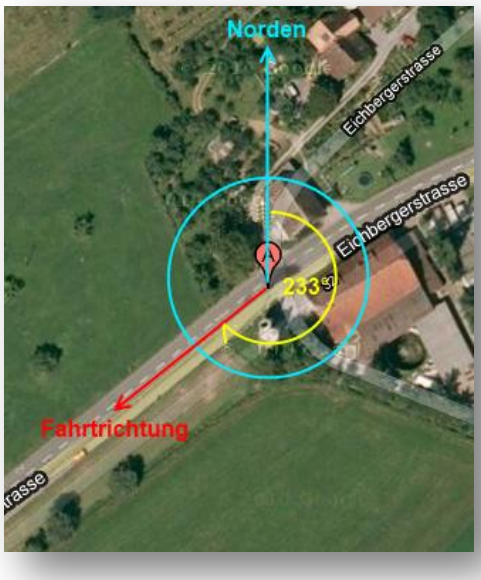


Abbildung 12: Beschreibung GPS Bearing

Das Bearing entspricht also der Fahrtrichtung. Falls sich das Gerät im Stillstand befindet wird 0.0 zurückgeliefert.

4.2.3. Foto / Video / Streaming

Ziel

Es soll geprüft werden, ob Android die Möglichkeit bietet im Hintergrund Videos und Bilder aufzunehmen und diese an den Server zu senden.

Ergebnis

Die Analyse hat ergeben, dass es nicht möglich ist, im Hintergrund Bilder und Videos aufzunehmen, solange diese nicht in einem Preview auf dem Screen angezeigt werden. Diese Einschränkung kann jedoch einfach umgangen werden in dem man einen Preview erstellt und diesem wie folgt die Grösse von 1x1 px zuweist:

```
public void setPreviewSize (int width, int height)
```

Somit ist das Aufnehmen von Videos und Bildern im Hintergrund möglich.

Die zweite Problemstellung ist das Streaming von einem Android-Gerät zu einem Server. Android bietet von sich aus keine solche Möglichkeit. Die Recherchen haben ergeben, dass es aber durch einige Umwege möglich wäre diese zu implementieren. Bestes Beispiel dafür sind bereits existierende Videotelefonie-Anwendungen.

Die Grundidee sieht dabei wie folgt aus:

```
String hostname = "your.host.name";
int port = 1234;
Socket socket = new Socket(InetAddress.getByName(hostname), port);
ParcelFileDescriptor pfd = ParcelFileDescriptor.fromSocket(socket);
MediaRecorder recorder = new MediaRecorder();
// Additional MediaRecorder setup (output format ... etc.) omitted
recorder.setOutputFile(pfd.getFileDescriptor());
recorder.prepare();
recorder.start();
```

Das Problem dabei liegt darin, dass der MediaRecorder die Daten aufnimmt und diese in den OutputStream schreibt. Sobald die Aufnahme beendet wurde springt der MediaRecorder im Stream zurück und schreibt an den Anfang die nötigen Informationen zur Länge der anzuzeigenden Pakete. Da dies in einem Stream, der über das Netzwerk gesendet werden kann, nicht ohne weiteres möglich ist, benötigt diese Lösung grösseren Implementationsaufwand. Aus zeitlichen Gründen wurde auf eine erweiterte Analyse verzichtet und dieses Feature als „Nice-to-Have“ definiert.

Für weitere Arbeiten an diesem Feature kann das SipDroid-Projekt[3] zu Hilfe gezogen werden. Dieses packt die durch den MediaRecorder erhaltenen Daten in - **RealTime Streaming Protocol** (*RTSP*)-Pakete ab und sendet diese an einen definierten Server.

4.2.4. Beschleunigungssensoren

Nebst der Lieferung der Rohdaten der Beschleunigungssensoren des Smartphones wurde untersucht, ob es eine Möglichkeit gibt, die tatsächliche Beschleunigung in Fahrtrichtung und die seitlichen Beschleunigungen zu berechnen. Im ersten Anlauf wurde angenommen, dass das Smartphone

[3] http://sipdroid.org/

während der Fahrt nicht fixiert wird. Dies hat zur Folge, dass nebst der eigentlichen Beschleunigung des Fahrzeuges noch zusätzlich Beschleunigungskräfte durch das Bewegen des Smartphones entstehen können. Aus diesem Grund wurde folgender Ansatz ausgearbeitet:

Beschleunigungsberechnung mit automatischer Kalibrierung

Solange das Gerät nicht beschleunigt wird, liefern die Sensoren 9.81 m/s^2. Dies entspricht der Erdbeschleunigung.

$$\vec{g} = 9.81 \frac{m}{s^2}$$

$$|\vec{s}| = Werte\ von\ Sensoren = \sqrt{x^2 + y^2 + z^2}$$

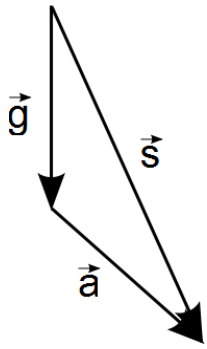


Abbildung 14: Beschleunigungsvektoren

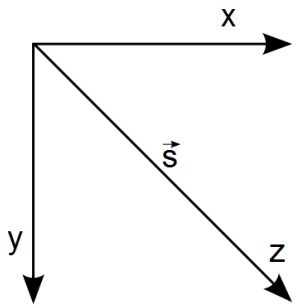


Abbildung 13: Beschleunigungsvektor s im Raum

$$wenn\ |\vec{s}| - |\vec{g}| = 0 \rightarrow keine\ Beschleunigung$$
$$wenn\ |\vec{s}| - |\vec{g}| \neq 0 \rightarrow Beschleunigung\ \vec{a}$$

$$\vec{a} = \begin{pmatrix} S_x - & g_x \\ S_y - & g_y \\ S_z - & g_z \end{pmatrix}$$

$$\left.\begin{matrix} g_x \\ g_y \\ g_z \end{matrix}\right\} Werte\ bei\ der\ letzten\ Kalibrierung$$

Idee: solange $|\vec{s}| - |\vec{g}| = 0$ kann dies als kalibrierter Zustand genommen werden. Wenn unmögliche Abweichungen auftreten, kann davon ausgegangen werden, dass das Smartphone durch eine Handbewegung beschleunigt wird. Sobald sich die Werte in einem annehmbaren Bereich befinden, kann die obige Berechnung vorgenommen werden.

Lieferung der Rohdaten der Beschleunigungssensoren

Als bekannt wurde, dass davon ausgegangen werden kann, dass das Smartphone im Auto stabilisiert wird, wurde auf die obige Berechnung verzichtet. Nun werden die Rohdaten der Beschleunigungssensoren gespeichert und übertragen.

4.2.5. Datenübertragung

Ziel

Bei der Analyse der Datenübertragung wird gezeigt, ob es möglich ist, JavaScript Object Notation (JSON) -Formatierte Nachrichten über das http-Protokoll nach dem Representational State Transfer

(REST)-Prinzip zu übertragen. Des Weiteren sollte eine Software-Komponente entwickelt werden, welches das Senden und Empfangen von Anfragen respektive Antworten, erleichtert. Diese Komponente soll auf der Android Plattform laufen und alle Interessenten über die Antwort informieren können.

Um die oben genannten Anforderungen zu erreichen, wurde ein Testsystem aufgebaut. Es besteht aus einem Android-Handy und einem Windows 2008 Server, auf dem ein Tomcat-Webserver läuft. Danach wurde auf dem Server eine Testschnittstelle erstellt, welche REST-Anfragen verarbeiten kann. Auf dem Mobiltelefon wurde ein Service implementiert, welcher den Service auf dem Server ansprechen kann.

Ergebnis
Die Analyse hat gezeigt, dass alle Anforderungen erfüllt werden konnten. Die Grundfunktionalitäten in den Anwendungen sind vorhanden und dadurch wurde gezeigt, dass die Datenübertragung so möglich ist. Es müssen aber sicher im Verlauf des Projekts Anpassungen und Erweiterungen am Service durchgeführt werden. Für die Umwandlung von Java-Objekten ins JSON-Format wurde eine externe Komponente namens Gson verwendet.

4.2.6. Gangberechnung

Ziel
Die Gangerkennung der Symbian-Anwendung funktioniert gut. Dies wurde uns auch von Herrn Peter Heinzmann bestätigt. Der bestehende Algorithmus sollte analysiert und gegebenenfalls verbessert werden.

Ergebnis
Wir konnten die Formel sowie die Idee des Algorithmus grösstenteils übernehmen. Da wir zu diesem Zeitpunkt der Analyse nur in etwa die Daten abschätzen können welche durch das Auto geliefert werden wurde hier auf eine weitere Analyse verzichtet.

Während der Implementation wurden diverse Testfahrten mit verschiedenen Autos durchgeführt bei denen einige Mängel des bestehenden Algorithmus erkannt werden konnten. Welche Mängel wir ermitteln konnten und was für eine Lösung dazu implementiert wurde ist dem Kapitel [KAPITEL GANGERKENNUNG] zu entnehmen.

4.2.7. WiFi

Ziel
Durch die Analyse der WiFi Schnittstellen wurde geprüft, wie die Kommunikation zwischen OBD und Smartphone via WiFi aufgebaut werden kann.

Ergebnis
Durch das Studium der Android WiFi Dokumentation wurde ersichtlich, wie eine Verbindung mit einem WiFi Netzwerk aufgebaut werden kann. Zugriff auf das WiFi erhält man über den entsprechenden Manager welchen man wie folgt erhält:

```
wifiManager = (Manager) context.getSystemService(Context.WIFI_SERVICE);
```

Als nächstes wurde versucht, den vorhanden PLX Kiwi WiFi Adapter zu verbinden. Die Tatsache, dass dieser im Adhoc Modus läuft, stellte sich als grösseres Problem dar. Es wurde ersichtlich, dass Android bis und mit der aktuellsten Version nicht in der Lage ist, eine Adhoc WLAN Verbindung aufzubauen. Nach kurzer Recherche im Internet wurde diese Erkenntnis bestätigt[4]. Es ist jedoch von einer Umgehungslösung durch das Rooten des Gerätes die Rede, dies ist jedoch kein Ansatz für unsere aktuelle Arbeit. In Rücksprache mit Herr Peter Heinzmann und Omid Afshari wurde beschlossen, dass die WiFi Unterstützung von EcoHelper für Android in dieser Version nicht implementiert werden soll.

4.2.8. Bluetooth

Ziel

Durch die Analyse der Bluetooth-Schnittstelle wurde geprüft, wie die Kommunikation zwischen OBD und Smartphone via Bluetooth aufgebaut werden kann.

Ergebnis

Der Grundgedanke hinter Bluetooth ist, dass man die beiden Geräte, welche miteinander kommunizieren wollen zuerst paaren muss. Sobald das die Paarung erfolgreich war kann der Client (das Smartphone) mit dem Server (der OBD-Adapter) eine Verbindung aufbauen.

Unter Android kann man wie folgt auf den Bluetooth-Adapter zugreifen:

```
BluetoothAdapter adapter = BluetoothAdapter.getDefaultAdapter();
```

Man erhält so Zugriff auf den Bluetooth-Adapter des Smartphone. Über diesen kann man die gepaarten Geräte auslesen und auf das gewünschte Verbinden.

Unsere Tests mit verschiedenen Geräten haben ergeben, dass HTC teilweise eine fehlerhafte Implementation der Bluetooth-Unterstützung ausgeliefert hat. Diese Problematik kann aber mit entsprechenden Workarounds umgangen werden:

```
// Workaround für HTC-Geräte
// Funktioniert auf allen anderen Geräten ebenfalls
String name = "createRfcommSocketToServiceRecord";
Method method = device.getClass().getMethod(name, new Class[]{UUID.class});
BluetoothSocket socket = (BluetoothSocket)method.invoke(device,
UUID.fromString(wellKnownBluetoothDeviceUUID));
UUID uuid = UUID.fromString(wellKnownBluetoothDeviceUUID);

// Normale Implementation
BluetoothSocket socket = device.createRfcommSocketToServiceRecord(uuid);
```

4.3. Tripdefinition

Das Ziel dieser Analyse war, den Zusammenhang von Trips, Fahrzeugen, Fahrern und Mobiltelefonen zu untersuchen. Des Weiteren wurde geschaut, wann ein Trip startet bzw. endet und wie dies im Falle eines Zwischenhaltes aussieht.

4.3.1. Trip – Fahrer

[4] http://code.google.com/p/android/issues/detail?id=82&cnum=500&cstart=175

1. Variante Ein Fahrer gehört zu einem Trip, bei Fahrerwechsel wird der aktuelle Trip beendet und ein neuer Trip gestartet.

2. Variante Einem Trip können mehrere Fahrer zugewiesen werden, bei Fahrerwechsel muss dies dem System mitgeteilt werden.

3. Variante Ein Trip hat einen "fixen" Fahrer, bei Fahrerwechsel wird der Trip ohne Änderungen für das System fortgeführt.

4.3.2. Trip – Fahrzeug

1. Variante Ein Trip wird von Anfang bis Ende mit demselben Auto durchgeführt.

2. Variante Ein Autowechsel während einem Trip kann dem System mitgeteilt werden.

3. Variante Zu einem Trip können mehrere Fahrzeuge gleichzeitig gehören, dies wäre der Fall, wenn z.B. eine Gruppe mit zwei Autos in die Ferien fährt und ihre Daten als gemeinsamen Trip betrachten möchten.

4.3.3. Trip – Smartphone

1. Variante Ein Trip wird von Anfang bis Ende mit demselben Smartphone aufgezeichnet.

2. Variante Während dem Trip fällt das Smartphone aus, um die Aufzeichnung weiterzuführen, wird der Trip an der Abbruchstelle mit einem neuen Smartphone weitergeführt.

3. Variante Zu einem Trip können gleichzeitig mehreren Smartphones gehören, wenn es sich um die 3. Variante von Trip-Fahrzeug Beziehung handelt.

4.3.4. Fahrer – Fahrzeug

1. Variante Ein Fahrer benutzt immer dasselbe Fahrzeug.

2. Variante Ein Fahrer kann verschiedene Fahrzeuge benutzen. Dies hat zur Folge, dass der Fahrer sein aktuelles Fahrzeug auswählen muss.

4.3.5. Fahrer – Smartphone

1. Variante Ein Fahrer benutzt immer dasselbe Smartphone.

2. Variante Ein Fahrer benutzt verschiedene Smartphones für die Aufzeichnungen. Hier muss sich der Fahrer bewusst sein, dass lokale Daten nicht auf den unterschiedlichen Smartphones verfügbar sind.

4.3.6. Wann/Wie wird ein Trip gestartet bzw. beendet?

1. Variante Fahrer startet und stoppt Trip manuell, dazwischen wird der Trip weitergeführt, egal wie viele Pausen es gibt.

2. Variante Fahrer startet Trip manuell. Er besitzt jedoch die Möglichkeit, diesen zu pausieren. Anschliessend kann er den pausierten Trip weiterführen oder beenden.

4.3.7. Beenden von Trip

Ein Trip kann wie folgt beendet werden:

- Beenden der Anwendung (nur durch abschiessen mit Hilfe eines Tasks Managers möglich)
- Herunterfahren des Smartphone
- Manuelles beenden durch Fahrer

4.3.8. Fazit

Der Fokus des Projektes wird durch folgende Punkte festgelegt:

- Ein Trip hat einen "fixen" Fahrer, bei Fahrerwechsel wird der Trip ohne Änderungen für das System fortgeführt. Um einen Fahrerwechsel zu simulieren, muss ein neuer Trip gestartet werden.
- Ein Trip wird von Anfang bis Ende mit demselben Auto durchgeführt.
- Ein Trip wird von Anfang bis Ende mit demselben Smartphone aufgezeichnet.
- Ein Fahrer kann verschiedene Fahrzeuge benutzen. Dies hat zur Folge, dass der Fahrer sein aktuelles Fahrzeug auswählen muss.
- Ein Fahrer benutzt immer dasselbe Smartphone.
- Fahrer startet Trip manuell. Er besitzt jedoch die Möglichkeit, diesen zu pausieren. Anschliessend kann er den pausierten Trip weiterführen oder beenden

5. Realisiertes System

5.1. Einführung

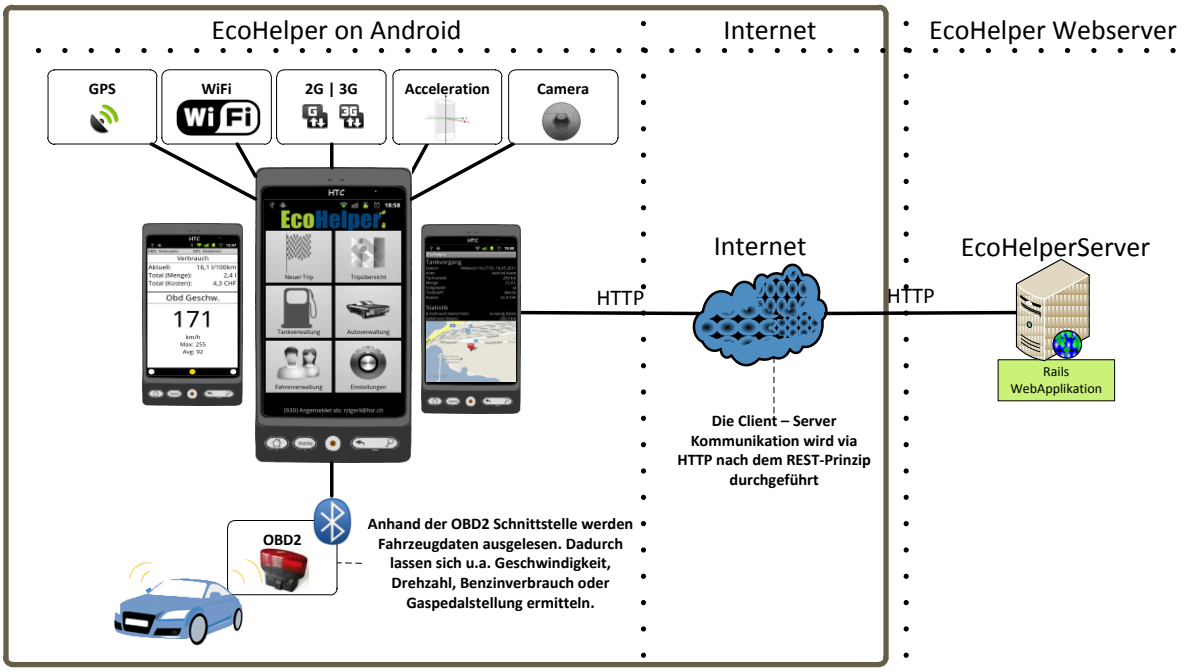


Abbildung 15: Systemübersicht EcoHelper für Android

Die wichtigsten Elemente, die Systemgrenzen und die Systemumgebung zu der in dieser Bachelorarbeit entwickelten EcoHelper Anwendung sind in „Abbildung 15: Systemübersicht EcoHelper für Android“ dargestellt. Die realisierte Anwendung wird auf einem Android Smartphone ausgeführt. Der EcoHelper zeichnet anhand der OBD2-Schnittstelle des Autos via Bluetooth Fahrzeugdaten auf. Daraus lässt sich u.a. die Geschwindigkeit, die Drehzahl, der Benzinverbrauch oder die Gaspedalstellung ermitteln. Weiter erfasst die Anwendung Lokationsdaten mittels GPS und Mobilfunk- / WLAN Informationen, sowie Beschleunigungswerte durch die eingebauten Beschleunigungssensoren. Alle Daten werden lokal auf dem Smartphone gespeichert. Anschliessend werden die Daten für eine genauere Analyse über http im JavaScript Object Notation (JSON) Format an den durch cnlab entwickelten EcoHelper-Server übertragen. Die Daten liefern Informationen über die Fahrweise und den Verbrauch des Benutzers.

5.2. Architektur

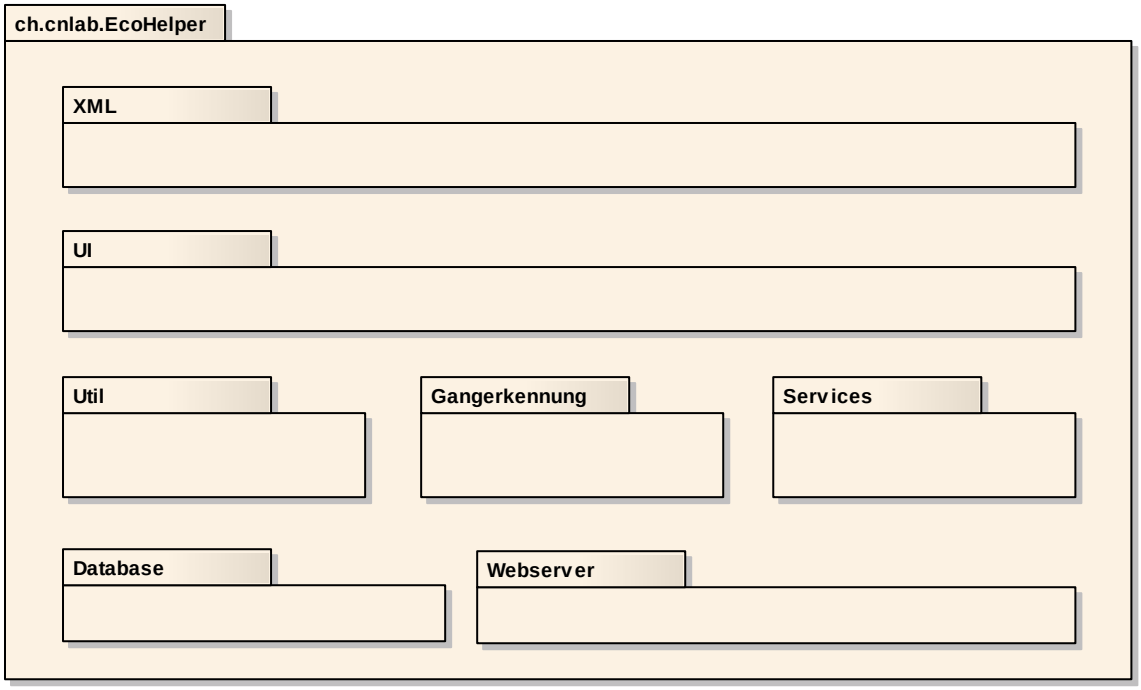


Abbildung 16: Architekturübersicht

Das Package-Diagramm zeigt die wesentlichen Komponenten der EcoHelper-Architektur. Dabei sind folgende Komponenten wichtig:

- XML: Definition der grafischen Oberfläche in XML
- UI: alle Activities der Anwendung
- Util: diverse Hilfsklassen
- Gangerkennung: Logik zur Gangerkennung
- Services: Der Service, welcher während eines Trips im Hintergrund gestartet wird und ständig das UI mit Informationen versorgt.
- Database: Die ganze Kommunikation mit der SQLite-Datenbank.
- Webserver: Ist für die Kommunikation mit dem Webserver zuständig.

In den folgenden Abschnitten werden wichtige Komponenten erklärt und zum Teil mit Beispielen deren Verwendung demonstriert. Dies dient hauptsächlich dem Weiterentwickler und soll demjenigen helfen, sich einfach im Programmcode zurechtzufinden.

5.3. Client-Server-Kommunikation

EcoHelper funktioniert nur zusammen mit der von HSR/cnlab durch Omid Afshari entwickelten Rails-Serverapplikation. Deshalb wird in diesem Abschnitt auf die wesentlichen Komponenten der Kommunikationsschnittstelle eingegangen. Der Fokus liegt aber ganz klar auf der Clientseite.

Die Serverapplikation wurde mit dem Ruby on Rails-Framework erstellt. Der Client hingegen arbeitet mit Java. Für die Kommunikation zwischen diesen zwei Technologien wurde das Hyper-Text-Transfer-Protokoll (http-Protokoll) verwendet. Um die Schnittstellen auf der Serverseite einfach zu halten, wurde der Softwarearchitekturstil Representational State Transfer (REST) eingesetzt. Somit wurde auch auf eine zusätzliche Transportschicht, wie zum Beispiel SOAP, welche die Architektur schwerer

machen würde, verzichtet. Für eine detaillierte Erklärung und Beschreibung von REST wird an dieser Stelle auf den REST-Wikipedia-Artikel[5] und die Dissertation von Roy Fielding[6] verwiesen.

Wegen seines kompakten Formates und seiner weiten Verbreitung wurde zudem JavaScript Object Notation (JSON) als technologieunabhängiges Datenaustauschformates gewählt. Somit sind alle Nachrichten, welche zwischen dem Server und dem Client ausgetauscht werden, im JSON-Format.

Alle Nachrichten stellen auf der Clientseite ein Java-Objekt dar. All diese Nachrichten-Objekte befinden sich im Package Webserver/Messages. Für die Serialisierung und Deserialisierung der Java-Objekte in das JSON-Format wird die Gson-Library eingesetzt. Sie ist äusserst effizient, schnell und kann auch mit grossen Datenmengen umgehen.

Um effizient und einfacher mit dem Server zu kommunizieren, wurde eine wichtige Klasse erstellt - die RequestTask-Klasse. Anhand des Login-Vrogangs soll deren Verwendung und Funktionsweise erläutert werden.

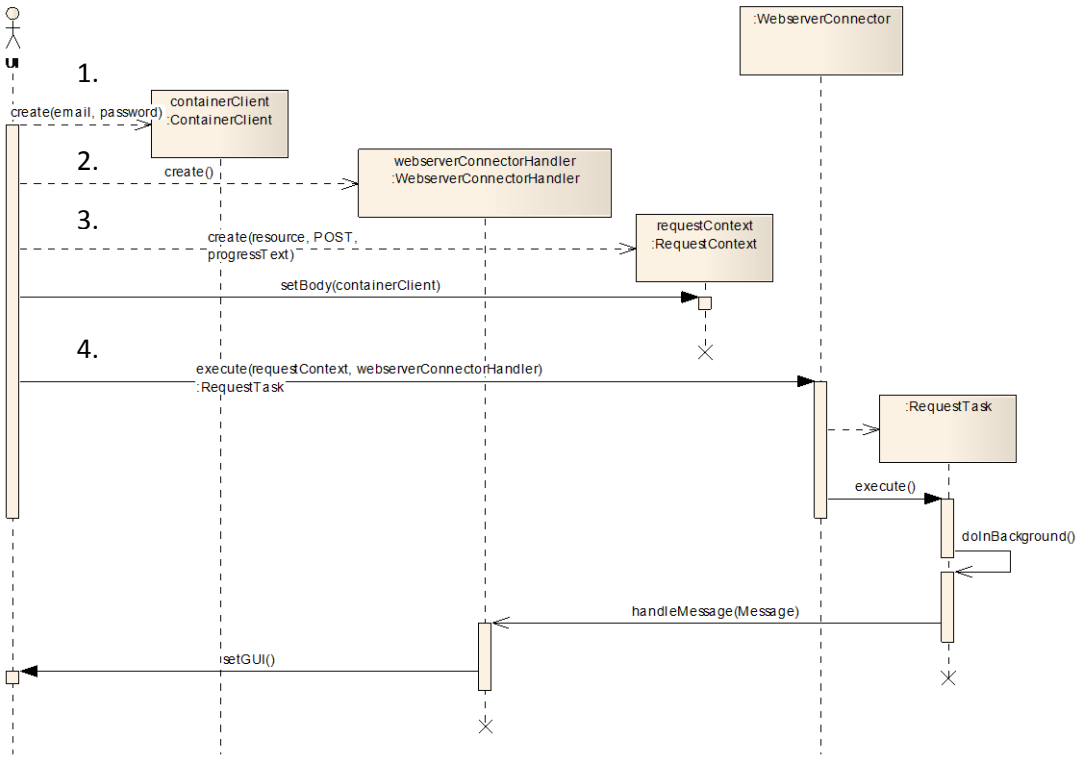


Abbildung 17: Ablauf Login

Das obige Sequenz-Diagramm zeigt die wichtigsten Punkte einer Anfrage. Alle Arten von Anfragen (PUT, POST, DELETE, GET) laufen nach dem gleichen Prinzip ab. Hier wird gezeigt, was bei einem Login passiert. Die folgende Nummerierung bezieht sich auf die Nummerierung im obigen Diagramm.

1. Der Initiator, meist das Grafical User Interface (GUI) beziehungsweise der Benutzer startet eine Anfrage. Dafür muss ein Objekt erstellt werden, welches der Server-Schnittstelle entspricht. Hier im

[5] http://de.wikipedia.org/wiki/Representational_State_Transfer
[6] http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Beispiel wurde dafür ein Objekt der Klasse ContainerClient verwendet. Dieses Objekt besitzt hier einfachheitshalber nur zwei Attribute – email und password.

2. Danach muss ein WebserverConnectorHandler erstellt und definiert werden. Innerhalb dieses Handlers wird die Antwort verarbeitet. Der Grund für diese Aufteilung der Verarbeitung wird später ersichtlich.

3. Zudem muss ein RequestContext erstellt werden, dieser enthält die Nachricht und den Antwortverarbeiter (WebserverConnectorHandler).

4. Diese zwei Objekte können dann dem WebserverConnector übergeben werden. Dort erstellt der WebserverConnector einen RequestTask. Das ist ein AsyncTask[7]. Wie der Name schon sagt, wird hier die Anfrage asynchron in einem eigenen Thread ausgeführt. Dies hat den Vorteil, dass das GUI nicht blockiert und nicht lange auf die Antwort warten muss. Jedoch besteht jetzt das Problem, dass der asynchrone RequestTask nicht mehr auf Elemente des GUI-Threads zugreifen darf. Dies wurde durch den vorhin erwähnten WebserverConnectorHandler umgangen. Da dieser Handler[8] innerhalb des GUI-Threads erstellt wurde kann er immer noch auf Elemente des GUIs zugreifen. Deshalb sendet der RequestTask die erhaltene Antwort an den mitgegebenen WebserverConnectorHandler und dieser kann dann entsprechend auf die Antwort reagieren.

[7] http://developer.android.com/resources/articles/painless-threading.html
[8] http://developer.android.com/reference/android/os/Handler.html

5.4. User Interface (UI)

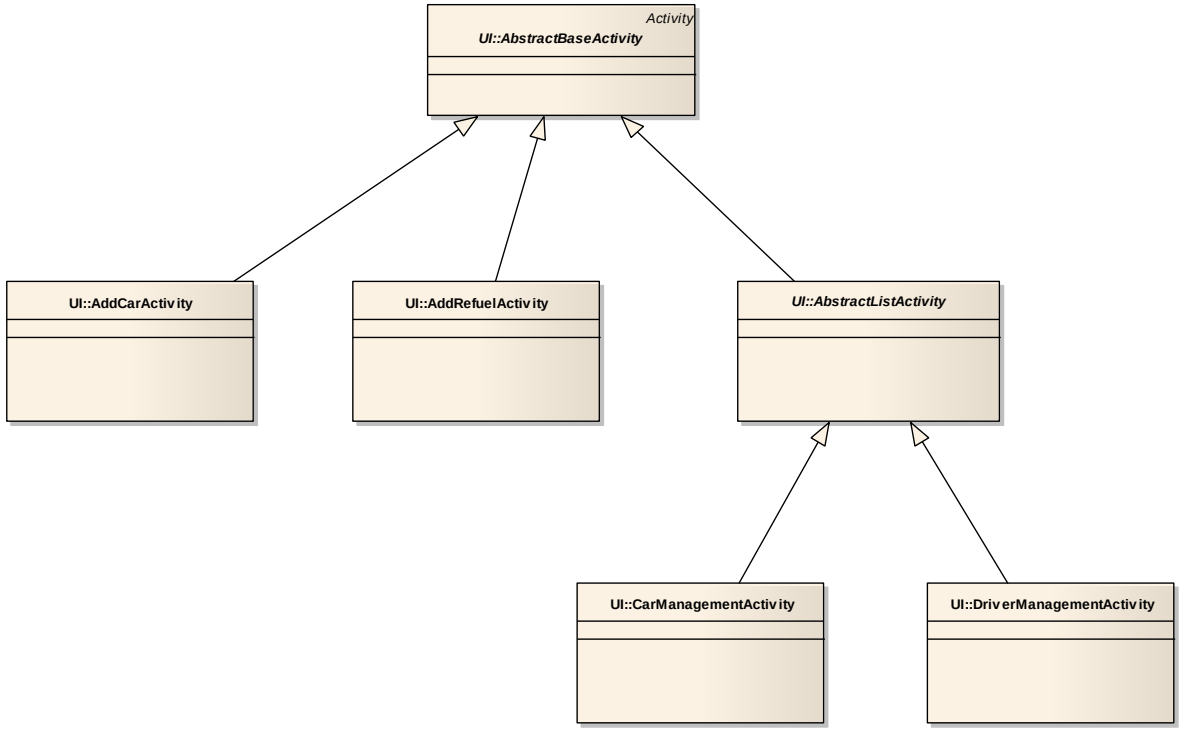


Abbildung 18: GUI-Architektur

Das obige Klassendiagramm zeigt, dass alle Activities, hier im Diagramm nur exemplarisch die AddCarActivity und die AddRefuelActivity, die AbstractBaseActivity erweitern. Diese Basis Activity stellt verschiedene Hilfsmethoden, wie zum Beispiel den Zugriff auf die Datenbankschnittstelle zur Verfügung.

Zusätzlich erweitern alle ListActivities (CarManagementActivity, DriverManagementActivity, …) die AbstractListActivity. Diese Klasse ist zuständig für die ganze Synchronisation der entsprechenden Listeneinträge und stellt eine Art von Objekten in einer Liste dar. Zudem müssen dann verschiedene Funktionalitäten, wie zum Beispiel das Löschen, von der konkreten Klasse implementiert werden. Diese Abstraktion wurde gemacht, weil das Synchronisieren grundsätzlich immer gleich abläuft und somit müssen nur die Unterschiede speziell behandelt respektive implementiert werden. Erwähnenswert ist noch, dass der AbstractListActivity, über die generische Schnittstellen, Abhängigkeiten angegeben werden können. Eine solche Abhängigkeit sind bei der RefuelManagementActivity die Autos. Tankvorgänge können erst erfasst werden, wenn der Benutzer ein Auto erfasst hat. Wird, zum Beispiel, die Anwendung auf einem neuen Gerät installiert und der Benutzer wechselt direkt nach dem Login zur Tankverwaltung, wären die Autos noch nicht synchronisiert. Dies würde dazu führen, dass der Benutzer zuerst zur Autoverwaltung wechseln müsste. Da aber mit Hilfe dieser Abhängigkeiten die Autos auch noch gleichzeitig synchronisiert werden, sind diese verfügbar und der Benutzer kann seine Tankvorgänge ansehen und sofort einen neuen Tankvorgang erfassen.

5.4.1. Google Map View

Die EcoHelper-Anwendung nutzt die Google Map View. Dafür wird ein Application Programming Interface (API) Key von Google verwendet[9]. Dieser Key ist nur für genau ein Android-Keystore-File gültig. Über dieses Keystore-File wird jede Anwendung signiert. Damit nun sämtliche Entwickler Zugriff auf Google Maps haben und sich nicht jeder einen eigenen Google Map Key erstellen muss wurde ein gemeinsamer Keystore erstellt und im Projektordner abgelegt. Das Passwort für den Zugriff auf diesen Keystore lautet „android“. Des Weiteren ist das Passwort für den Alias „androiddebugkey“ ebenfalls „android“.
Weitere Informationen zum Signieren, beziehungsweise Exportieren und Erstellen von Android-Anwendungen sind zu finden unter:
http://developer.android.com/guide/publishing/app-signing.html

Falls die EcoHelper- Anwendung mit einem anderen Keystore-File und Google Map API Key erstellt werden möchte, müssen folgende Schritte durchgeführt werden:

1. In den Eclipse Einstellungen den gewünschten „Custom debug Keystore“ auswählen.

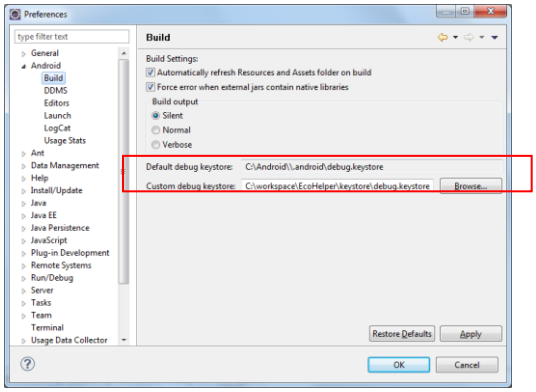


Abbildung 19: Eclipse Custom Keystore angeben

Beschreibung zur obigen Abbildung: Der rote Rahmen zeigt, wo der Pfad zum Custom Keystore anzupassen ist.

2. Im Workspace des Android EcoHelper-Projektes den zum Keystore passenden Google Map API Key in der Datei „strings.xml“ ändern.

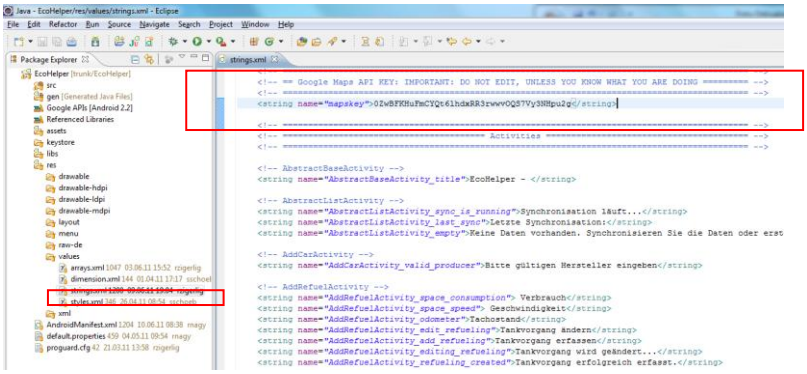


Abbildung 20: Eclipse Google Maps API Key anpassen

Beschreibung zur obigen Abbildung: Die grössere Markierung zeigt die Stelle in der Datei strings.xml, an welcher der Google Map API Key zu ändern ist.

[9] http://developer.android.com/resources/tutorials/views/hello-mapview.html

5.5. Datenaufzeichnung

Der EcoHelper soll während des Trips Daten aus diversen Quellen auslesen und je nach Modus direkt zum Server senden oder einfach in die Datenbank für den späteren Upload speichern. Folgende Datenquellen sind verfügbar:

- Smartphone externe
  - OBD via Bluetooth
  - OBD via WiFi
- Smartphone interne
  - Mobileinformationen
  - Ortungsinformationen
  - Beschleunigungssensoren

Sämtliche Daten sollen während dem Trip im Hintergrund aufgezeichnet werden. Ob die Daten gleichzeitig auch auf der Oberfläche dargestellt werden, ist dabei nebensächlich.

Zur Abwicklung im Hintergrund wurde ein Android-Service eingesetzt. Dieser wurde DataCaptureService benannt und stellt sämtliche Funktionalitäten zur Aufzeichnung eines Trips zur Verfügung.

5.5.1. Aufzeichnungskonzept

Der EcoHelper kennt den Offline sowie den Online-Modus. Diese unterscheiden sich lediglich im Punkt, dass im Online-Modus die Daten so schnell wie möglich zum Server gesendet werden. Da aber ein Smartphone während eines Trips nicht immer eine Internetverbindung haben muss, benötigen wir dennoch eine Art Buffer.

Zu Beginn sind beide Modus genau gleich, es wird immer in fixen Zeitabständen ein Schnappschuss der aktuellen Daten aller Sensoren in die Datenbank gespeichert. Im Online-Modus wird dann ebenfalls in fixen Zeitabständen der Datenbestand aus der Datenbank an den Server gesendet. Im Offline-Mode bleiben diese n der Datenbank, bis der Benutzer nach dem Trip die Daten manuell zum Server sendet.

5.5.2. DataCaptureService

Der DataCaptureService hat die folgenden Aufgaben:

- Starten/stoppen sämtlicher Messgeräten
- Überwachen der Messgeräte
- Oberfläche über abonnierte Datenänderungen informieren
- Speichern der Daten in der Datenbank
- Upload der Daten im Online-Modus

Um alle diese Ziele zu erreichen wurde die folgende Klassenstruktur implementiert:

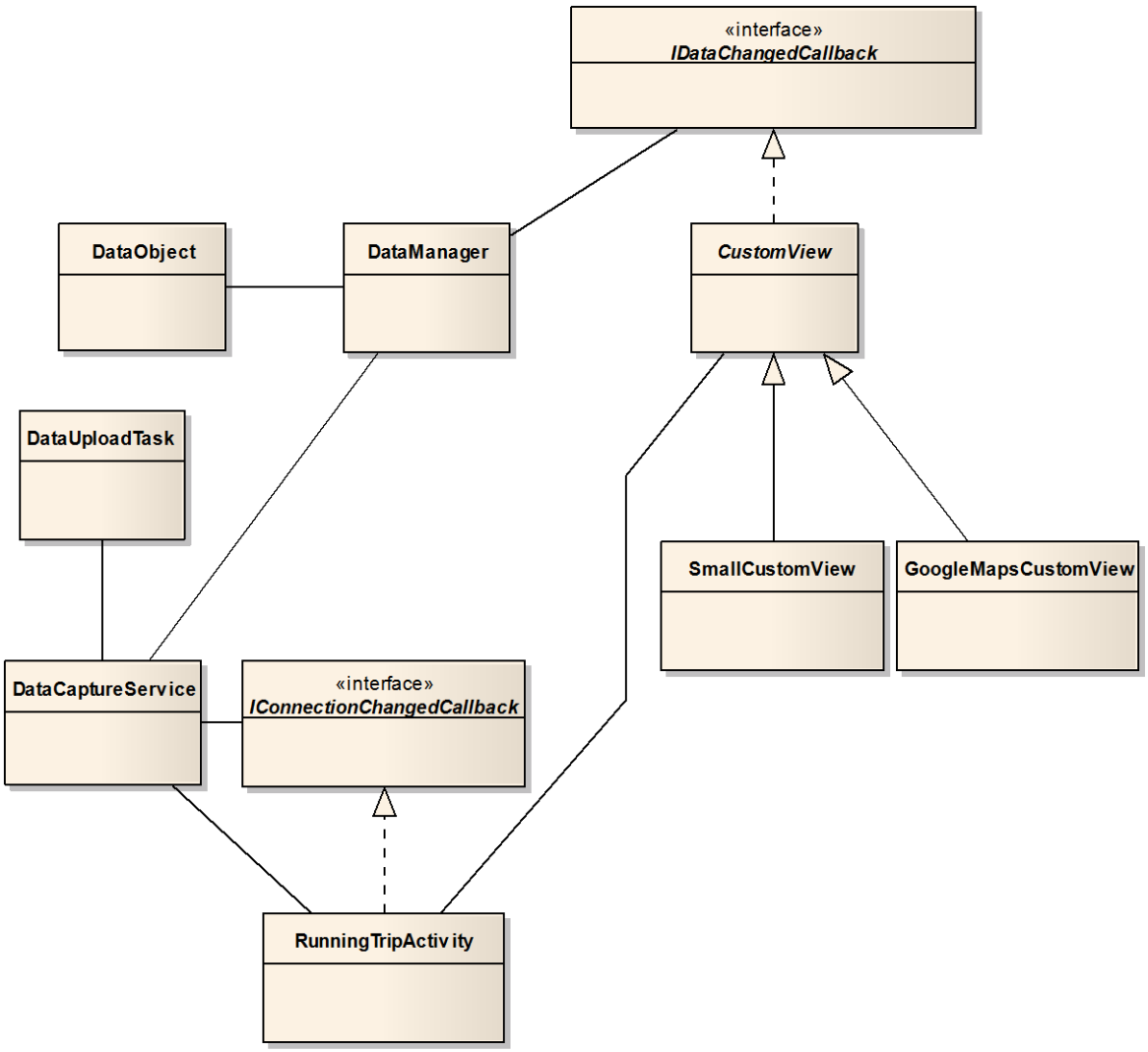


Abbildung 21: Architektur Datenaufzeichnung

DataCaptureService

Wie bereits beschrieben, ist der DataCaptureService die koordinierende Instanz. Er liest sämtliche Daten aus den Sensoren aus und handelt die entsprechenden Verbindungen. Wie die Sensoren angesprochen werden, ist den entsprechenden Folgeabschnitten dokumentiert.

DataManager

Diese Klasse ist zuständig für die aktuellen Daten. Bei Änderungen der Daten werden gegebenenfalls betroffene generische Parameter wie z.B. der aktuelle Gang oder Verbrauch automatisch durch den DataManager neu berechnet. Weiter bietet er einfachen Zugriff auf Minimal, Maximal und Durchschnittswerte der einzelnen Parameter.

Intern verwendet der DataManager für jeden Parameter ein DataObject welches in sich zuständig für die Berechnung der Minimal, Maximal und Durchschnitts-Werte ist.

Sämtliche Parameter-Werte werden immer als Doubles behandelt.

Komponenten welche über Änderungen der Daten informiert werden möchten, müssen das IDataChangedCallback-Interface implementieren. Somit können sie sich beim DataManager für die

gewünschten Parameter registrieren. Sobald der entsprechende Parameter durch einen Sensor ändert, ruft der DataManager sämtliche für diesen Parameter registrierten Komponenten, auf.

DataUploadTask
Der DataUploadTask wird während dem Trip verwendet, wenn der Online-Modus aktiviert ist. Im Offline-Modus wird der Upload durch die TripDetailActivity gestartet, wobei diese ebenfalls diesen DataUploadTask verwendet.

Implementiert wurde diese Klasse als Android AsyncTask. D.h. der Upload läuft in einem eigenen Thread. Während des Trips läuft dieser in einer Endlosschlaufe, wobei er in einem gesetzten Zeitintervall (20 Sekunden) die Daten aus der Datenbank zum Server sendet.

RunningTripActivity / CustomView
Die RunningTripActivity sieht der Benutzer während des laufenden Trips. Er kann darauf verschiedene CustomViews hinzufügen, welche die Informationen des Trips (Sensordaten, Obd-Daten, grafische Darstellungen, etc.) darstellen.

Jede CustomView muss das IDataChangedCallback-Interface implementieren, um sich beim DataManger für Daten-Updates registrieren zu können.

IConnectionChangedCallback
Stellt der DataCaptureService fest, dass eine Verbindung zu einem Messgerät (vorwiegend passiert dies bei GPS oder der OBD2-Schnittstelle) den Status geändert hat (z.B. könnte die Verbindung verloren gehen), muss die Oberfläche informiert werden. Dazu muss diese das IConnectionChangedCallback-Interface implementieren und sich beim DataCaptureService für das entsprechende Update registrieren.

5.5.3. Parametersystem
Aus den zahlreichen Datenquellen werden eine grosse Anzahl an Parametern mit unterschiedlichen Masseinheiten gesammelt. Gewisse Parameter, z.B. ein Gang, besitzen keine Masseinheit. Bei einigen Parametern besteht die Möglichkeit, über die Einstellungen die Masseinehit z.B. von „Meter“ in „Meilen“ zu ändern.

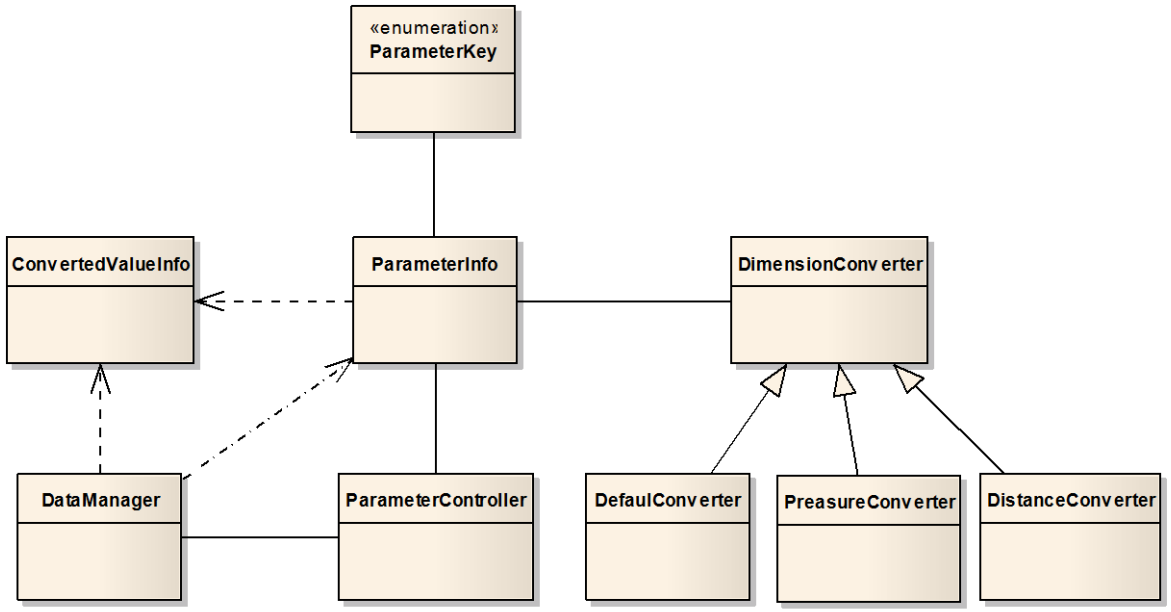


Abbildung 22: Architektur Parametersystem

In diesem Beispiel wird die Benützung des ParameterSystems durch den DataManager gezeigt. Jede andere Klasse könnte genau gleich auf die Parameterinformationen zugreifen. Zu beachten ist, dass es sich hier nicht um die Werte der Parameter handelt, sondern nur um Metadaten zu diesen.

ParameterController

Klasse, welche sämtliche Metainformationen zu den vorhandenen Parametern kennt und Auskunft über jeden geben kann. Die Informationen werden nur einmal initialisiert und anschliessend bei jeder Verwendung wiederverwertet.

ParameterInfo

Beschreibt einen einzelnen Parameter. Dazu gehören folgende Informationen:

- Verwendeter Converter
- ParameterKey zur eindeutigen Identifikation
- Boolean, ob der Parameter über den ganzen Trip berechnet wird oder nur einen aktuellen Messwert einer Datenquelle darstellt
- Keys zu den Ressourcen für Name, ShortName und Beschreibungstext
- Abhängigkeit des Parameters. z.B. ob der Gang-Parameter von Rpm und Geschwindigkeit abhängig ist.
- PID bei OBD2-Parameter
- Priorität, um bei OBD-Parametern die Abfragehäufigkeit festzulegen

Weiter bietet die Klasse Funktionen zum Konvertieren von Werten in das durch die ParameterInfo definierte Format.

ConvertedValueInfo

Eine reine Datenklasse, welche Informationen nach einer Konvertierung beinhaltet. Dazu gehören:

- Wert vor Konvertierung
- Wert nach Konvertierung
- Formatierter und konvertierter Wert
- Dimension als String des konvertierten Wertes

DimensionConverter

Für jeden Parameter kann ein Converter angegeben werden. Für alle die keine speziellen Anforderungen haben, wurde ein DefaultConverter implementiert. Die entsprechenden Converter rechnen den übergebenen Wert, je nach Anwendungs-Einstellung entsprechend, um. Weiter wird der konvertierte Wert je nach Einstellung des Konverters formatiert.

5.6. Schnittstellen

5.6.1. OBD2-Schnittstelle

Die OBD2-Schnittstelle ist die Hauptquelle für die EcoHelper-Anwendung. Diese ermöglicht die Kommunikation mit dem Auto. Dabei muss das Smartphone mit der OBD2-Schnittstelle des Autos verbunden werden. Dafür existieren verschiedene Adapter, welche eingesetzt werden können. Diese stellen die Verbindung via Bluetooth oder WiFi her.

Abfragesyntax Allgemein

Ist man mit der OBD2-Schnittstelle verbunden, können über Text-Kommandos Informationen abgefragt werden.

Der Standard (ISO-Norm 15031-6) definiert dafür verschiedene sogenannte „Modes“, welche für verschiedene Informationen/Operationen verwendet werden können:

Mode	Beschreibung
01	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non CAN only)
06	Test results, other component/system monitoring (oxygen sensor monitoring for CAN only)
07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
08	Control operation of on-board component/system
09	Request vehicle information

Tabelle 5: OBD Modes

Hinweis: Tabelle wurde von http://en.wikipedia.org/wiki/OBD-II_PIDs übernommen.

Zu jedem Mode können verschiedene Informationen abgefragt werden. Dafür gibt es die sogenannten Parameter-IDs, kurz PID genannt.

Beispiel:
Es kann mit Mode 01 und PID 0D die aktuelle Geschwindigkeit des Autos ausgelesen werden.

Antwort auf eine Anfrage

Die Antwort auf eine Anfrage ist wie folgt aufgebaut:

```
[ECHO]
[STATE] [COMMAND] [AA] [BB] [CC] [DD] [EE]
```

Folgende Tabelle erklärt die Bytes welche die Antwort enthält:

Platzhalter	Bedeutung
ECHO	Echo des abgesendeten Kommandos. Sendet man 01 0D, kommt hier 01 0D zurück
STATE	Status der Antwort, 7F heisst, dass die abgefragte PID bei diesem Mode nicht unterstützt wird. Falls die Abfrage geklappt hat, wird 4 + der Mode zurückgegeben. z.B. eine Abfrage im Mode 1 ergibt 41
COMMAND	PID die abgefragt wurde
AA, BB, CC, DD, EE	Fünf Datenbytes, welche die Antwort enthalten. Jedes Byte ist im Hex-Format abgelegt und muss für die Weiterverarbeitung in das Dezimalsystem umgerechnet werden.

Tabelle 6: OBD-Antwort Elemente

Abfrage Fahrzeugdaten

Aus den Daten, welche man aus den Datenbytes erhält, ist eine Formel definiert, um den korrekten Wert zu erhalten. In folgender Auflistung sind sämtliche PIDs mit den entsprechenden Formel abgebildet, welche durch den EcoHelper verwendet werden:

PID	Beschreibung	Formel	Masseinheit
03	Fuel System Status	A	
04	Calculated engine load value	A*100/255	%
05	Engine coolant temperature	A-40	°C
0A	Fuel pressure	A*3	kPa
0B	Intake manifold absolute pressure	A	kPa
0C	Engine RPM	((A*256)+B)/4	Rpm
0D	Geschwindigkeit	A	km/h
0F	Intake air temperature	A-40	°C
10	MAF air flow rate	((A*256)+B) / 100	grams/sec
11	Gaspedalposition	A*100/255	%

Tabelle 7: Unterstützte OBD PID's

Hinweis: Sämtliche hier aufgezeichneten PID's befinden sich im Mode 01.

Beispiel:

Vom Auto soll die aktuelle Umdrehungszahl (RPM → PID 0C) ausgelesen werden

```
Abfrage an Auto:  01 0C
Antwort Auto:     01 0C 41 0C 0F 91
A = 0x0F     →    15
B = 0x91     →    145
Formel:           ((A*256)+B)/4 → ((15*256)+145)/4 → 996 RPM
```

Abfrage Supported PID's

Im Mode 01 kann mit der PID 00 die unterstützten PID's abgefragt werden. Die Antwort kommt im gleichen Format wie eine normale Abfrage eines Parameters. Die Verarbeitung muss wie folgt durchgeführt werden:

Die Datenbytes werden ohne Abstand aneinandergefügt:

```
AA BB CC DD → AABBCCDD
```

Dieser Hex-Wert kann dann in einen binären Wert umgewandelt werden. Als Beispiel wird hier die vom Simulator geleiferte Antwort verwendet:

```
B87BB010 → 10111000011110111011000000010000
```

Bei diesem binären String entspricht nun jede Ziffer einer PID. Dabei heisst 1, dass die PID vom Auto unterstützt wird und 0 entsprechend nicht unterstützt.

Um hier abzufragen, ob RPM unterstützt wird, muss wie folgt vorgegangen werden:

```
PID für RPM:      0C
PID in Dezimal:   0C → 12
Supported PID's:  10111000011110111011000000010000
```

Das heisst, dass das 12te Bit von links sagt, ob dieses Auto RPM unterstützt. In diesem Falle ist die Frage mit Ja zu beantworten.

Abfrage VIN

Im Mode *09* kann mit der PID *02* die VIN des Autos abgefragt werden.

Unter VIN versteht man die „Vehicle Identification Number". Dies ist eine Nummer, mit der Fahrzeuge eindeutig identifiziert werden können.

Bei der VIN-Abfrage werden fünf Zeilen zurückgegeben. Dabei hat jede Zeile das selbe Format wie eine einzelne normale Datenzeile ohne das Echo.

```
49 02 01 00 00 00 41
49 02 02 47 56 2D 4D
49 02 03 49 4E 49 2D
49 02 04 53 49 4D 20
49 02 05 56 31 2E 33
```

Wenn wir jede Zeile einzeln betrachten, ist diese wie Folgt zusammengesetzt:

```
[BESTÄTIGUNG][PID][COUNTER][VIN BYTE 1][VIN BYTE 2][VIN BYTE 3][VIN BYTE 4]
```

Die Bestätigung, PID und Counter gehören nicht zur Vin und müssen aus der Antwort entfernt werden. Der restliche Teil ist die ASCII-Codierte VIN in Hex.

```
41 47 56 2D 4D 49 4E 49 2D 53 49 4D 2056 31 2E 33
```

Implementation

Der EcoHelper soll unabhängig der zugrundeliegenden Verbindung (Bluetooth/WiFi) mit dem Auto kommunizieren können. Dafür haben wir folgende Klassenstruktur implementiert:

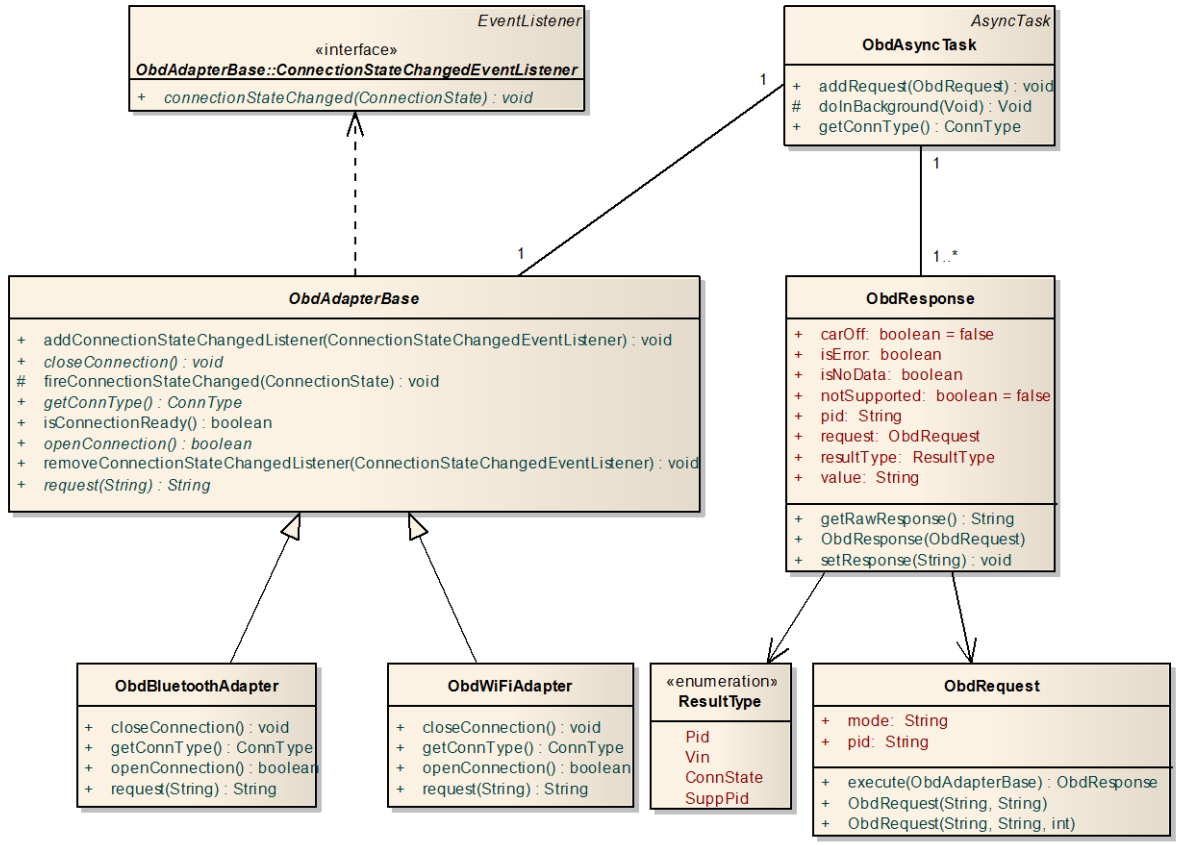


Abbildung 23: Architektur OBD Schnittstelle

Der EcoHelper implementiert in dieser Version nur die Bluetooth-Schnittstelle, wäre aber jederzeit bereit auch mit einem entsprechenden WiFi-Adapter zu arbeiten. Der Grund, warum momentan keine WiFi-Kommunikation unterstützt wird, liegt darin, dass der einziger momentan verfügbare WiFi Obd2-Adapter ein AdHoc-Netzwerk aufbauen möchte. Dies wird zum jetzigen Zeitpunkt von Android nicht unterstützt.

ObdAsyncTask

Der ObdAsyncTask stellt die Schnittstelle für den DataCaptureService zur Verfügung. Er beinhaltet eine Instanz eines ObdAdapterBase, über welche er mit dem Auto kommuniziert. Dabei entscheidet er selbst, welche Schnittstelle er verwenden möchte. In unserem Falle ist dies immer die Bluetooth-Schnittstelle, da der WiFi-Teil nicht implementiert wurde.

ObdResponse / ObdRequest

Der ObdAsyncTask hat eine Liste von ObdRequests, welche er noch an das Auto senden muss. Der Request kann sich dabei selbst auf dem ObdAsyncTask ausführen und generiert dann automatisch die ObdResponse. Diese definiert über das resultType-Property, wie die Antwort geparst wurde.

5.6.2. Beschleunigungssensoren

Durch die Beschleunigungssensoren des Smartphone kann die Beschleunigung in sämtliche Richtungen gemessen werden. Der EcoHelper zeichnet dabei nur die Rohdaten auf, welche so an den Server übermittelt werden. Auf eine Auswertung der Werte wurde aus zeitlichen Gründen verzichtet.

Android bietet für den Zugriff auf die Sensoren einen SensorManager. Bei diesem kann man sich für jegliche Art von Sensoren (Akzelerator, Gyroskop, Kompass, etc.) anmelden. Über die registerListener-Methode kann ein SensorEventListener beim Manager registriert werden, welcher bei Änderungen der Sensordaten entsprechend benachrichtigt wird.

5.6.3. Mobileinformationen

Während eines Trips sollen sämtliche Informationen zum Mobilfunknetz des Fahrers gesammelt werden. Dazu gehören u.a. Signalstärke, Verbindungstyp (UMTS, EDGE, etc.) und OperatorId.

Android bietet für den Zugriff auf Mobilfunknetz-Informationen einen TelephonyManager. Bei diesem registriert der EcoHelper ein PhoneStateListener, welcher über Änderungen der gewünschten Parameter informiert wird.

5.6.4. Ortungsinformationen

Android bietet verschiedene Quellen zur Abfrage von Ortungsinformationen. Diese werden Provider genannt. Verwaltet werden sämtliche Ortungsinformationen durch einen LocationManager. Der EcoHelper registriert sich beim LocationManager für den besten verfügbaren Provider. Dazu wird ein LocationListener implementiert, welcher nebst den Änderungen der Parameter, auch über Wechsel der aktivierten Provider informiert wird. Sollte ein besserer Provider aktiviert werden, wechselt der EcoHelper automatisch zu diesem.

Folgende Provider werden von Android unterstützt und durch den EcoHelper verwendet:

- Network (Positionsbestimmung durch Daten der verfügbaren Sendeantenne sowie WiFi-Access-Points)
- GPS

5.7. Gangerkennung

Der EcoHelper soll während der Fahrt den aktuellen Gang anzeigen. Da die OBD2-Schnittstelle diesen nicht als Wert liefert, muss dieser aus anderen Werten berechnet werden. Namentlich sind dies RPM und Geschwindigkeit.

Wie aus der folgenden Abbildung zu erkennen ist, hat jeder Gang ein gewisses Übersetzungsverhältnis. Der Gangerkennungsalgorithmus soll die in der Abbildung ersichtlichen Ebenen erkennen und dem entsprechenden Gang zuordnen. Wie man sehen kann, ist der kleinste Gang jeweils der tiefste. Je höher das Übersetzungsverhältnis, desto höher ist auch der Gang.

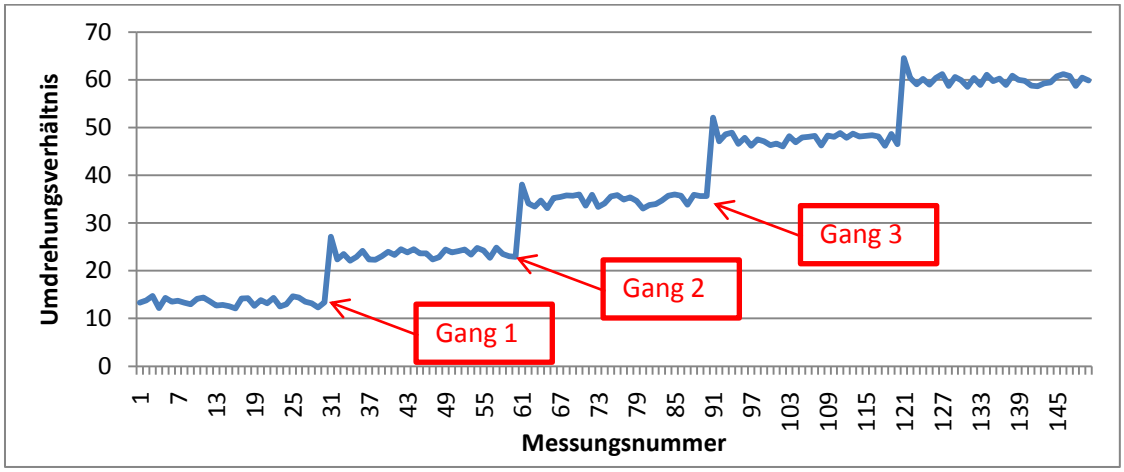


Abbildung 24: Diagramm Umdrehungsverhältnisse während Fahrt

5.7.1. Formel

Zur Berechnung des Übersetzungsverhältnisses wird folgende Formel verwendet:

```
double ratio = (speed * 1000.0 * 100.0 / rpm / 60.0);
```

speed	→	Geschwindigkeit durch OBD2 ausgelesen in km/h
1000	→	km/h in m/h umrechnen
100	→	Geschätzter Radumfang in cm
rpm	→	RPM durch OBD ausgelesen in U/min
60	→	U/min umrechnen in U/sec

Hinweis: *Die Formel wurde nicht selbst hergeleitet sondern vom Symbian EcoHelper-Projekt übernommen. Der Radumfang wurde dabei mit 100cm wesentlich zu klein geschätzt. Da es sich dabei jedoch um einen konstanten Faktor handelt, wurde dieser aus Kompatibilitätsgründen zu früheren EcoHelper-Anwendungen so übernommen.*

5.7.2. Problematik Mittelgangerkennung

Solange der EcoHelper nicht sämtliche Gänge erkannt hat, kann er nicht sagen, welche Gangnummer ein Übersetzungsverhältnis hat. Angenommen der Fahrer beginnt seine Fahrt im 2. Gang so wird dieser als 1. Gang erkannt, da dies der einzige ist, welcher der EcoHelper kennt.

Zur Lösung dieses Problems haben wir aus Testfahrten Durchschnittswerte eruiert, welche meist die entsprechenden Gänge darstellen. Dabei weisen wir den Gängen die folgenden Standard-Übersetzungsverhältnis zu:

Gang	Übersetzungsverhältnis
1.	12.0
2.	22.0
3.	33.0
4.	46.0
5.	58.3

Tabelle 8: Standard Übersetzungsverhältnis

Falls der EcoHelper nun genügend Messungen hat um einen neuen Gang festzulegen, prüft er, ob das Übersetzungsverhältnis einem Standard-Gang zugewiesen werden kann. Falls dies möglich, ist wird die entsprechende Nummer verwendet und der Gang damit abgelegt. Dadurch kann mit einer grossen Anzahl von Autos die Fehlerquote im Lernmodus verkleinert werden. Sollte kein Standardverhältnis passend sein, wird der Gang normal als nächst möglicher eingeordnet.

5.7.3. Problematik Toleranz

Wie im Bild „Abbildung 24: Diagramm Umdrehungsverhältnisse während Fahrt“ zu erkennen ist, gibt es immer wieder Ausschläge und Ungenauigkeiten in den gelieferten Übersetzungsverhältnissen. Damit nicht für jede Ungenauigkeit ein neuer Gang erkannt wird, muss eine gewisse Toleranz in der Erkennung vorhande.

Der erste Ansatz war eine fixe Toleranz, mit welcher jedoch folgende Problematik ergab:

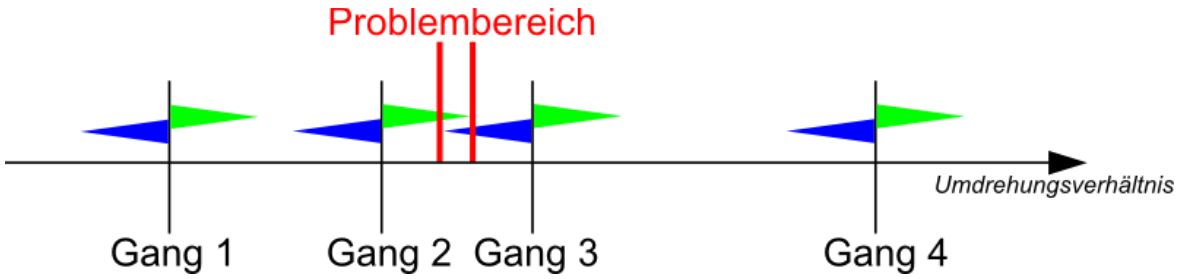


Abbildung 25: Gangerkennung Problematik Toleranz

Wenn zwei Gänge innerhalb der zweifachen Toleranz erkannt werden, ist nicht mehr eindeutig erkennbar, zu welchem Gang das Übersetzungsverhältnis gehört.

Als Lösung für diese Problematik berechnet die EcoHelper Gangerkennung die Toleranz dynamisch. Das heisst, dass wenn zwei Gänge näher sind als die doppelte maximale Toleranz, wird der Unterschied zwischen den Gängen durch zwei geteilt und jeweils eine Hälfte als Toleranz verwendet. Entsprechend dem obigen Beispiel sieht die neue Toleranzberechnung wie folgt aus:

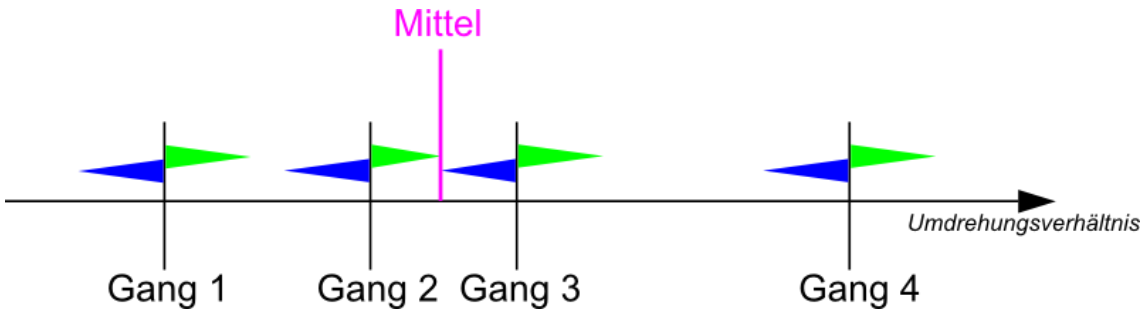


Abbildung 26: Gangerkennung Problematik Toleranz Lösung

Nebst dem Problem der überschneidenden Toleranzen wurde ersichtlich, dass die Ungenauigkeit je nach Auto variieren kann. Entsprechend muss, die Toleranz dem Auto angepasst werden. Es wurde eine Grundtoleranz festgelegt, welche für die meisten Autos ausreicht und sicherlich keine Gänge „verschluckt“. Dies wurde durch zahlreiche Testfahren mit verschiedenen Autos verifiziert.

Die Toleranz wird nun angepasst, sobald für ein Auto zu viele Gänge erkannt wurden. Da bei jedem Auto beim Erfassen die Anzahl Gänge definiert werden muss, ist diese maximale Anzahl bekannt. Wird nun ein Gang zu viel erkannt, wird zuerst der 1. oder 2. Gang zum Rückwärtsgang gemacht. Kommt dann noch ein neuer Gang hinzu, wird die gesamte Gangerkennung zurückgesetzt und die Toleranz um eins erhöht. Dies geschieht solange, bis nicht mehr zu viele Gänge erkannt werden.

5.7.4. Problematik Rückwärtsgang

Der Rückwärtsgang ist nicht von den anderen Gängen zu unterscheiden. Sprich die OBD2-Schnittstelle liefert keine speziellen Informationen, ob der Fahrer sich im Rückwärtsgang befindet.

Entsprechend kann es vorkommen, dass wenn der Fahrer eine gewisse Strecke im Rückwärtsgang zurücklegt, dieser als weiterer Gang erkannt wird und somit die Gangpositionen mit einem höheren Übersetzungsverhältnis fälschlicherweise um eins nach Hinten verschoben werden.

Erfahrungen mit den getesteten Autos zeigen, dass sich das Übersetzungsverhälntis des Rückwärtsgangs immer im Bereich des 1. und 2. Ganges befindet. Es kann dabei sein, dass er sich direkt mit einem der beiden Gänge überschneidet, was eine Erkennung des Rückwärtsganges unmöglich macht. Jedoch tritt in diesem Fall auch keine Falscherkennung auf.

Falls er zwischen den Gängen liegt, erkennt das System diesen erst als weiteren Gang. Solange die Gangerkennung noch nicht abgeschlossen ist, verbleibt dieser so. Sollte der EcoHelper aber nun einen Gang mehr erkennen, als für das Auto definiert ist, wird anhand der Messhäufigkeit der ersten beiden erkannten Gänge der Rückwärtsgang bestimmt. Dies wurde so festgelegt, da davon ausgegangen werden kann, dass der Fahrer häufiger den 1. anstatt den Rückwärtsgang benutzt.

5.7.5. Algorithmus

In diesem Abschnitt soll der gesamte Gangerkennungsalgorithmus in Pseudocode erklärt werden.

```
// Zuletzt abgefragte Ratio
lastRatio

// Anzahl unbekannter Ratios in gleichem Toleranzbereich
anzahlUnbekannteRatios

// Anzahl Messungen von unbekannten Ratios bis diese
// als neuer Gang erkannt wird
minRatiosForNewGear

Funktion getGear(Geschwindigkeit, Rpm)

    ratio = [Berechnung durch Formel]

    // Suche ob ratio zu bekanntem Gang gehört
    Für jeden bekannten Gang
            ist ratio in Toleranzbereich des Ganges
                return Gang ist bekannt, gibt Gangnummer zurück

    // Versuchen ratio als neuer Gang zu erkennen
    ist ratio im gleichen Toleranzbereich wie lastRatio
        anzahlUnbekannteRatios++

        ist anzahlUnbekannteRatios grösser minRatiosForNewGear
            passt ratio zu einem standardGang
                Füge Gang mit Nummer aus Standarderkennung hinzu
            sonst
                Füge Gang mit nächst möglicher Nummer hinzu

    sonst
        anzahlUnbekannteRatios = 0

    lastRatio = ratio

    // Verhindert Ausschläge bei einzelnen Fehlmessungen
    wenn anzahlUnbekannteRatios < 3
        return gib zuletzt erkannten Gang zurück

    return gib <Gang unbekannt> zurück
ende Funktion
```

5.8. Workspace

Während eines Trips wird dem Benutzer der sogenannte Workspace angezeigt. Dabei handelt es sich um drei Fenster, welche mit Elementen gefüllt werden können. Das Prinzip ist vom Android Homescreen übernommen. Per Drag&Drop können Elemente verschoben und vom Fenster entfernt werden. Zwischen den Fenstern kann mit einer seitlichen Finger-Bewegung gewechselt werden.

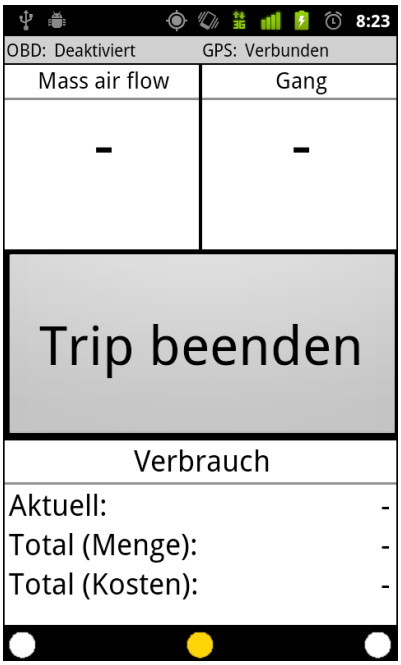


Abbildung 27: Screenshot Workspace

Die Abbildung 27 zeigt ein einzelnes Fenster mit mehreren Elementen. Man erkennt, dass ein Element frei gestaltet werden kann. In diesem Beispiel sind zwei Einzelparameteranzeigen, ein Button sowie eine Parametersammlung zum Verbrauch konfiguriert.

Weiter wurde der Screen in drei Blöcke unterteilt. Im Kopfbereich werden Informationen zur Verbindung mit OBD2 und GPS angezeigt. Der grosse mittlere Bereich bietet Platz für Elemente, welche dynamisch angepasst werden können. Der untere Block zeigt an, auf welchem Fenster sich der Benutzer befindet. In diesem Beispiel wird das mittlere Fenster angezeigt.

5.8.1. Implementation

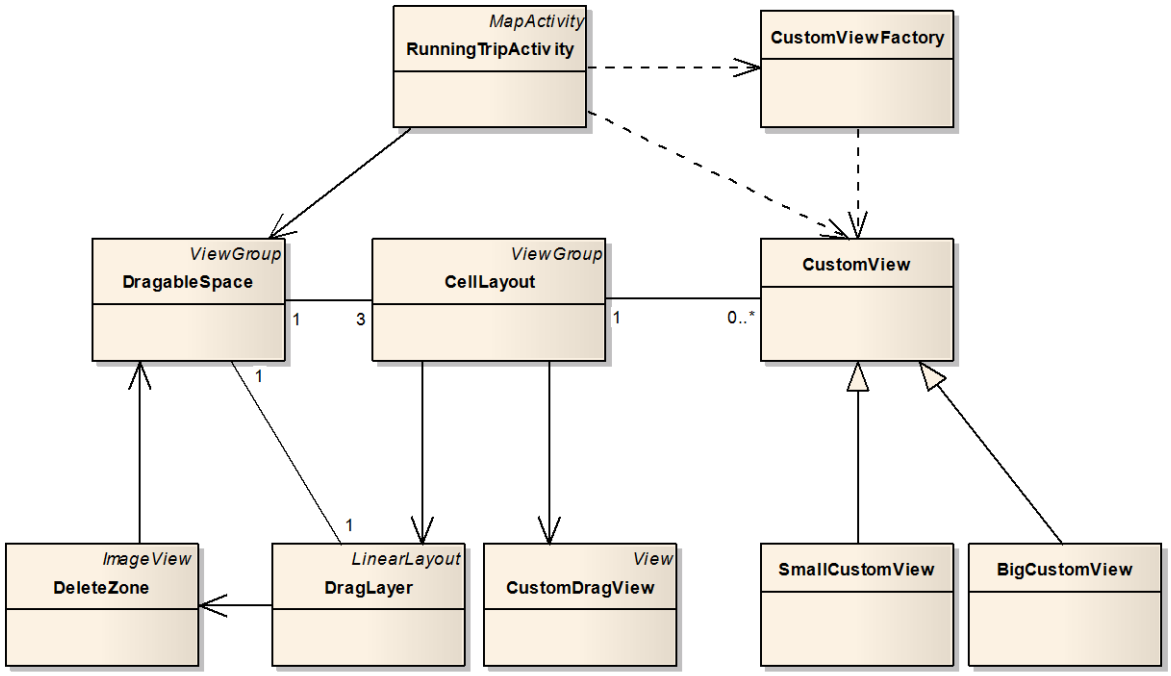


Abbildung 28: Architektur Workspace

RunningTripActivity

Die Android-Activity, welche dem Benutzer angezeigt wird. Sie delegiert hauptsächlich die Benutzerinteraktionen an die entsprechenden Klassen.

DragableSpace

Beherbergt die drei Screens (CellLayout's) und behandelt die Benutzerinteraktionen zum Wechsel zwischen den einzelnen Fenstern.

CellLayout

Entspricht einem einzelnen Fenster. Beherbergt in sich die für das Fenster anzuzeigenden CustomView's. Das CellLayout stellt eine Art Grid zur Verfügung. Dabei teilt es sämtlichen verfügbaren Raum des mittleren Blocks auf dem Bildschirm in ein 2x3-Raster auf.

Abbildung 29: Screenshot CellLayout

CustomView / SmallCustomView / BigCustomView

Ein CellLayout beinhaltet mehrere CustomViews. Eine CustomView kann sich für Informationen des DataCaptureService registrieren. Wenn dann die entsprechenden Messwerte ändern, wird die CustomView automatisch informiert. Die View in sich zeigt die Informationen je nach Wunsch an. Sie selbst kann bestimmen, wie viel Platz (Anzahl X und Y-Zellen) sie im CellLayout benötigt.

DeleteZone

Wenn der Benutzer ein Element per Drag&Drop verschiebt, wird im unteren Block der Papierkorb angezeigt. Wenn der Benutzer eine CustomView über den Papierkorb zieht, wird diese rot hinterlegt. Lässt er in diesem Modus die CustomView los, wird diese vom Fenster gelöscht.

Die DeleteZone repräsentiert diesen Papierkorb.

Abbildung 30: Screenshot Element entfernen

DragLayer

Der DragLayer umfasst den DragableSpace sowie die DeleteZone. Er sorgt dafür, dass im richtigen Moment die DeleteZone ein resp. ausgeblendet wird. Weiter fast der DragLayer den gesamten Bereich zusammen in dem CustomViews per Drag&Drop verschoben werden können.

CustomDragView

Während des Drag&Drop-Vorgangs wird nicht direkt die CustomView angezeigt und verschoben. Es wird eine Art Screenshot der CustomView erstellt welche dem Benutzer während des gesamten Drag&Drop-Vorgangs angezeigt wird. Grund für dieses Verhalten ist, dass das dabei generierte Bitmap einfach gehandelt werden kann und eine Einfärbung, wie sie über der DeleteZone gemacht wird, kein Problem darstellt.

5.9. Datenbank

Jede Android-Anwendung hat eine private SQLite-Datenbank zur Verfügung. Diese liegt im data-Ordner der Anwendung und ist somit durch das Dateisystem vom Zugriff vor Drittanwendungen geschützt. Wenn die Anwendung deinstalliert wird, wird das data-Verzeichnis der Anwendung gelöscht und somit automatisch auch die Datenbank

5.9.1. SQL Datenbank

Der EcoHelper verwendet intern folgendes Datenbankschema:

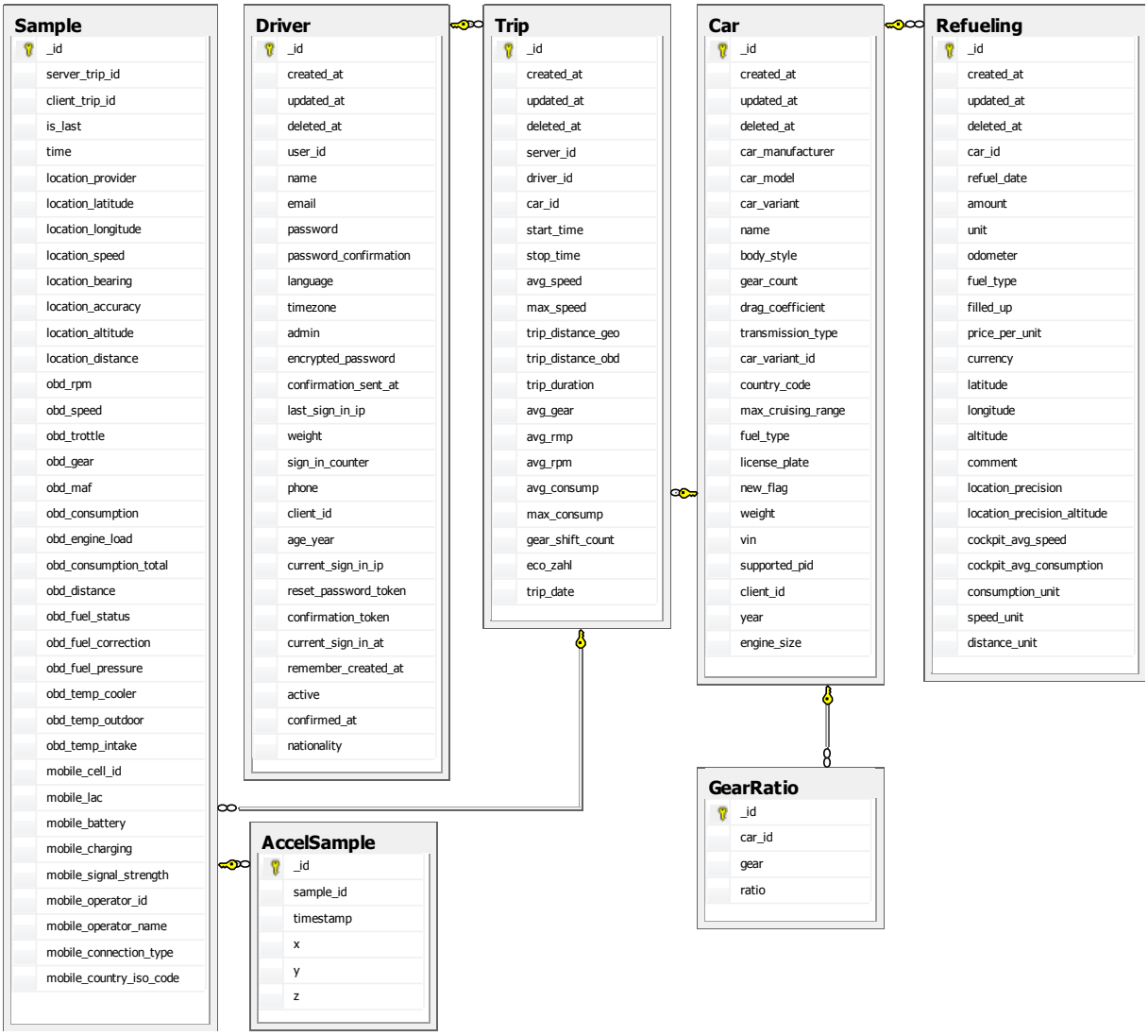


Abbildung 31: Datenbankschema

Sample

Beinhaltet sämtliche Messungen während eines Trips, welche noch nicht an den Server gesendet wurden. Eine Messung gehört immer zu einem Trip.

AccelSample

Da die Beschleunigungssensoren wesentlich häufiger Messungen in der Datenbank speichern, wurden diese in eine eigene Tabelle ausgelagert. Ein AccelSample gehört immer zu einem Sample.

Trip
Speichert sämtliche Trips des aktuell angemeldeten Benutzers. Falls der Trip nur lokal gespeichert ist, ist das Feld server_id null.

Driver
Speichert alle für diesen Benutzer verfügbaren Fahrer. Alle hier vorhandenen Fahrer sind auch auf dem Server vorhanden (Fahrer können nur erfasst werden, wenn Smartphone online ist)

Car
Speichert alle für diesen Benutzer verfügbaren Autos. Alle hier vorhandenen Autos sind auch auf dem Server vorhanden (Autos können nur erfasst werden, wenn Smartphone online ist)

GearRatio
Speichert die Übersetzungsverhältnisse, die zu einem Auto erkannt wurden. Bei jedem Trip werden diese neu in der Datenbank gespeichert. Ein GearRatio gehört immer zu genau einem Auto.

Refueling
Speichert sämtliche Tankvorgänge des aktuell angemeldeten Benutzers. Alle hier vorhandenen Tankvorgänge sind auch auf dem Server vorhanden (Tankvorgänge können nur erfasst werden, wenn Smartphone online ist)

5.9.2. Datenbankzugriff

Um den Zugriff auf die Datenbank aus dem ganzen System möglichst einfach zu gestalten, sowie die Datenintegrität und die Datenaktualität jederzeit zu gewährleisten, wurde die folgende Klassenarchitektur entwickelt:

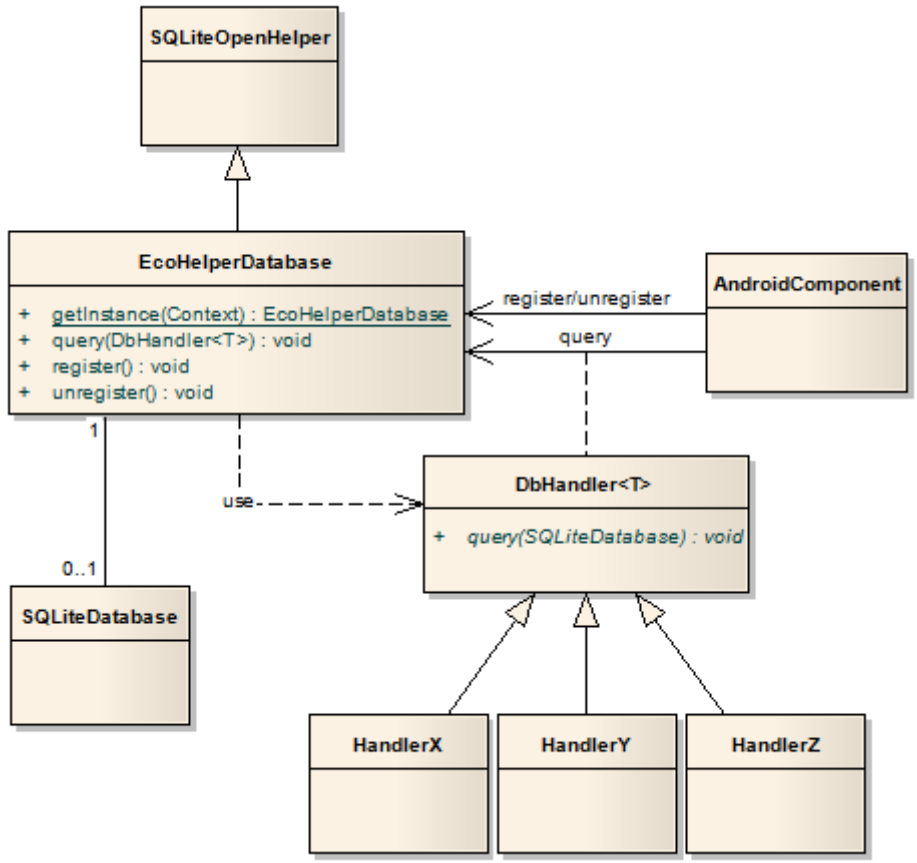


Abbildung 32: Architektur Datenbankzugriff

Dabei handelt es sich nicht um eine Neuentwicklung. Die Idee dieser Architektur wurde bereits während der Studienarbeit GSM Tester entwickelt und ein erstes Mal verwendet.

Das Diagramm wurde bewusst sehr allgemein gehalten, wiederspiegelt jedoch genau das Konzept, welches vom EcoHelper verwendet wird.

SQLiteOpenHelper

Klasse aus dem Android Framework. Diese untersützt die EcoHelperDatabase im Datenbankmanagement betreffend Zugriff auf die Datenbank, sowie durch Funktionen zum einfachen Upgraden bei Versionssprüngen.

SQLiteDatabase

Klasse aus dem Android-Framework, welche direkten Zugriff auf die der Anwendung zur Verfügung stehenden Datenbank liefert. Das Erstellen der Datenbank wird durch den SQLiteOpenHelper durchgeführt.

AndroidComponent

Jede Android-Komponente (Service, Activity), welche Zugriff auf die Datenbank benötigt, muss sich in der onCreate-Methode beim Service registrieren, sowie bei der onDestroy abmelden. Somit kann die EcoHelperDatabase, die Datenbank, sobald diese nicht mehr benötigt wird, automatisch schliessen, bzw. falls sie benötigt wird, öffnen.

Die Komponente kann, sobald sie sich registriert hat, auf der Datenbank beliebige Operationen durchführen. Dazu muss die gewünschte Operation (ein DbHandler) an die EcoHelperDatabase übergeben werden.

EcoHelperDatabase

Die EcoHelperDatabase erbt von SQLiteOpenHelper. Dadurch erhält die Klasse die zwei Methoden onCreate() und onUpgrade(), welche vom SQLiteOpenHelper, je nach aufgefundener Datenbank, automatisch aufgerufen werden.

Weiter wird über die query()-Methode für sämtliche Android-Komponenten eine Schnittstelle zur Datenbank zur Verfügung gestellt, über welche die DbHandler ausgeführt werden können.

DbHandler<T>, HandlerX, HandlerY, HandlerZ

Jede Operation, die auf der Datenbank ausgeführt wird, ist in eine Handler-Klasse gekapselt. Vorteile dieser Lösung:

- Handler können wiederverwendet werden
- Trennung der einzelnen Datenbankoperationen
- Allgemeine Operationen (z.B. Resultat einer Abfrage in ein ContentValues-Objekt speichern) können in der Basisklasse allen Handlern zur Verfügung gestellt werden.

Der Handler wird von der Android-Komponente erstellt und der EcoHelperDatabase.query()-Methode übergeben. Die EcoHelperDatabase selbst ruft dann die query()-Methode des Handlers mit der aktuellen SQLiteDatabase-Instanz auf.

5.9.3. Datenbankabbildung

In Kapitel 0 Datenbankzugriff wurde beschrieben, wie auf die Datenbank zugegriffen wird. In den einzelnen Handlern, sowie für die Delete und Update-Scripts der Tabellen, werden entsprechende SQL-Statements oder zumindest die Spaltennamen benötigt. (Für einfache Statements können die entsprechenden SqliteDatabase-Methoden wie z.B.- insert(..) verwendet werden. Diesen muss Tabellennamen sowie Spalten, Einschränkung, etc. übergeben werden). Damit die Tabellen- sowie Spaltenamen nicht über sämtliche Handler verstreut werden.

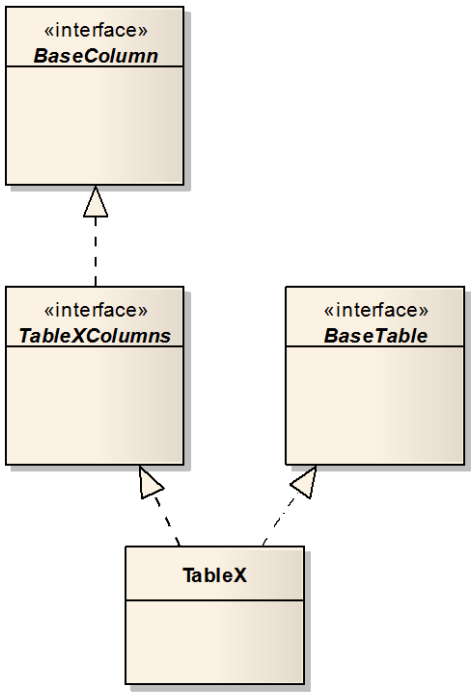


Abbildung 33: Architektur Datenbankabbildung

BaseColumn
Beinhaltet Spalten, welche sämtliche Tabelle aufweisen. Z.B. eine Id-Spalte.

TableXColumns
Beinhaltet sämtliche Spalten der angegebenen Tabelle.

BaseTable
Beinhaltet SQL-Statements (z.B. DROP-Statement), welche sämtliche Tabellen benötigen

TableX
Beinhaltet DROP- und CREATE-Statements für genau diese Tabelle. Weiter wird hier der Tabellen-Name definiert.

5.10. Preferences

Damit der Benutzer die Anwendung seinen Wünschen entsprechend konfigurieren kann, wurde ein Einstellungssystem implementiert. In diesem Kapitel werden die wichtigsten Elemente erläutert. Die benötigten Einstellungen wurden in folgende Kategorien unterteilt:

- Synchronisation
- Autoupload
- Bluetooth Einstellungen
- Ortungs Einstellungen
- Energie Einstellungen
- Masseinheit Einstellungen
- Allgemeine Einstellungen

5.10.1. Synchronisation

Um Trips, Autos, Fahrer und Tankvorgänge automatisch zu synchronisieren, ist diese Option standardmässig aktiviert.

5.10.2. Autoupload

Damit die Messdaten während einem Trip automatisch an den Server übertragen werden, ist diese Option standardmässig aktiviert.

5.10.3. Bluetooth Einstellungen

In den Bluetooth Einstellungen kann die Bluetooth Schnittstelle aktiviert werden. Zusätzlich kann nach verfügbaren Bluetooth Geräten gesucht und verbunden werden. Dies wird für den Verbindungsaufbau zu einem OBD2 Adapter benötigt. Es besteht ebenfalls die Möglichkeit, in die Android Bluetooth Einstellungen zu gelangen.

5.10.4. Ortungs Einstellungen

Die Ortungs Einstellungen bieten Zugriff auf die Android Einstellungen für Standort & Sicherheit. Hier kann angegeben werden, ob der Standort mittels WLAN / Mobilfunknetz bestimmt werden kann. Ebenfalls wird hier definiert, ob GPS für die Lokalisierung verwendet wird.

5.10.5. Energie Einstellungen

Um den Energieverbrauch den persönlichen Wünschen anzupassen, können folgende Einstellungen vorgenommen werden:

- Bildschirm immer an → Während Trip bleibt der Bildschirm immer aktiv
- GPS-Service an → Service für GPS Datenaufzeichnung ist aktiviert
- OBD-Service an → Service für OBD2 Datenaufzeichnung ist aktiviert
- Beschleunigungs-Service an → Service für Datenaufzeichnung der Beschleunigungssensoren ist aktiviert

5.10.6. Masseinheiten Einstellungen

Damit die Masseinheiten der Eingabewerte den persönlichen Präferenzen entsprechen, kann der Benutzer die Masseinheiten folgender Gruppen anpassen:

- Distanz (Meter; Meile)
- Treibstoffmenge (Liter; Imperial Gallone; US Gallone)
- Verbrauch (l/100km; mpg (US); mpg (UK))
- Temperatur (Grad Celsius; Grad Fahrenheit)
- Druck (Bar; Psi)

5.10.7. Allgemeine Einstellungen

Unter den Allgemeinen Einstellungen kann definiert werden, ob ein Warnton abgespielt werden soll, wenn die Verbindung zur OBD2-Schnittstelle unterbrochen wurde. Zusätzlich kann der aktuelle Literpreis für den Treibstoff angegeben werden. Dieser Wert wird zur Berechnung der Kosten benötigt, die in der Tripanzeige aufgeführt sind. Ebenfalls könne in den allgemeinen Einstellungen die Datenschutzbestimmungen gelesen werden.

5.11. Permissions

Der EcoHelper benötigt diverse Android-Permissions. Diese werden dem Benutzer bei der Installation der Anwendung angezeigt. In dieser Auflistung soll festgehalten werden, für was welche Permission benötigt wird.

Permission	Verwendung
ACCESS_COARSE_LOCATION	Zugriff auf Netzwerk Ortungsdienst
ACCESS_FINE_LOCATION	Zugriff auf GPS-Ortungsdienst
ACCESS_NETWORK_STATE	Zugriff auf die Information, mit was für einem Netzwerk das Smartphone verbunden ist (UMTS, EDGE, …)
ACCESS_WIFI_STATE	Zugriff auf WiFi-Verbindung; Wird benötigt wenn ein WiFi-OBD2-Adapter implementiert wird.
BLUETOOTH	Zugriff auf Bluetooth zur Kommunikation mit Bluetooth-OBD2-Adapter
BLUETOOTH_ADMIN	Zugriff auf administrierende Bluetooth-Operationen wie Device-Discovery oder Pairing von neuen Geräten.
CAMERA	Zugriff auf die Kamera zum Erstellen von Bildern der OBD2-Schnittstellen
CHANGE_WIFI_STATE	Zugriff auf WiFi-Verbindung; Wird benötigt, wenn ein WiFi-OBD2-Adapter implementiert wird.
INTERNET	Zugriff auf die Internet-Verbindung; Wird für sämtliche Kommunikation mit dem Server benötigt
READ_PHONE_STATE	Zugriff auf Informationen zur aktuellen Netzverbindung (z.B. Signalstärke)
UPDATE_DEVICE_STATS	Wird benötigt, um WiFi zu aktivieren.
VIBRATE	Wird vom Workspace der RunningTripActivity verwendet. Beim Start des Drag&Drop wird dieser von einem kurzen vibrieren angezeigt.
WAKE_LOCK	Der Benutzer kann einstellen, ob während dem Trip der Bildschirm aktiviert bleiben soll.
WRITE_EXTERNAL_STORAGE	Zugriff auf die sdcard des Smartphones. Ein Android Smartphone hat immer eine sdcard. Falls physisch keine vorhanden ist, wird eine durch Android emuliert. Wird benötigt, um Kamerabilder temporär zu speichern.

Tabelle 9: Permissions

5.12. Konfigurierbarkeit

Der EcoHelper kann an diversen Punkten konfiguriert werden. Hier wird von Einstellungen gesprochen, welche nicht durch den Benutzer durchgeführt werden können, sondern nach einer Änderung eine neue Version der Anwendung nach sich ziehen. D.h. eine Codeänderung ist nötig.

5.12.1. Mehrsprachig

Eine Anwendung kann in Android einfach in mehreren Sprachen ausgeliefert werden. Die Anwendung selbst hat dabei selbst nicht die Möglichkeit, die Sprache umzustellen. Es wird immer die im System definierte Sprache verwendet. Falls die Anwendung, die für die angegebene Sprache benötigten Ressourcen zur Verfügung stellt, werden diese verwendet. Andernfalls werden die Standardressourcen der Anwendung geladen.

Die Ressourcen befinden sich im Ordner „res“. Für sprachspezifische Ressourcen muss an den gewünschten Ordner nur die Länder-, bzw. Sprachkennung angefügt werden:

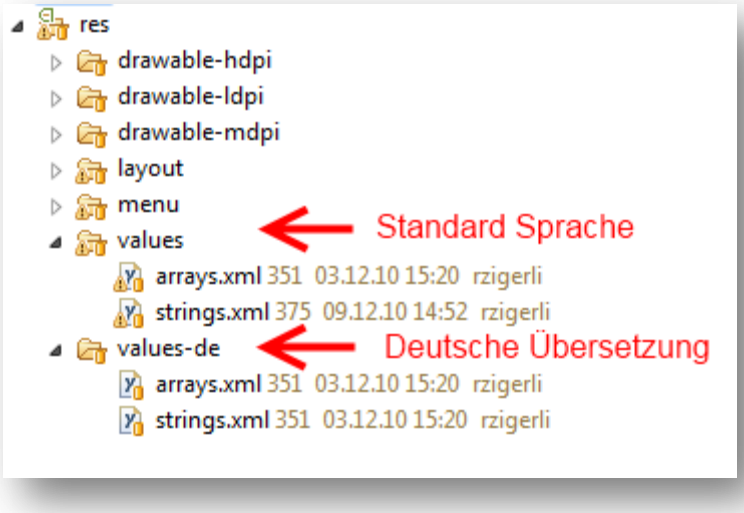


Abbildung 34: Ressource-Ordner für Mehrsprachigkeit

In diesem Beispiel liegen im Ordner „values“ die Standardressourcen, welche für sämtliche Sprachen ausser Deutsch verwendet werden. Für Deutsch wurden im Ordner „values-de“ eigene Ressourcen definiert.

Bei der Implementation des EcoHelper wurde darauf geachtet, dass sämtliche Texte, Klänge und Bilder in den Ressourcen abgelegt sind. Somit kann durch ein einfaches Kopieren der Ordner eine neue Sprache hinzugefügt werden.

5.12.2. OBD2-Parameter hinzufügen

Der EcoHelper implementiert eine Basissammlung der OBD2-Parameter. Falls weitere Parameter hinzugefügt werden sollen, kann dies durch folgende Änderungen im Code gemacht werden:

Klasse ParameterController

- Dem **Enum ParameterKey** ist der Name des neuen Parameters hinzuzufügen
- In der Methode **initObdParameter(..)** muss für den neuen Parameter eine entsprechende ParameterInfo-Instanz erstellt und der obdParam-ArrayList hinzugefügt werden.

Interface SampleColumns

Hier eine neue Spalte für den neuen Parameter erfassen.

Interface SampleTable

Sicherstellen, dass der in **SampleColumns** erfasste Parameter korrekt in dem Create-Script erstellt wird.

Klasse EcoHelperDatabase

SchemaVersion um eins erhöhen, damit der EcoHelper beim nächsten Start der Anwendung die Datenbank neu erstellt und somit die neue Spalte erstellt wird.

Klasse ParameterController

In der Methode **getDbField(..)** für den definierten ParameterKey die korrekte Spalte zurückgeben.

Klasse ObdSample

Ein für den Parameter zutreffendes Public-Field erstellen.

Achtung: Aus diesem Objekt wird automatisch durch GSON die JSON-Repräsentation generiert. Der FieldName entspricht dann dem Key im JSON-Format. Der Webserver muss den neuen Key ebenfalls kennen, ansonsten schlägt die Kommunikation mit dem Server fehl!

Klasse GetSamplesForUploadHandler

Sicherstellen, dass der neue Parameter aus der Datenbank in das public Field, welches in der Klasse ObdSample erstellt wurde, abgefüllt wird.

Klasse ObdResponse

In der Methode **calculateDataValue()** muss die für den Parameter korrekte Berechnung eingetragen werden.

5.12.3. Sample-Intervall

Der EcoHelper kennt zwei unterschiedliche Sample-Intervalls:

- Normale Sample (OBD2, Ortungsdaten)
- Acceleration Sample (Beschleunigungsdaten)

Grund dafür ist, dass die Beschleunigungssensoren wesentlich häufiger die Daten aktualisiert.

In der Klasse **DataCaptureService** können dazu die folgenden Konstanten angepasst werden:

```
/**
 * Interval in dem Snapshot aller Daten gemacht werden in ms
 */
private static final Integer snapshotInterval = 500;

/**
 * Interval in dem Snapshots der beschleunigungs-Daten gemacht werden in ms
 */
private static final Integer accelerationSnapshotInterval = 200;
```

Abbildung 35: Codesnippet Sample-Intervall

5.12.4. Upload

Intervall

Im Online-Mode sendet der EcoHelper während dem Trip in einem angegebenen Intervall (in ms) die vorhandenen Messdaten an den Server. Dieses Intervall kann in der Klasse **DataUploadTask** angepasst werden:

```
/**
 * Interval in dem die in der Datenbank gespeicherten
 * Daten-Snapshots an den Server gesendet werden.
 */
private static final int uploadInterval = 15000;
```

Abbildung 36: Codesnippet Upload-Intervall

Anzahl Samples

Pro Upload wird eine angegebene Anzahl an Messungen (Samples) an den Server gesendet. Wie viele Samples maximal pro Upload gesendet werden dürfen, kann in der Klasse **DataUploadTask** angepasst werden:

```
/**
 * Maximale Anzahl Messungen welche pro Upload an den
 * Server gesendet werden.
 */
private static final int samplesPerUpload = 50;
```

Abbildung 37: Codesnippet Samples pro Upload

5.13. Exceptionhandling

Während der Entwicklung haben wir laufend Testversionen an Herr Heinzmann und Omid Afshari verteilt. Um auftretende Abstürze nachvollziehen zu können, wurde ein Logging implementiert.

Um dies möglichst einfach zu gestalten wurde ein eigener ExceptionHandler implementiert. Dieser geht im Falle einer nicht abgefangenen Exception wie folgt vor:

- Wenn Internetverbindung vorhanden → Upload der Informationen zum EcoHelper-Server
- Wenn keine Internetverbindung vorhanden → Speichern der Informationen als .stacktrace-Datei auf der sdcard des Smartphones.

Nebst dem Stacktrace werden folgende weitere Informationen zum Gerät, der Android sowie der EcoHelper-Version mit gesendet:

Information	Beschreibung
Anwendung-Version-Code	Version-Code der EcoHelper- Anwendung
Anwendung-Version-Name:	Version-Name der EcoHelper- Anwendung
Android-Version	Android-Version (z.B. 2.2)
Board	Name des zugrundeliegenden Boards
Brand	Der Hersteller für den Android modifiziert wurde (z.B. HTC)
Device	Name des Geräts durch den Hersteller gegeben
Display	Angezeigte Android-Build-Version (z.B. FRF91)
Model	Modelname des Smartphone (z.B. HTC Desire)
Imei	Imei des Benutzers von dem der Bericht kommt
Zeit	Zeitpunkt des Absturz
Stacktrace	Stacktrace zur Absturzstelle

Tabelle 10: Mit gesendete Informationen Exceptionhandling

6. Testing

6.1. Systemtests

Durch den Systemtest wurden diverse Eigenschaften der Anwendung getestet und verifiziert. Unter anderem wurde der Workflow der Anwendung gründlich untersucht. Ebenfalls konnte die Usability k nochmals verbessert werden. Der Systemtest befindet sich zur Einsicht im Anhang.

6.2. Unittests

Auf Unittests wurde in Rahmen des EcoHelper-Projekts aus zeitlichen Gründen grösstenteils verzichtet. Grund dafür ist, dass man sehr viel mit aufwändigeren Unittest-Verfahren (z.B. hätte man diverse Mocks erstellen müssen) arbeiten, müsste um unter Android brauchbare Unittest zu erstellen.

Für den Test der Gangerkennung wurde ein einfaches Unittest-System erstellt. Dies erlaubt es, Log-Files aus Testfahrten direkt als Unittests auszuführen. Damit konnten wir die Gangerkennung ständig weiterentwickeln und gleichzeitig sicherstellen, dass die Änderungen keine negativen Auswirkungen auf bestehende Testdaten haben.

6.3. Testautos

Während der Entwicklung haben wir Testfahrten mit folgenden Testautos durchgeführt:

Auto	Treibstoff	Gänge	Getriebe
Ford Focus	Benzin	6	Handschaltung
Opel Combo	Benzin	5	Handschaltung
Renault Clio	Diesel	5	Handschaltung
Smart	Benzin	6	Halbautomat
Mercedes GLK	Diesel	7	Automat
Audi A5	Benzin	6	Automat
Hyundai i20	Benzin	5	Handschaltung
Chevrolet Aveo	Benzin	5	Handschaltung
Mazda 3 Hatchback 1.6 Confort	Benzin	5	Handschaltung

Tabelle 11: Testautos

6.4. Testgeräte

Der EcoHelper wurde auf folgenden Endgeräten entwickelt und getestet.

Gerät	Android Version
Samsung Nexus S	2.3.4
HTC Desire	2.2
HTC Desire HD	2.3.3

Tabelle 12: Android Testgeräte

6.5. Durchgeführte Unittests

Folgende Tabelle gibt Aufschluss darüber, aus welchen Testautos mit welchen Testgeräten Daten für die Unittests ausgelesen wurden:

Auto	Samsung Nexus S	HTC Desire	HTC Desire HD
Ford Focus			
Opel Combo			
Renault Clio			
Smart			
Mercedes GLK			
Audi A5			
Mazda 3 Hatchback 1.6 Confort			

Tabelle 13: Durchgeführte Unittests

6.6. Durchgeführte Systemtests

Folgende Tabelle gibt Aufschluss darüber, mit welchen Testgeräten die Systemtests durchgeführt wurden:

Gerät	Android Version	Status
Samsung Nexus S	2.3.4	OK
HTC Desire	2.2	OK
HTC Desire HD	2.3.3	OK
Emulator	2.1 update 1	NOK (6.7.2 Android 2.1)
Emulator	2.2	OK
Emulator	2.3	OK

Tabelle 14: Durchgeführte Systemtests

6.7. Anpassungen für tiefere SDK

EcoHelper wurde auf Android 2.3 entwickelt. In diesem Kapitel wurde die Kompatibilität mit eventuell nötigen Anpassungen zu tieferen Versionen untersucht.

6.7.1. Android 2.2

Änderungen an der API durch Android 2.2:
http://developer.android.com/sdk/android-2.2.html

Die Angegebenen Änderungen haben keinen Einfluss auf EcoHelper.

6.7.2. Android 2.1

Änderungen an der API durch Android 2.1:
http://developer.android.com/sdk/android-2.1.html

In der Version 2.1 gibt es die Methode

```
public long[] getCheckedItemIds()
```

nicht. Diese wird in der Anzeige „Trips vergleichen“ benötigt. Dies müsste wieder mit der deprecated Methode

```
public long[] getCheckItemIds ()
```

implementiert werden.

7. Weitere Arbeiten

Aus zeitlichen Gründen wurde bei der Implementation des EcoHelper auf folgende Optimierungen und Funktionen verzichtet.

7.1. Optimierung

7.1.1. Datenübertragung

Während der gesamten Entwicklung der Kommunikation wurde das Hauptaugenmerk auf die Funktionalität sowie die Einfachheit der Übertragenen Daten gelegt. Entsprechend wurde das Übertragungsformat nicht optimiert, wodurch erheblich an übertragenem Datenvolumen gespart werden könnte. Ein Ansatz zur Minderung des Datenvolumens wäre, die Attribut-Namen nicht mehr voll auszuschreiben. Optimal würde man für jeden Attribute-Namen einen Key definieren:

z.B. würde aus „carId“ neu „1“ oder aus „obd_samples_attributes“ neu „2“. Mit diesem Ansatz könnte bis zu 70% übertragenen Daten eingespart werden.

7.1.2. Performance

Die Performance der Anwendung kann noch verbessert werden. Z.B. liegt die Prozessorauslastung während der Datenaufzeichnung bei rund 20%. Hier kann nebst der Performance auch eine bessere Akkulaufzeit erreicht werden.

7.2. Weitere Funktionen:

7.2.1. Übersetzung in andere Sprachen

Um die Anwendung einem weltweitem Publikum zur Verfügung zu stellen, kann EcoHelper mit geringem Aufwand in verschiedene Sprachen übersetzt werden.

7.2.2. Diagramme für RunningTrip

Für eine bessere Darstellung der Messwerte können Anzeigeelemente mit Graphen und Diagrammen entwickelt werden.

7.2.3. Geführte Gangerkennung

Um die Gangerkennung zu vereinfachen, kann eine geführte Gangerkennung entwickelt werden. Bei dieser wird dem Benutzer gesagt, dass er zu Beginn im ersten Gang fahren soll. Sobald der erste Gang korrekt erkannt wurde, wird dem Benutzer mitgeteilt, dass er nun im zweiten Gang fahren soll usw.

7.2.4. Video- / Fotostreaming

Damit ein Trip mit einem Video- oder Fotostream erweitert werden kann, kann diese Funktion noch implementiert werden.

7.2.5. OBD2 WiFi Unterstützung

Um OBD2 Werte auch über einen WiFi Adapter auszulesen, kann diese Funktion, sobald Android Ad-hoc Verbindungen zulässt, implementiert werden.

7.2.6. Anpassung für Tablets

Damit EcoHelper auch auf Android Tablets genutzt werden kann, muss das GUI an die Grösse der Tablets angepasst werden.

7.2.7. Appstyle verbessern

Um einen einheitlichen Stil der Anwendung zu erreichen, können neue Icons definiert und ein eigener Style erstellt werden.

7.2.8. Optimaler Schaltzeitpunkt

Eine CustomView entwickeln welche dem Benutzer den optimalen Zeitpunkt während eines Trips anzeigt.

7.2.9. Userrollen

Anhand der Userrollen bei der Registration könnten unterschiedliche Funktionalitäten angeboten werden.

7.3. Bugs

Die Bugs sind zur Verwaltung in Redmine erfasst.

7.3.1. Erstellung Foto der OBD2 Schnittstelle schlägt fehl

Beim Erstellen eines Fotos der OBD2 Schnittstelle kommt es bei Android Tablets (Motorola Xoom, Acer A500) und dem HTC Desire HD zu Problemen. Sobald die Android Camera-Activity im Landscape Mode verlassen wird, wird die ObdInfoActivity neu gestartet, anstatt das gemachte Foto in der ursprünglichen Activity anzuzeigen. Dieser Bug ist auf die eigene Implementation der Benutzeroberfläche (Android Camera-Activity) der Hersteller zurückzuführen. Sobald es neue Versionen der herstellerspezifischen Benutzeroberflächen gibt, kann dieser Bug behoben werden.

7.3.2. Einlieferung OBD2 Daten

Es werden keine OBD2 Werte für consumption, load, fuel_status, fuel_correction, distance übertragen. Bei fuel_status sind falsche Werte in der DB => 100,90,... sollten 1,2,4,8 etc. sein.

7.3.3. Meldung, wenn Passwort auf Server geändert wurde

Sobald der User das Passwort auf dem Server geändert hat, muss eine entsprechende Meldung auf dem Gerät erscheinen, der den User auffordert, das Passwort erneut einzugeben.

8. Schlussfolgerungen

Das EcoHelper-Android Projekt war ein voller Erfolg. Die wichtigen Funktionen und Ideen aus früheren Arbeiten und anderen Anwendungen wurden in EcoHelper-Android umgesetzt. Dabei konnten eigene neue Impulse in die Implementierung eingebracht werden. Auf erweiterte Funktionen wie z.B. das Videostreaming musste aus zeitlichen Gründen verzichtet werden.

Die Benutzerführung wurde in der gesamten Anwendung einheitlich gestaltet. Dabei wurde grossen Wert auf ein „Android-taugliches“ Verhalten gelegt. D.h. dass Aktionen immer über das Menü erreichbar sind oder das Listenelemente entsprechend über Longclicks editiert werden können. Weiter sollte die Oberfläche ohne zusätzliche Hilfsmittel wie ein Stift o.ä. bedienbar sein. Aus diesem Grund wurde darauf geachtet, dass sämtliche Navigationselemente problemlos mit dem Finger bedient werden können. Die Anzeige der Parameter während eines Trips konnte durch eine sehr einfach erweiterbare Oberfläche für den Benutzer optimal gestaltet werden. Die Navigation durch Finger-Gesten verleiht der EcoHelper-Anwendung dabei einen frischen und modernen Charakter.

Als Problem während der Entwicklung ist uns einzig aufgefallen, dass das Vorgehen zur Spezifikation der Webserverschnittstellen nicht optimal geregelt wurde. Omid Afshari hat parallel zur EcoHelper-Anwendung die Webanwendung komplett neu entwickelt. Dabei haben die Schnittstellen, über welche die Serverkommunikation läuft, geändert. Hier wäre eine klare Spezifikation zu Beginn des Projekts eine sinnvolle Lösung gewesen.

9. Verzeichnisse

9.1. Abbildungsverzeichnis

9.2. Tabellenverzeichnis

10. Glossar

Begriff	Beschreibung
Ad-hoc	WiFi-Verbindung zwischen zwei Endpunkten ohne einen Access Point http://de.wikipedia.org/wiki/Wireless_Local_Area_Network#Ad-hoc-Modus
ANR	Application Not Responding. Eine Android-Fehlermeldung die auftritt, wenn eine Anwendung länger als fünf Sekunden nicht auf eine Benutzerinteraktion reagiert oder mehr als zehn Sekunden für die Verarbeitung benötigt.
API	Das Application Programming Interface definiert die Programmanbindung auf Quelltextebene.
Bearing	Stellt die aktuelle Abweichung in Grad zwischen Norden & der aktuellen Fahrtrichtung dar.
CSV	Comma Separated Values. Dateiformat http://de.wikipedia.org/wiki/CSV_(Dateiformat)
EcoHelper	Titel der Anwendung. Es ist eine eigene Wortschöpfung aus eco/umweltfreundlich und „helper“ Helfer.
EcoZahl	Die EcoZahl berechnet sich aus der Formel: (Geschwindigkeit [km/h] * Gewicht [kg]) / (Verbrauch [l/100km]) Je höher der Wert ist, desto effizienter ist die Fahrweise.
Friendly User Test	Test mit ausgwähltem Personenkreis (Betreuer, Assistent & Fahrlehrer)
GPS	Global Position System: Satelitenbasiertes Positionierungs-System
GUI	Grafische Benutzeroberfläche
Imei	Die International Mobile Station Equipment Identity identifiziert ein Mobilgerät eindeutig
JSON	JavaScript Object Notation; Ein kompaktes Datenformat zum Austausch von Daten zwischen Anwendungen.
Keyhole Markup Language KML	Ein Datenformat zum beschreiben von Geodaten. Kann direkt in Google Maps importiert und angezeigt werden.
Longclick	Unter Android wird ein Longclick als langer Klick auf ein Element definiert. Meist wird ein Longclick auf ein Element verwendet um z.B. ein Context-Menü anzuzeigen.
MAF (Mass Air Flow)	Die Masse der pro Zeiteinheit durch den Verbrennungsmotor durchströmenden Luft.
OBD2	Das On-Board-Diagnose Interface 2 ist eine standardisierte Schnittstelle um diverse Parameter eines Fahrzeugs zu lesen/schreiben. Diese muss in Benzin-Autos ab 2001 eingebaut werden.
PID	OBD2 Parameter-ID zur klaren Identifikation eines Parameters.
REST	Representational State Transfer; Beschreibung einer einfachen http-Übermittlung ohne Session und Cookies. Jeder Aufruf muss entsprechend die kompletten Informationen mit senden.
RUP	Rational Unified Process; Interativer Softwareentwicklungsprozess http://de.wikipedia.org/wiki/Rational_Unified_Process
Schubabschaltung	Im allgemeinen Sprachgebrauch auch „Motorenbremse“ genannt. Dem Motor wird die Treibstoffzufuhr abgestellt wodurch er sich nur durch seine eigene Schwungmasse weiterantreibt. Wenn die Schubabschaltung aktiv ist, wird kein Treibstoff benötigt.

SDK	Software Development Kit
Trip	Eine Fahrt des Benutzers der EcoHelper-Anwendung.
Usability Tests	Test, um Gebrauchstauglichkeit zu überprüfen
VIN	Vehicle Identification Number; Eindeutige Nummer zur Identifikation eines Autos. http://en.wikipedia.org/wiki/Vehicle_Identification_Number
Windows Mobile 5.0	Ältere Windows Variante die für mobile Endgeräte gedacht war. Wurde von Microsoft durch Windows Phone 7 ersetzt.

Tabelle 15: Glossar

11. Quellen

Folgend sind allgemeine Quellen aufgeführt. Spezifische Quellen wurden direkt als Fussnote vermerkt.

11.1. Internet

Allgemein Android
Letzter Zugriff: 14.06.2011
Beschreibung: Offizielle Dokumentation der Android API.
http://developer.android.com/index.html

Allgemeine Probleme
Letzter Zugriff: 14.06.2011
Beschreibung: Frage/Antwort-Portal.
http://stackoverflow.com/questions/tagged/android

Android Progress Bar
Letzter Zugriff: 14.06.2011
http://guides.wikinut.com/Progress-Bar-On-Android/4ws2d1y7/

Android Google Maps
Letzter Zugriff: 14.06.2011
http://mobiforge.com/developing/story/using-google-maps-android
http://stackoverflow.com/questions/4141388/current-position-on-google-maps
http://blogs.itemis.de/frey/2009/04/15/location-based-services-on-android-part-33-customoverlays/

Android Bluetooth
Letzter Zugriff: 14.06.2011
http://developer.android.com/guide/topics/wireless/bluetooth.html

Android Video Broadcast
Letzter Zugriff: 14.06.2011
http://www.mattakis.com/blog/kisg/20090708/broadcasting-video-with-android-without-writing-to-the-file-system

Android Sensoren
Letzter Zugriff: 14.06.2011
http://www.touchqode.com/misc/20101025_jsug/20101025_touchqode_sensors.pdf

HTC Bluetooth Bug
Letzter Zugriff: 14.06.2011
http://code.google.com/p/android/issues/detail?id=5427#c14

Android Drag&Drop
Letzter Zugriff: 14.06.2011
http://stackoverflow.com/questions/4441464/how-to-get-the-homescreen-behaviour-of-drag-to-bin-in-my-own-application

Android Homescreen
Letzter Zugriff: 14.06.2011

http://android.git.kernel.org/?p=platform/packages/apps/Launcher2.git;a=tree;h=refs/heads/master;hb=refs/heads/master
http://stackoverflow.com/questions/3467461/developing-an-android-homescreen
http://androidworkz.com/2010/07/06/source-code-imageview-flipper-sd-card-scanner/

GSON
Letzter Zugriff: 14.06.2011
http://sites.google.com/site/gson/gson-user-guide

UTF8-Encoding
Letzter Zugriff: 14.06.2011
http://sites.google.com/site/gson/streaming

OBD2 Informationen Allgemein
Letzter Zugriff: 14.06.2011
http://en.wikipedia.org/wiki/OBD-II_PIDs

OBD2 Abfrageintervall
Letzter Zugriff: 14.06.2011
http://www.mp3car.com/engine-management-obd-ii-engine-diagnostics-etc/140147-optimizing-scan-rate-2.html

http://www.obd2crazy.com/techatst.html

OBD2 Einbauorte
Letzter Zugriff: 14.06.2011
http://de.openobd.org/einbauorte.htm#audi_a8_4d

Tomcat
Letzter Zugriff: 14.06.2011
http://www.avajava.com/tutorials/lessons/how-do-i-deploy-to-tomcat-using-ant.html

Android Kamera
Letzter Zugriff: 14.06.2011
http://code.google.com/p/krvarma-android-samples/source/browse/#svn%2Ftrunk%2FCameraDemo

Verbrauchumrechnung
Letzter Zugriff: 14.06.2011
http://de.wikipedia.org/wiki/Kraftstoffverbrauch#Umrechnung_zwischen_l.2F100_km_und_mpg

Mobile Provider List
Letzter Zugriff: 14.06.2011
http://www.gsmcenter.pl/pg.php?tid=88&s=&dodruku=1

Beep-Ton für OBD Verbindungsabbruch
Letzter Zugriff: 14.06.2011
http://www.soundjay.com

Auto-Icon
Letzter Zugriff: 14.06.2011

http://www.bestfreeicons.com/c47-3d-icons-0.html

People-Icon
Letzter Zugriff: 14.06.2011
http://www.iconspedia.com/icon/people-4-33.html

11.2. Bücher

Android 2 – Grundlagen der Programmierung
Bruno Becker / Marcus Pant
ISBN: 978-3-89864-677-2
2. aktualisierte und erweiterte Auflage
dpunkt.verlag

11.3. Frühere Arbeiten

EcoTrainer Windows Mobile 5.0
Bruno Krieg, Daniel Wydler
1. Oktober – 23. November 2007

EcoTrainer Android
Cédric Menzi, Mathias Sulser
18. Februar – 6. Juni 2008

EcoHelper – Webapplikation
Stephan Hauser, Michael Wagner
15. September – 19. Dezember 2008

JTourLive – www.tourlive.ch
Stephan Hauser, Michael Wagner
16. Februar – 12. Juni 2009

EcoHelper für das iPhone
Selim Akyol, Edon Berisha
22. Februar – 18. Juni 2010

12. Anhang

12.1. Inhalt CD

Ordner	Inhalt
Bericht	Sämtliche Dokumente zum Bericht
index.html	HTML-Datei die eine Übersicht über die CD vermittelt
Poster	Poster für die öffentliche Präsentation der Arbeit
Präsentation	Durchgeführte Präsentation
Sitzungsprotokoll	Sitzungsprotokolle während der gesamten Arbeit
Source Code	Quellcode des EcoHelper
Videos	Videos einer Beispielfahrt im Zusammenspiel mit EcoHelper
Webseite	Enthält Ressourcen der index.html

Tabelle 16: Inhalt Abgabe CD

12.2. Persönliche Berichte

12.2.1. Raphael Nagy

Die Bachelorarbeit begann damit, dass wir uns in das komplexe Thema einlesen mussten. Herr Heinzmann gab uns auch gleich nach dem ersten Meeting die früheren Bachelorarbeiten zum Thema EcoHelper. Danach konnten wir uns richtig in die Arbeit stürzen und uns einen guten Überblick verschaffen, was auf uns zukommen könnte, da schon einige Arbeiten, auf anderen Plattformen, erstellt wurden. Die Aufteilung des zu lesenden Stoffes klappte wie gewohnt sehr gut. Von Anfang an gab es nie Probleme mit meinen zwei Kommilitonen Reto Zigerlig und Stephan Schöb.

Nach einer fast vierwöchigen Analyse und Anforderungsaufnahme-Phase hatten wir genügend Informationen und Hintergrundwissen um mit der Implementation zu beginnen. Die Entscheidung mit dem GUI-Papierprototyp anzufangen war richtig und ich würde jederzeit wieder so vorgehen. Dadurch, dass der GUI-Papierprototyp direkt Testpersonen gezeigt werden konnte und man wusste worüber man diskutierte war sehr nützlich.

Nachdem der Papierprototyp fertiggestellt war traf ich mich mit Fahrlehrer Rolf Bader aus Engelburg SG. Bei diesem Treffen stellte ich ihm das Projekt und den Papierprototyp vor. Er fand das Konzept sehr gut.

Gleichzeitig begann Reto Zigerlig und Stefan Schöb mit der Implementation. Bei der Implementations-Phase war ich hauptsächlich für die Schnittstelle zur Webserver-Anwendung zuständig, welche von Omid Afshari von HSR/cnlab AG gleichzeitig entwickelt wurde, und die Implementation der Tankverwaltung. Auch diese Zusammenarbeit verlief sehr gut. Es gab nie ernsthafte Probleme und die Anwendung nahm schnell Formen an.
Es gab einige Schwierigkeiten bezüglich den Technologieunterschieden von Ruby und Java. Aber durch die intensive Kommunikation mit Omid Afshari waren auch diese schnell gelöst. Der Einsatz der Gson-Library, für die Objekt De-/Serialisierung, war sehr hilfreich und ich würde diese Library jederzeit wieder einsetzten.

Auch die wöchentlichen Meetings fand ich sehr gut und trugen wesentlich zum Projekterfolg bei. Herr Heinzmann und Omid Afshari gaben immer sehr hilfreichen Input und Ideen oder Anregungen. Mit der Betreuung war ich vollkommen zufrieden und möchte mich an dieser Stelle bei Prof. Dr. Peter Heinzmann und Omid Afshari, sowie bei meinen zwei Teamkollegen, bedanken.

Der Knowhow-Gewinn, in allen Bereichen in denen ich gearbeitet hatte, war enorm. Dadurch, dass wir uns hauptsächlich auf die Client-Anwendung konzentrieren konnten, vertiefte ich nochmals mein Wissen über die Android-Plattform.

Rückblickend war das Projekt ein voller Erfolg nicht zuletzt wegen den Eingesetzten Tools, wie Redmine, Skype und vor allem wegen den kompetenten Teamkollegen.

12.2.2. Stefan Schöb

In der ersten Woche ging es zuerst darum, sich in die Thematik und bestehende Projekte einzuarbeiten. Ich habe dabei das Hauptaugenmerk auf die bestehende Android sowie die Windows Mobile 5.0-Anwendung gelegt. Nach der ersten Sitzung mit Peter Heinzmann und Omid Afshari war uns auch bereits klar, dass es hier nicht nur um ein reines Software-Projekt geht. Im ersten Teil der Arbeit war das Ziel hauptsächlich das Finden von Ideen für die.

Aus den bestehenden Arbeiten haben wir dann die besten Funktionen zusammengefasst und versucht in die neue Android-Anwendung unterzubringen. Ein erster Vorschlag haben wir mit Paperprototypes visualisiert und diente die nächsten Wochen als Gesprächsgrundlage. Gleichzeitig haben wir mit der Implementierung der Oberfläche begonnen.

Meine erste Aufgabe bestand darin, eine Art Workspace zu implementieren auf dem der Benutzer seine Elemente per Drag&Drop selbst platzieren kann. Ich habe mich dabei auf die Android-Implementation des Homescreen gestürzt und zahlreiche Beispiele durchgearbeitet. Nach kurzer Zeit hatte ich einen ersten Prototyp fertig welcher die Basisfunktionalität bereitstellte. Bis dieser jedoch sämtliche gewünschten Funktionen stabil unterstützte, benötigte ich ca. drei weitere Wochen.

Als zweite grosse Aufgabe folgte die Datenaufzeichnung. Es ging dabei darum, sämtliche Datenquellen (OBD, Sensoren, Ortungsdienste) abzufragen und in eine Datenbank zu speichern. Die Ortungsdienste sowie Sensoren stellten dabei keine Schwierigkeit dar. Wesentlich anspruchsvoller war die Implementation der OBD-Kommunikation. In der Aufgabenstellung wurde definiert, dass diese sowohl über WiFi wie auch Bluetooth funktionieren muss. Das System wurde entsprechend aufgebaut, auch wenn Reto nach kurzer Recherche feststellen musste, dass wir den WiFi-Teil aus technischen Gründen nicht implementieren können. Der Bluetooth-Teil wurde durch mich relativ schnell implementiert. Getestet in einer ersten Version auf dem Nexus S. Erste Tests durch Omid Afshari ergaben jedoch, dass HTC-Geräte diverse Probleme mit der Bluetooth-Implementation aufweisen wodurch einige Änderungen und Workarounds nötig wurden.

Die dritte grosse Aufgabe war die Gangerkennung. Wir haben hier auf der Implementation der bestehenden Java-Lösung aufgebaut und diese verbessert. In zahlreichen Testfahrten und Besprechungen im Team konnten wir hier wesentliche Punkte verbessern.

Das ganze Projekt ist aus meiner Sicht sehr erfolgreich verlaufen. Wir hatten im Team keine Differenzen und technische Probleme konnten wir optimal in der Gruppe diskutieren. Durch die ständige Anbindung per Skype sowie die fast wöchentlichen Sitzungen war auch die Kommunikation mit unserem Betreuer kein Problem. Wir konnten unsere Ideen in das Projekt einbinden. Diese wurden meist positiv angenommen oder durch handfeste Argumente verworfen.

Einziges Problem während der Implementation war teilweise die Ungewissheit, wie die Kommunikation mit dem Server durchgeführt wird. Benutzen wir jetzt Container? Doch nicht? Gehört dieses Attribut zu diesem Objekt? Usw. Diese Fragen konnten wir jeweils per Skype mit Omid Afshari klären, besser wäre hier vielleicht eine fixe Schnittstellenspezifikation gewesen, welche man vor der Implementationsphase definiert hätte.

12.2.3. Reto Zigerlig

Da meine vorangegangenen Arbeiten im Bereich der Softwareentwicklung für Android stattfanden, wollte ich auch in der Bachelorarbeit etwas in diesem Umfeld entwickeln. Nachdem wir unser Team zusammen hatten, entstand schnell eine Präferenz für die EcoHelper-Anwendung. Grund dafür war einerseits die Thematik an sich, den Autofahrstil auf ökologische Aspekte zu untersuchen, anderseits begeisterte mich die Kommunikation mit einem Webserver. Über den Zuschlag für die Arbeit freuten wir uns sehr und starteten voller Elan in die Bachelorarbeit.

Zu Beginn des Projektes wurde eine ausführliche und weitreichende Analyse durchgeführt. Dies war ein wichtiger Einstieg der es erlaubte, einen ersten Überblick über die Thematik rund um das EcoHelper-Projekt zu erlangen. Ebenfalls empfand ich die Analyse der bestehenden Arbeiten und der vorhandenen Anwendungen als sehr nützlich. Anschliessend an die Analysephase wurde der Projektumfang erarbeitet und es wurde eine priorisierte Featurelist erstellt. In dieser Anfangsphase entschlossen wir uns, einen ausführlichen Paperprototype zu erstellen. Dieser diente als Diskussionsgrundlage und erwies sich als sehr nützlich. Nach der Rücksprache mit Herrn Heinzmann, Omid Afshari und Herrn Bader konnte der Funktionsumfang definiert werden. Nun wurde die Arbeit anhand des Paperprototypes gemäss den persönlichen Präferenzen auf die Teammitglieder aufgeteilt. Ich beschäftigte mich zu Beginn mit der Ausarbeitung einer Lösung für das Hinzufügen eines Autos. Dieser Prozess enthielt die Gestaltung des GUI, die Kommunikation mit der Webanwendung sowie die Speicherung der Daten in die lokale Datenbank. In diesem Zusammenhang habe ich die lokale Datenbank für die Anwendung erstellt. Eine weitere Aufgabe bestand darin, ein Bild des OBD2 Anschlussplatzes zu erstellen und dieses mit Zusatzinformationen auf den Sever zu übertragen. Durch diverse Testfahrten haben wir die Anwendung gründlich getestet und können somit eine stabile Version vorweisen.

Während dem gesamten Projekt haben wir in regelmässigen Meetings den aktuellen Stand und das weitere Vorgehen mit Herrn Heinzmann und Omid Afshari besprochen. Diese Meetings waren sehr informativ und halfen uns, das Projekt in die richtige Richtung zu bringen. Die übliche Tatsache, dass während einem Projekt die Anforderungen ständig wieder angepasst werden, trifft auch auf dieses Projekt zu. Diese Challenge war nicht immer einfach zu bewältigen, gestaltete das Projekt jedoch sehr realitätsgetreu. Während der Entwicklung wurde die Webanwendung von Omid Afshari neu entwickelt. Dies führte zum Teil zu unerwartetem Verhalten der Anwendung, konnte jedoch jeweils sehr rasch geklärt werden. In diesem Zusammenhang möchte ich mich bei Herrn Heinzmann und Omid Afshari für die kompetente Betreuung des Projektes bedanken.

Die Arbeit in unserem Dreierteam funktionierte sehr gut. Bei wichtigen Entscheidungen wurden alle Meinungen angehört und diskutiert. Wenn jemand Probleme hatte, wurde gegenseitig Unterstützung geboten. Dies führte zu einem grossen Wissensaustausch der jeweiligen Teilgebiete. Da mich die Arbeit an diesem Projekt stets motivierte, war es auch kein Problem, mehr Zeit zu investieren als ursprünglich geplant.

Durch dieses Projekt konnte ich meine Fähigkeiten in der Erstelllung von Android Anwendungen stark verbessern und erhielt Einblick in die Anbindung einer mobilen Anwendung an eine Webanwendung. Das EcoHelper Projekt bot jederzeit eine interessante Domäne, in der die Resultate unserer Arbeit praxisnahe wiederspiegelt wurden.

Abschliessend möchte ich mich bei meinen Projektpartnern Raphael Nagy und Stefan Schöb für die erfolgreiche Zusammenarbeit bedanken.

12.3. Lessons Learned

Folgende Punkte fassen die Lessons Learned zusammen:

- Das Vorgehensmodell RUP, welches gewählt wurde, musste durch die Agilität, die dieses Projekt verlangte, angepasst werden.
- Android gibt es seit Oktober 2008, jedoch gibt es noch viele Funktionen, welche noch nicht, oder ungenügend unterstützt werden. Wie zum Beispiel die Ad-Hoc WLAN Verbindungen.
- Viele Anleitungen und Tipps, rund um Android, im Internet sind schlecht erklärt und enthalten zum Teil Fehler, die erst bei der Ausführung der Anwendung auftreten. Diese Fehler zu finden und zu beheben kostet massiv Zeit.
- Das Android-Framework ist im Vergleich zu anderen Technologien sehr gut dokumentiert, auch wenn die eine oder andere Lücke noch zu finden ist.
- Dropbox eignet sich sehr gut um Dokumente auszutauschen.
- Die Kommunikation mit Omid Afshari über den Skype-Chat funktioniert sehr gut, jedoch sind persönliche Meetings bei komplexeren Problemen unabdingbar.
- Das Redmine-Tool ist als Projektmanagement-Tool nicht geeignet. Es ist ein Bug-Tracking-Tool.
- Will man eine Teamarbeit effektiv, wie in diesem Projekt, anpacken, sollte eine geeignete Aufteilung der Arbeiten gemacht werden. Wichtig ist auch, dass die Schnittstellen klar definiert werden.

12.4. Datenvolumen

Da EcoHelper mit einem Webserver kommuniziert, haben wir die anfallenden Datenmengen analysiert und in folgender Tabelle zusammengefasst. Dabei handelt es sich um Testmessungen im Testbetrieb. Werte im laufenden Betrieb können von diesen Daten abweichen.

Aktion	Datenmenge	Bemerkung
Login	672 Byte	Kann je nach Anmeldename/Passwort variieren. Sollte sich sonst immer in diesem Rahmen befinden.
Auto hinzufügen	3.72 KB	Dieser Wert kann stark variieren. Je nach Grösse der Auto-Datenbank welche auf dem Server erfasst wurde. Der hier angegebene Wert wiederspiegelt pro Auswahl (Hersteller/Model/Variante) eine Auswahl von etwa 10 Elementen.
Tankvorgang hinzufügen	1002 Byte	Kann je nach Länge der Bemerkung variieren. Sollte ansonsten in diesem Rahmen bleiben.
Fahrer hinzufügen	476 Byte	Kann je nach Länge von Namen und E-Mail-Adresse im kleinen Bereich variieren. Sollten ansonsten in diesem Rahmen bleiben
Trip mit OBD	187.8 KB pro Minute	
Trip ohne OBD	32.6 KB pro Minute	
Trip synchronisieren	1,33 KB pro Trip	Kann sich im kleinen Rahmen durch grössere Werte der einzelnen Parameter verändern.
Tankvorgänge synchronisieren	578 Byte pro Tankvorgang	Kann je nach Beschreibung des Tankvorgangs variieren.
Fahrer Synchronisieren	437 Byte pro Fahrer	Kann je nach Name und E-Mail des Fahrers variieren.
Autos synchronisieren	554 Byte pro Auto	Wert kann variieren, je nachdem ob die Gang-Übersetzungsverhältnisse mit übertragen werden.

Tabelle 17: Datenvolumen

12.5. Android Crashkurs

Folgender Abschnitt soll eine kurze Einführung in Android geben und dem Android-fremden Leser ein Grundverständnis der Android-Bauweise vermitteln.

12.5.1. Komponentenbauweise

Android-Anwendungen bestehen aus Komponenten welche voneinander unabhängig sind. Dies ermöglicht es, eine Komponente einer anderen Anwendung einfach in der eigenen Anwendung zu verwenden.

Es gibt verschiedene Komponenten, je nach Typ ist der Lebenszyklus sowie die Verwendung unterschiedlich.

Folgend eine Auflistung der wichtigsten Komponenten:

Name	Beschreibung
Activity	Jeder, für den Benutzer sichtbare Screen basiert auf einer Activity. Die Activity agiert dabei als Controller für die View. Sie erhält die Events bei Userinteraktionen und kann diese entsprechend behandeln.
Service	Ein Service ist eine Hintergrundkomponente welche der Benutzer nie sieht. Dieser wird benötigt um Aufgaben im Hintergrund auszuführen. Der Service selbst stellt dabei keine asynchrone Funktionalität zur Verfügung. D.h. wenn der Programmierer nicht selbst einen Thread für die Hintergrundverarbeitung erstellt läuft diese im UI-Thread. Dies kann schnell zu einem Application Not Responding (ANR) führen.
BroadcastReceiver	Um zwischen Anwendungen oder mit dem System zu kommunizieren, gibt es Broadcasts. Diese funktionieren ähnlich wie im Netzwerk. Die Komponente die informieren will, sendet einen Broadcast an das System. Dieses leitet den Broadcast an alle Komponenten weiter, welche sich mit einem BroadcastReceiver beim System angemeldet haben. Ein Beispiel für einen solchen Broadcast ist die Information, dass eine SMS erhalten wurde.

Tabelle 18: Android Komponenten

12.5.2. Oberfläche

Die Oberfläche kann in Android komplett in XML erstellt werden. Dies ermöglicht eine komplette Trennung von Anzeige und Businesslogik.

12.5.3. Weitere Ressourcen

Um mehr über die Android-Plattform zu erfahren empfehlen wir folgende offizielle Dokumentation von Google:
http://developer.android.com/resources/index.html

Weiter sind die Videos der Google I/O sehr empfehlenswert:
http://developer.android.com/videos/index.html

12.6. Symbian Testfahrten

Testfahrt 1:
Datum: 04.03.2011
Auto: Toyota Aygo

Nachdem der ODB2 Adapter angeschlossen war, konnte die Anwendung gestartet werden. Das Erkennen der Schnittstelle verlief sehr zügig, jedoch wurde ich nicht über den Status informiert. Hier wäre eine Info an den User, dass die Verbindung zur ODB2 Schnittstelle aufgebaut ist, angebracht. Die Gangerkennung hat während dieser Testfahrt nicht funktioniert. Der 1. und der 2. Gang wurden nie erkannt, erst als ich im 3. Gang gefahren bin, hat mir die Anwendung angezeigt, dass ich im 1. Gang unterwegs sei. Die Verbrauchsdaten wurden ebenfalls nicht angezeigt, ich denke, dass diese Werte von der OBD2 Schnittstelle nicht geliefert werden. Des Weiteren gibt es keine Möglichkeit Fahrten bzw. Fahrer zu verwalten.

Gewisse Werte, die angezeigt wurden, sind nicht auf Anhieb klar und müssten aussagekräftiger beschrieben sein. Da die Werte rein numerisch dargestellt werden, ist es für den Fahrer relativ schwierig, ein informatives Feedback während der Fahrt zu erhalten. Hier wäre allenfalls eine graphische Darstellung zu bevorzugen.

Testfahrt 2:
Datum: 05.03.2011
Auto: Hyundai Tuxen 2.8l (Automat)

Während dieser Testfahrt wurden auch die Verbrauchswerte der ODB2 Schnittstelle aufgezeichnet. Diese Werte konnten eins zu eins mit dem Fahrstiel des Fahrers verglichen werden. D.h. sobald dieser auf der Autobahneinfahrt beschleunigt hat, sind die Verbrauchswerte rasant angestiegen. Während der Testfahrt versuchte ich, Einstellungen zu ändern, dies wird jedoch durch die Anwendung verhindert. Auch hier fehlt meiner Meinung nach eine Information an den Benutzer. Im Log konnte ich dann eine Meldung sehen, dass die Anwendung zuerst beendet werden muss, bevor Einstellungen verändert werden können. Meiner Meinung nach würde es jedoch noch Sinn machen, auch während einem Trip die Einstellungen anpassen zu können.

12.7. Requirements

12.7.1. Produkt Perspektive¶

Das Produkt EcoHelper soll als Expertenanwendung für Weiterbildungskurse (WAB) sowie für Privatpersonen als Fahrtenanalysewerkzeug genutzt werden. Es gibt keine Unterscheidung hinsichtlich der Funktionalität für die verschiedenen Einsatzzwecke.

Während und nach der Fahrt werden dem Benutzer Informationen zur Fahrt angezeigt. z.B. der aktuelle Gang oder die Geschwindigkeit. Für eine ausführlichere Analyse werden die Daten an die EcoHelper-Webseite übertragen.

12.7.2. Produkt-Funktionen

EcoHelper erhält von verschiedenen Geräten (externe – OBD2, sowie interne - Sensoren) Daten, die er bearbeitet und korrigiert, um diese dann auf einen Webserver zu übermitteln. Auf dem Smartphone werden nur die Resultate einer Fahrt gespeichert und dargestellt. Die Daten, welche hierzu gebraucht werden, werden nach erfolgreichem Transfer zum Server gelöscht, sofern sie nicht mehr relevant für weitere Berechnungen sind.

12.7.3. Benutzer Charakteristik

Die EcoHelper-Anwendung ist für die unten aufgeführten Zielgruppen gedacht. Dabei soll erwähnt werden, dass für alle Gruppen die gleiche Anwendung, mit den gleichen Funktionalitäten eingesetzt werden soll. Es wird kein besonderer Wert auf eine bestimmte Gruppe gelegt, sondern der Fahrer steht im Mittelpunkt.

- WAB-Kurse: Als Unterstützung zum Kurs und zum Kennenlernen der Anwendung.
- Fahrlehrer: Der Fahrlehrer zeigt seinen Fahrschülern den Nutzen, den er beim Gebrauch der Anwendung hat.
- Privatpersonen: Aus privatem Interesse oder weil sie es schon beim Fahrlehrer oder bei einem WAB-Kurs kennengelernt hatten.

Hinweis: Die Gruppen wurden aus der BA iPhone FS2010 übernommen.

12.7.4. Einschränkungen

Es sollen vorerst nur Fahrzeuge / Modelle unterstützt werden, welche in der OBD / Kennzahlen - Datenbank vorhanden sind. Fehlen nötige Informationen zur Berechnung von Analyse-Werten, werden nur jene genutzt und dargestellt, welche vorhanden sind. Zudem sollte die EcoHelper-Anwendung auch ohne OBD-Schnittstelleninformationen genutzt werden können, jedoch mit der Einschränkung, dass dann nicht alle Funktionalitäten nutzbar sein werden.

12.7.5. Annahmen

- Die Anpassungen auf der Serverseite ziehen keine grossen Änderungen im Webseitensystem mit sich.
- Die Anwendung liefert nur zu vorher konfigurierten Fahrzeugen (gespeicherten) genaue Werte.
- Es werden nur Autos mit Benzin-Motor unterstützt.
- Es werden sämtliche Getriebearten (Automat, Halbautomat, Handschaltung) unterstützt.

12.7.6. Spezifische Anforderungen

Ein grosser Teil der spezifischen Anforderungen lassen sich aus den Use Cases herleiten.

Länderspezifische Einstellungen

Es soll möglich sein die Anwendung in Deutsch und Englisch zu bedienen. Die Masseinheiten bleiben jedoch gleich, da vorerst nur für Europa entwickelt wird.

Datenablage

Der Benutzer wird durch das Speichern und Uploaden der Daten nicht gestört.

- Anwendung darf nicht blockieren wenn Daten gespeichert werden
- Falls Server nicht verfügbar Daten zwischenspeichern
- Upload der Daten im Hintergrund

Online- / Offline-Modus

Online

Die Daten werden direkt an den Webserver gesendet und dann auf dem Gerät gelöscht.

Offline

Die Daten werden in einer lokalen SQLite-Datenbank abgelegt und zu einem späteren Zeitpunkt dem Webserver übermittelt.

- Wie sollen die Daten übermittelt werden?
- Weitere Exportfunktionen erwünscht?

Android Version

EcoHelper für Android wird für die Android Version 2.2 und höher entwickelt.

12.7.7. Nichtfunktionale Anforderungen

12.7.8. Zuverlässigkeit

Die Anwendung soll so zuverlässig funktionieren, dass sie für den produktiven Einsatz bestimmt ist.

- Keine programmabstürze bei der Durchführung der Use Cases.

12.7.9. Aussehen und Handhabung

- Für mehr als 50% der Testpersonen soll die Anwendung ansprechend aussehen.
- Android Style Guide wird eingehalten

12.7.10. Benutzbarkeit

- Verständlich (mehr als ¾ aller Testpersonen haben keine Probleme bei der Nutzung)
- Sämtliche Hauptfunktionen sind mit maximal vier Klicks zu erreichen.
- Sämtliche Funktionen sind in der Hilfe vollständig dokumentiert, wodurch der Benutzer ohne externe Hilfe die Anwendung bedienen kann.
- ¾ der Testpersonen sind durch die Rückmeldungen der Anwendung genügend über deren aktuellen Status informiert.
- Fahrer nicht ablenken, sondern unterstützen (Keine Eingaben durch den Fahrer während einer Fahrt nötig)

12.7.11. Leistung und Effizienz

- Keine ANR bei der Durchführung der Use Cases.

- Auf den zur Verfügung stehenden Testgeräten ist bei der Durchführung der Use Cases kein Ruckeln festzustellen.

12.7.12. Wartbarkeit

- Gut dokumentiert (entsprechend den Anforderungen einer BA)

12.7.13. Portierbarkeit

Die Anwendung läuft auf Android 2.2 und neuer. Es werden keine weiteren Systeme unterstützt.

12.7.14. Sicherheitsanforderungen

- Daten werden immer einem Benutzer zugeordnet
- Benutzer hat keinen Zugriff auf Daten eines anderen Benutzers.
- Daten werden vor Zugriff anderer Anwendungen auf dem gleichen Gerät geschützt

12.7.15. Flexibilität

- Standards verwenden (vorhandene Libraries, Komponenten)

12.8. Use Cases

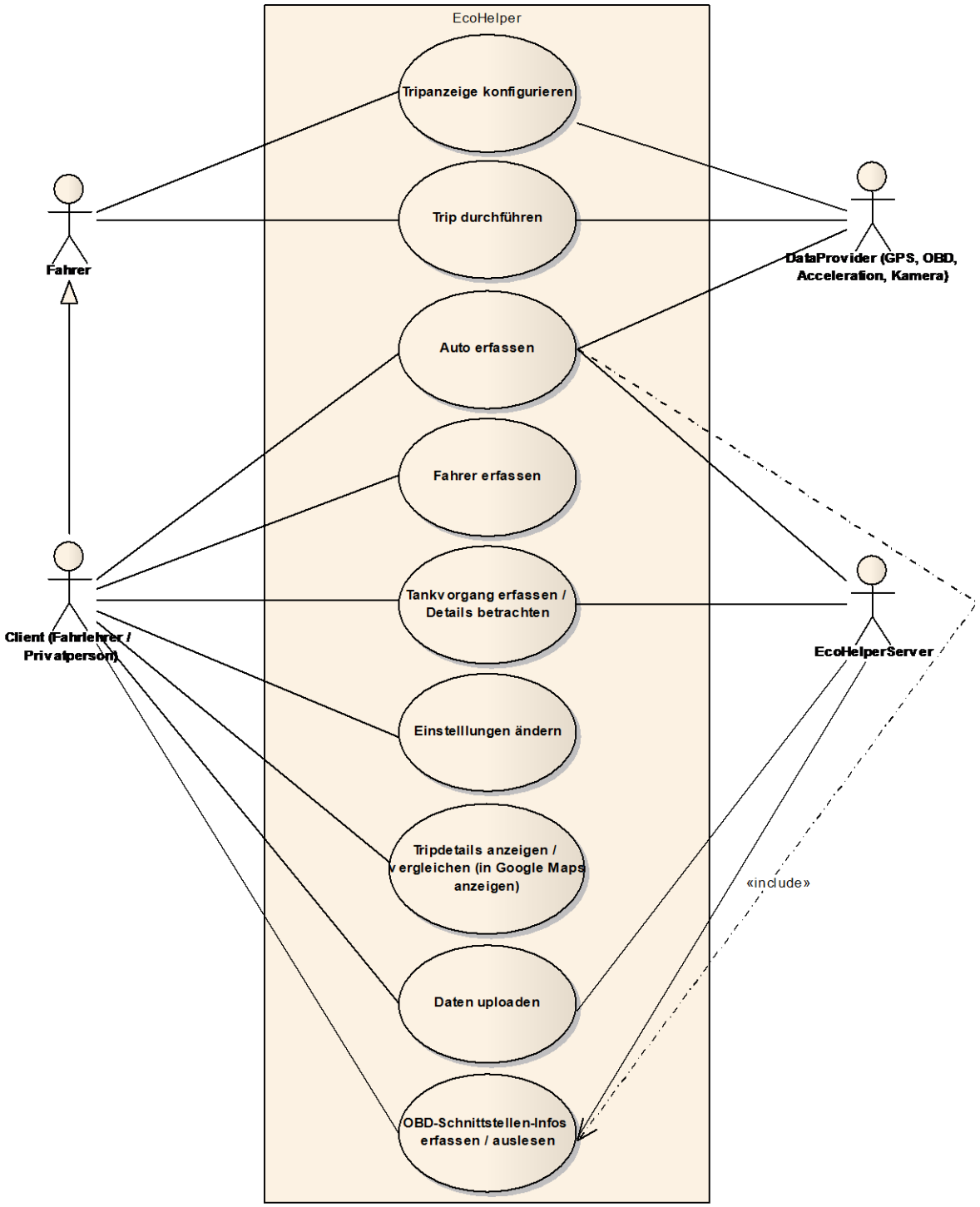


Abbildung 38: Use Case Diagramm

Die obige Grafik zeigt alle Use Cases und die involvierten Akteure.

12.8.1. UC 01: Auto erfassen

Primärakteur: Client (Fahrlehrer)
Interesse & Absicht: Erstellen von Auto mit Angabe von Details

Der User klickt in der Autoverwaltung im Menü auf „Auto hinzufügen“. Der Server wird kontaktiert und eine Liste mit den vorhandenen Autoherstellern wird geladen und angezeigt. Der User wählt einen Autohersteller aus. Der Server wird wiederum kontaktiert und es wird eine Liste mit den vorhandenen Automodellen zu dem gewählten Autohersteller geladen. Nachdem das Modell ausgewählt wurde, wird eine Liste mit den Varianten zu diesem Modell vom Server geladen. Nach der Variantenauswahl erhält der User die Möglichkeit, Detailangaben zum Auto zu erfassen. Dafür können folgende Werte angegeben werden: (***Pflichtfeld**)

- **Aufbau***
- **Getriebeart***
- **Anzahl Gänge***
- **Treibstoff***
- **Name***
- Hubraum
- Gewicht
- Jahrgang
- Maximale Reichweite pro Tankfüllung

Nach der Angabe der Werte kann der User mittels Klick auf „Speichern“ das Auto zum Server übertragen. Anschliessend steht das Auto in der Autoverwaltung zur Verfügung.

12.8.2. UC 02: OBD2-Infos erfassen

Primärakteur: Client (Fahrlehrer)
Interesse & Absicht: Anschluss für OBD2 Schnittstelle in Datenbank eintragen

Der User Befindet sich in den Autodetails zu seinem Auto. Nach einem Klick auf OBD2 Anschlussplatz wird der Server kontaktiert und gefragt, ob bereits OBD2 Informationen zu dem gewählten Auto vorhanden sind. Falls dem so ist, werden die Informationen inkl. Bild angezeigt. Um einen neuen Anschlussplatz zu erfassen, klickt der User auf Neuer Anschlussplatz erfassen. Zu Beginn wählt er eine von 9 vorgegebenen Positionen aus, an der sich die Schnittstelle befindet. Anschliessend kann er mit einem Klick auf „Foto aufnehmen“ ein Foto des Anschlusses machen. Zum Schluss kann er eine Beschreibung angeben und durch einen Klick auf „Upload“ die Daten an den Server übertragen. Von nun an steht der Eintrag in den OBD2 Informationen zu diesem Auto zur Verfügung.

12.8.3. UC 03: Fahrer erfassen

Primärakteur: Client (Fahrlehrer)
Interesse & Absicht: Trips unterschiedlichen Fahrern zuweisen.

Der User kann über die Fahrerverwaltung seine benötigten Fahrer erfassen. Dazu klickt er im Menü auf „Fahrer hinzufügen“. Dann gibt er den gewünschten Namen und eine E-Mailadresse an. Nach dem Klick auf „Erstellen“ werden die Daten an den Server übertragen. Somit steht der erstellte Fahrer zur Verfügung.

12.8.4. UC 04: Tripanzeige konfigurieren

Primärakteur: Fahrer
Interesse & Absicht: Tripanzeige nach persönlichen Bedürfnissen anpassen

Der User befindet sich in der Tripaufzeichnung. Mit einem Klick auf Menü/Anzeige hinzufügen gelangt der User in die Auswahl des benötigten Anzeigeelementes. Er kann zwischen Einzelparameteranzeige, Multiparameteranzeige und Button wählen. Mit einem Klick auf Einzelparameteranzeige gelangt er in eine weitere Auswahl. Er kann zwischen den Datenquellen Ortungsdienste, OBD2 und Sensoren wählen. Mit einem Klick auf OBD2 gelangt er in die Auswahl der verfügbaren OBD2 Parameter. Mit einem Klick auf Distanz kommt er in die Grössenauswahl. Dabei kann er auswählen, ob die gemäss OBD2 gefahrene Distanz als kleines oder als grosses Element angezeigt werden soll. Nach dem Klick auf Klick auf klein wechselt das System wieder in die Tripaufzeichnungsansicht. Das gewählte Element befindet sich nun links oben in der Anzeige. Durch einen Longclick kann es auf der Anzeige platziert oder auch wieder entfernt werden.

12.8.5. UC 05: Trip durchführen

Ebene	Anwenderziel
Primärakteur	User (Privatperson)
Überblick	Trip durchführen
Interesse & Absicht	Neuer Trip erstellen und durchführen Messwerte während der Fahrt aufzeichnen und anzeigen Tripzusammenfassung betrachten Messwerte an den Server übertragen
Vorbedingung	▪ Der User hat sich erfolgreich angemeldet und befindet sich auf der Hauptübersichtsseite ▪ Fahrer wurde erstellt und synchronisiert ▪ Auto wurde erstellt und synchronisiert ▪ Bluetooth ist aktiviert ▪ Das Pairing mit dem Bluetooth Adapter wurde erfolgreich durchgeführt ▪ Die Lokalisierung via GPS ist aktiviert ▪ Autoupload ist aktiviert ▪ GUI Elemente wurden bereits nach Bedarf angeordnet
Nachbedingung	Die Messwerte wurden komplett zum Server übertragen. Der User erhält eine Übersichtsseite mit den gemessenen Werten.

Standardablauf	User	System	Server
	1. User klickt auf Neuer Trip		
		2. System zeigt Trip Erfassungsdialog an	
	3. User wählt Fahrer aus		
	4. User wählt Auto aus		
		5. System zeigt Zustand der Messschnittstellen wie Ortung via GPS oder WiFi an. Ebenfalls wird angezeigt, ob Bluetooth aktiviert ist und wie der Status der Kommunikation mit OBD2 Schnittstelle ist.	
	6. User klickt auf Trip starten		
		7. System wechselt zur Trip-Messwert Anzeige	
		8. Die Aufzeichnung folgender Messwerte wird gestartet: Lokationsdaten (GPS/Mobilfunk Antennen), OBD2-Messwerte, Werte der Beschleunigungssensoren sowie die Mobilfunkwerte (Signalstärke, verbundene Antenne, Mobilfunkanbieter usw.)	
		9. Messwerte werden dem User auf drei Screens angezeigt, diese können dynamisch angepasst werden	
		10. Messwerte werden an den Server übertragen	
	12. User fährt los		11. Server empfängt Messwerte
	13. User beendet den Trip durch Klick auf Backbutton		
	15. User bestätigt mit Klick auf Ok		
		14. System fragt den User, ob Trip tatsächlich beendet werden soll	

<table>
<tr><td></td><td>17. Durch Betätigung der Backbuttontaste gelangt der User wieder zurück zur Triperfassungsseite</td><td>16. System überträgt die übrigen Messwerte und zeigt dies mittels Progress bar an. Nach erfolgreicher Übertragung wird eine Tripzusammenfassung mit folgenden Werten angezeigt:
<table>
<tr><td colspan="2">Tripdaten</td></tr>
<tr><td>Auto</td><td>Userauswahl</td></tr>
<tr><td>Datum</td><td>Systemzeit</td></tr>
<tr><td>Distanz</td><td>GPS</td></tr>
<tr><td>Distanz</td><td>OBD2</td></tr>
<tr><td colspan="2">Fahrdaten</td></tr>
<tr><td>Max. Geschwindigkeit</td><td>OBD2</td></tr>
<tr><td>Avg. Geschwindigkeit</td><td>OBD2</td></tr>
<tr><td>Max. RPM</td><td>OBD2</td></tr>
<tr><td>Avg. RPM</td><td>OBD2</td></tr>
<tr><td>Schaltvorgänge</td><td>OBD2</td></tr>
<tr><td colspan="2">Ökodaten</td></tr>
<tr><td>Verbrauch Total</td><td>OBD2</td></tr>
<tr><td>Avg. Verbrauch</td><td>OBD2</td></tr>
<tr><td>Eco-Zahl</td><td>OBD2</td></tr>
</table></td><td></td></tr>
<tr><td>Erweiterungen</td><td colspan="3">Offline Modus</td></tr>
<tr><td></td><td>User</td><td>System</td><td>Server</td></tr>
<tr><td></td><td>17.a Durch Klick auf Messdaten übertragen startet der User die Datenübertragung</td><td>10.a Messdaten werden lokal gespeichert
16.a System zeigt Tripzusammenfassung an
18.a Fortschritt der Datenübertragung wird angezeigt</td><td>19.a Server empfängt Messdaten</td></tr>
<tr><td>Erweiterungen</td><td colspan="3">Unterbruch der OBD2 Verbindung</td></tr>
<tr><td></td><td>User</td><td>System</td><td>Server</td></tr>
<tr><td></td><td></td><td>8. b Es wird ein Warnton abgespielt, der auf den</td><td></td></tr>
</table>

		Unterbruch hinweist. Nebst den OBD2 Daten werden alle weiteren Daten aufgezeichnet. 9. b Anstelle der OBD2 Messwerte werden nun Striche angezeigt, die symbolisieren, dass Momentan keine Werte verfügbar sind. Sobald die Verbindung wieder funktioniert, werden die aktuellen Messwerte wieder aufgezeichnet und angezeigt.	
Erweiterungen	Unterbruch der GPS Verbindung		
	User	System	Server
		8. c Nebst den GPS Daten werden alle weiteren Daten aufgezeichnet. 9. c Anstelle der GPS Messwerte werden nun Striche angezeigt, die symbolisieren, dass Momentan keine Werte verfügbar sind. Sobald die Verbindung wieder funktioniert, werden die aktuellen Messwerte wieder aufgezeichnet und angezeigt.	
Erweiterungen	Unterbruch des Uploads		
	User	System	Server
	17. d User klickt auf Abbrechen während dem Upload der Daten.	18. d System wechselt zur Tripzusammenfassung. Upload der Daten kann durch Klick auf „Upload zum Server“ fortgesetzt werden.	
Erweiterungen	Anwendung verliert den Fokus		
	User	System	Server
	13.e User klickt auf Home Button 15.e User öffnet EcoHelper wieder	14.e System wechselt auf Homescreen. Die Datenaufnahme läuft im Hintergrund weiter. 16.e System zeigt wieder Tripmesswert Anzeige an.	
Erweiterungen	Exception tritt auf		
	User	System	Server
		18.f Es tritt eine unerwartete	

		Exception auf. System Zeigt dem User, dass ein Problem aufgetreten ist. Stacktrace wird für Fehleranalyse an den Server übertragen. Programm wird an letzter Stelle wieder ausgeführt.	19.f Server empfängt Stacktrace.

Tabelle 19: Use Case 05, Trip durchführen

12.8.6. UC 06: Tripdetails & Trip in Google Maps anschauen

Primärakteur: Client (Fahrlehrer)
Interesse & Absicht: Details zu gefahrenem Trip anzeigen und in Google Maps betrachten

Der User befindet sich auf dem Hauptscreen. Er klickt auf Tripübersicht. Die Trips werden synchronisiert und in der Liste angezeigt. Mit einem Klick auf den gewünschten Trip gelangt er in die Tripdetails. Es werden folgende Werte zum Trip angezeigt:

Tripdaten	
Auto	Userauswahl
Datum	Systemzeit
Distanz	GPS
Distanz	OBD2
Fahrdaten	
Max. Geschwindigkeit	OBD2
Avg. Geschwindigkeit	OBD2
Max. RPM	OBD2
Avg. RPM	OBD2
Schaltvorgänge	OBD2
Ökodaten	
Verbrauch Total	OBD2
Avg. Verbrauch	OBD2
Eco-Zahl	OBD2

Tabelle 20: Use Case 06, Tripdetails, Angezeigte Parameter

Mit einem Klick auf „Trip in Google Maps anzeigen“ erhält der User einen Kartenausschnitt von Google Maps. Der durchgeführte Trip wird durch eine rote Linie dargestellt.

12.8.7. UC 07: Trips vergleichen

Primärakteur: Client (Fahrlehrer)
Interesse & Absicht: Mehrere gefahrene Trips miteinander vergleichen

Der User klickt in der Tripübersicht auf den Menüeintrag „Trip analysieren“. Dann wählt er die gewünschten Trips, die er vergleichen möchte, aus einer Liste aus. Nach einem Klick auf „Analysieren“ werden die Daten an den Server übertragen und es wird eine Webview geöffnet, die die Trips in der Analyseansicht auf dem Server anzeigt.

12.8.8. UC 08: Tankvorgang erfassen

Primärakteur: Client (Privatperson)
Interesse & Absicht: Tankverhalten protokollieren

Der User klickt in der Tankverwaltung auf den Menüeintrag „Tankvorgang erfassen“. Dann muss er folgende Angaben machen:

- Auto
- Treibstoffart
- Tachostand
- Getankte Menge
- Gesamtkosten
- Vollgetankt

Optionale Angaben:

- Bemerkungen
- Durchschnittsverbrauch
- Durchschnittsgeschwindigkeit

Dann klickt der User auf „Speichern“. Damit werden die Daten an den Server übertragen und stehen dann in der Tankverwaltung zur Verfügung.

12.8.9. UC 09: Tankvorgangdetails betrachten

Primärakteur: Client (Privatperson)
Interesse & Absicht: Tankverhalten analysieren

Der User klickt in der Tankverwaltung auf den gewünschten Tankvorgang. Dann erhält er eine Übersicht mit den erfassten Daten. Zusätzlich werden der durchschnittliche Verbrauch und die gefahrene Distanz berechnet und angezeigt. Falls vorhanden, wird die Tankposition in Google Maps angezeigt.

12.9. GUI Entwicklung

In diesem Kapitel wird der Entwicklungsprozess des Anwendungsflusses und des GUIs beschrieben.

12.9.1. Paper Prototype

In einer frühen Phase des Projektes wurde ein ausführlicher Paper Prototype erstellt. Dieser wurde als Diskussionsgrundlage für die Entwicklung von EcoHelper verwendet. Ebenfalls wurden die Navigation und die wichtigsten GUI Elemente anhand des Paperprototypes mit Herrn Bader besprochen. Der Paper Prototype kann in digitalisierter Form auf der CD betrachtet werden.

12.9.2. Implementiertes GUI

LoginActivity

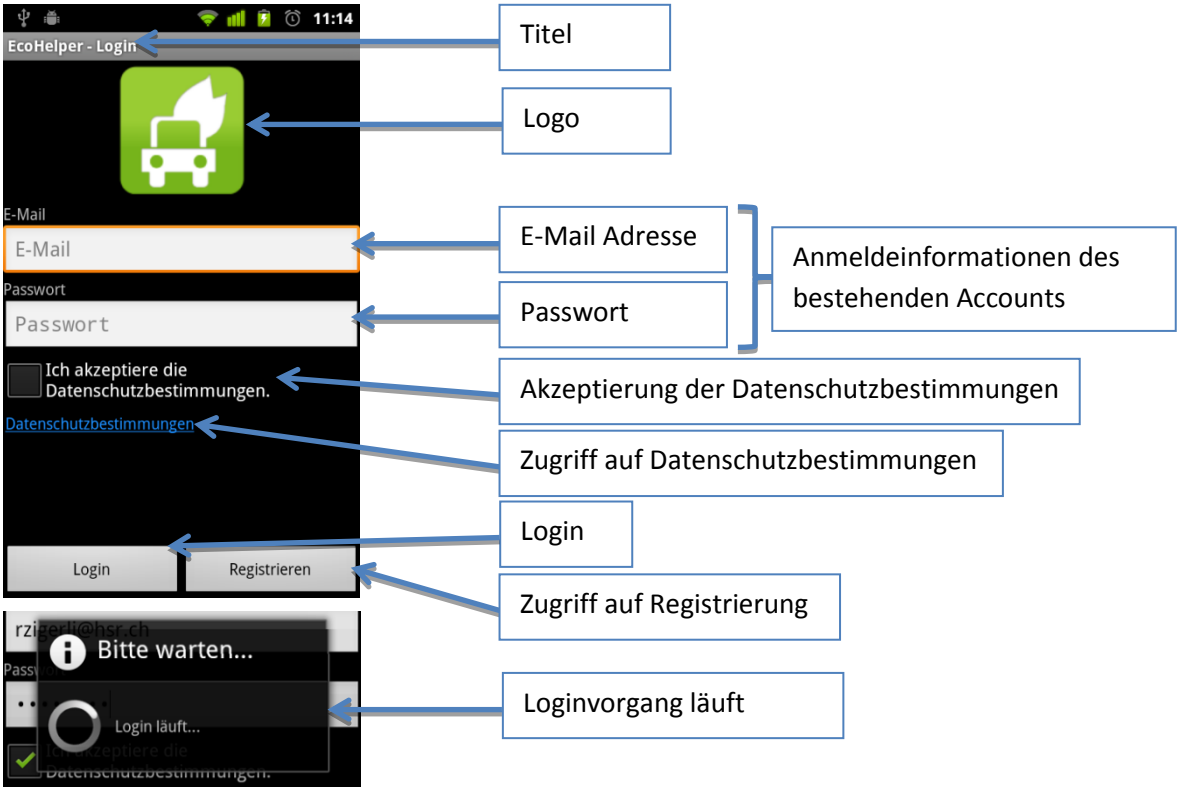


Abbildung 39: Loginbildschirm

RegisterActivity

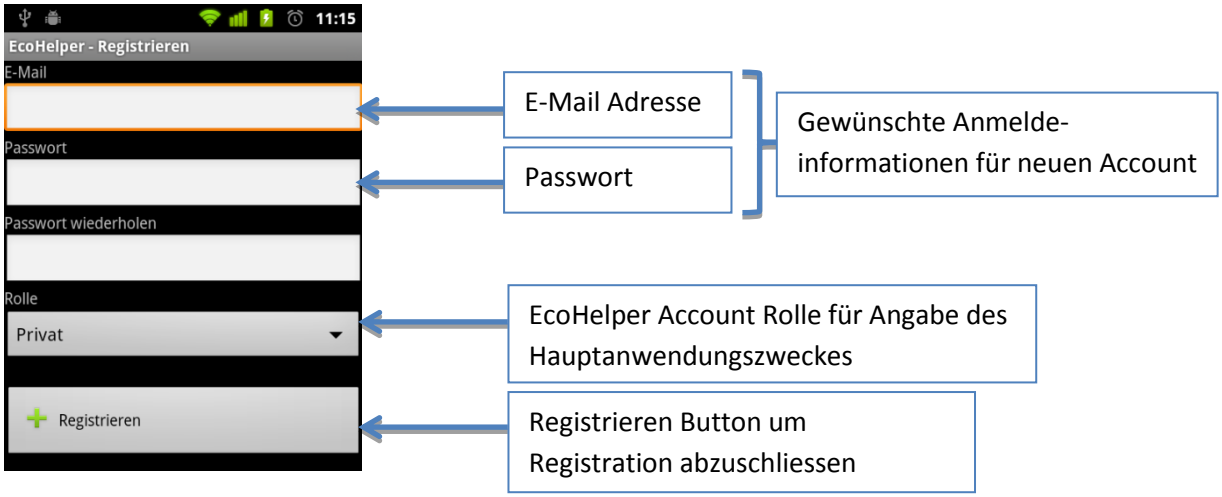


Abbildung 40: Registrierungsbildschirm

MainActivity

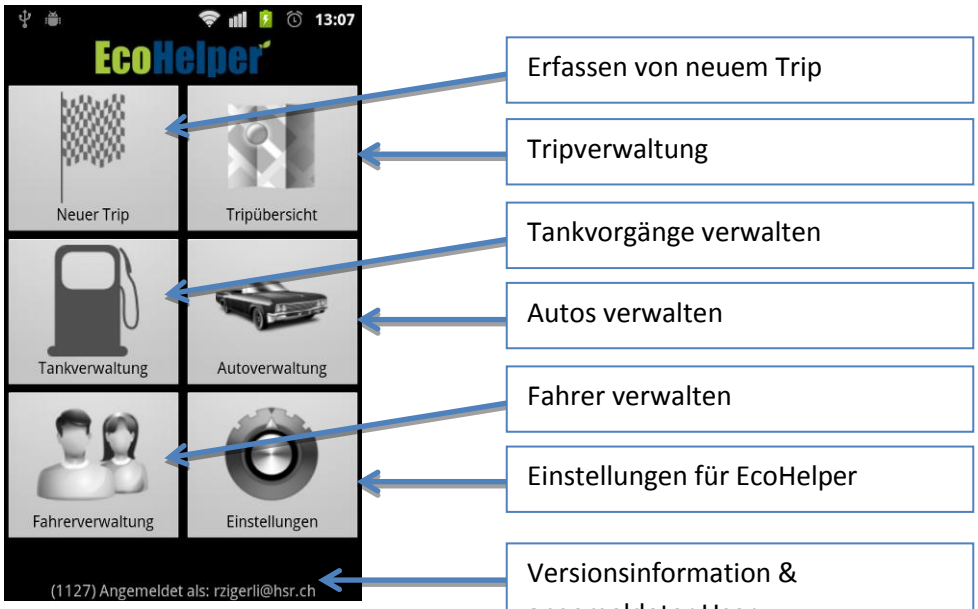


Abbildung 41: Hauptbildschirm

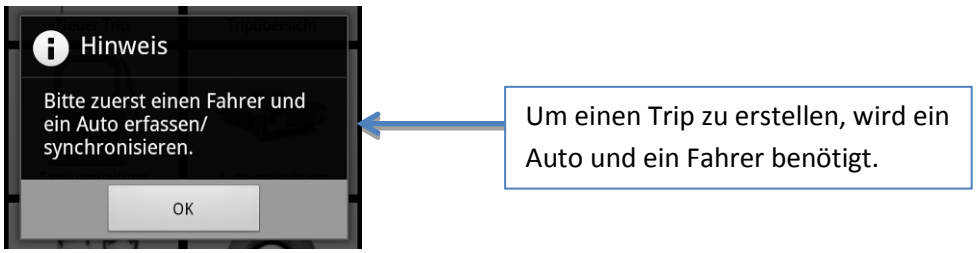
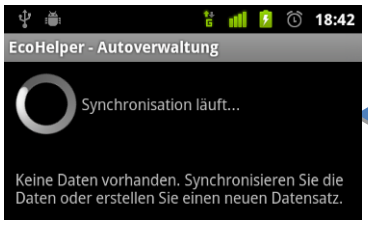


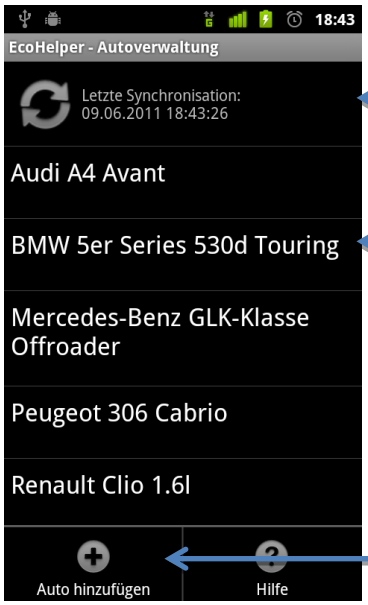
Abbildung 42: Hinweis für Erstellung/Synchronisation von Fahrer/Auto

CarManagementActivity



Synchronisation startet, sobald in Autoverwaltung gewechselt wird. Solange noch keine Daten vorhanden sind, wird dies angezeigt.

Abbildung 43: : Progressbar während Synchronisation von Autos



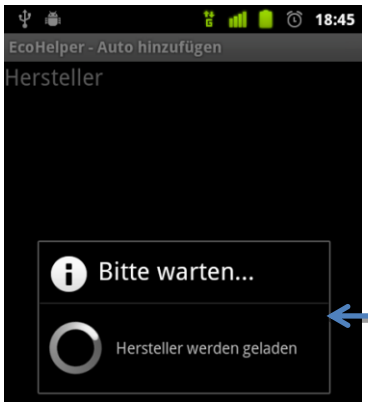
Durch Klick auf dieses Listenelement kann die Synchronisation manuell gestartet werden.

Sobald die Autos synchronisiert wurden, werden diese in der Liste angezeigt.

Über den Menübutton, „Auto hinzufügen“ kann ein neues Auto erfasst werden.

Abbildung 44: Liste mit bereits erfassten Autos

AddCarActivity



Nach dem Klick auf „Auto hinzufügen“ werden die Hersteller vom Server geladen. Dies wird durch eine Progressbar angezeigt.

Abbildung 45: : Progressbar während Laden von Herstellern

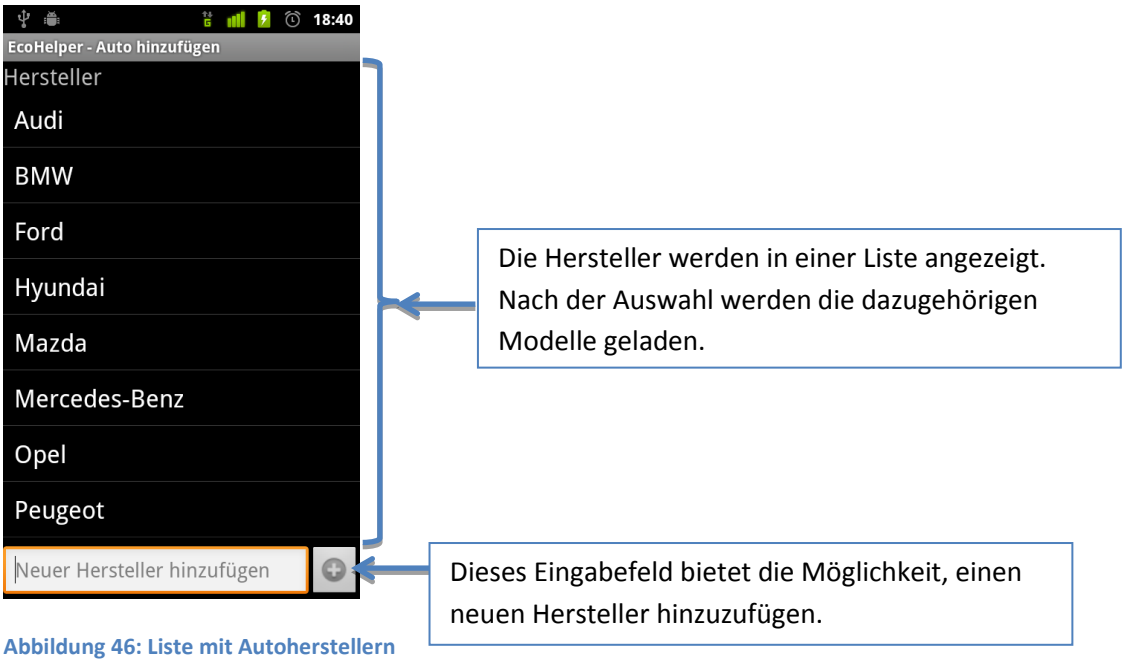


Abbildung 46: Liste mit Autoherstellern

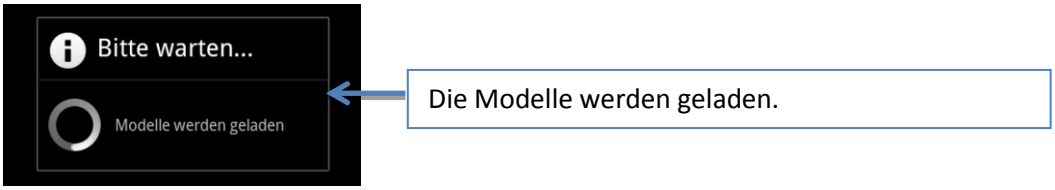


Abbildung 47: Progressbar während Laden von Modellen

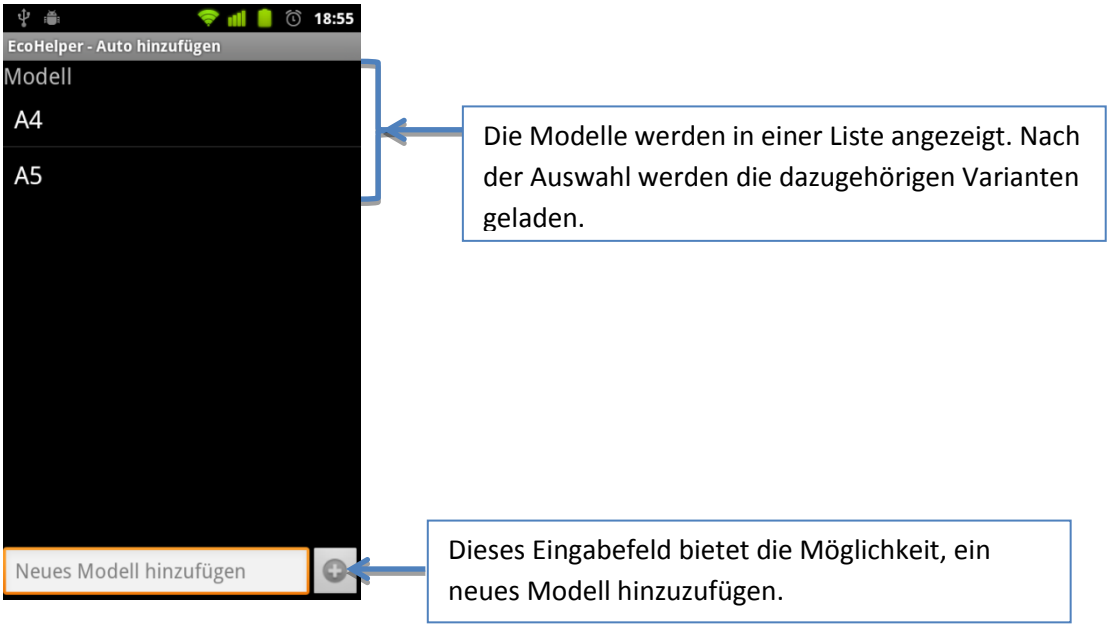


Abbildung 48: Liste mit Modellen

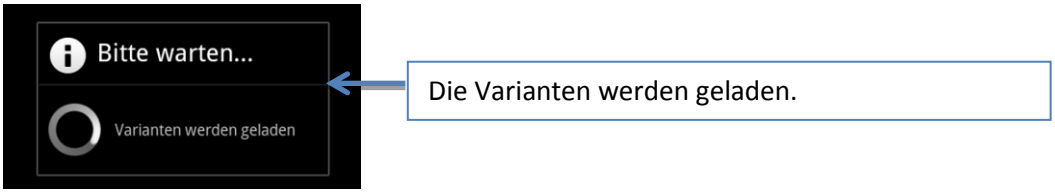
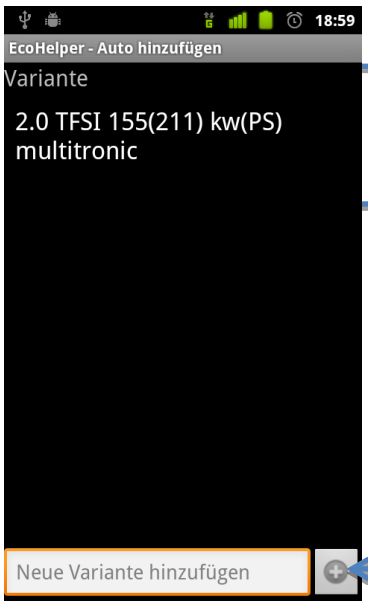


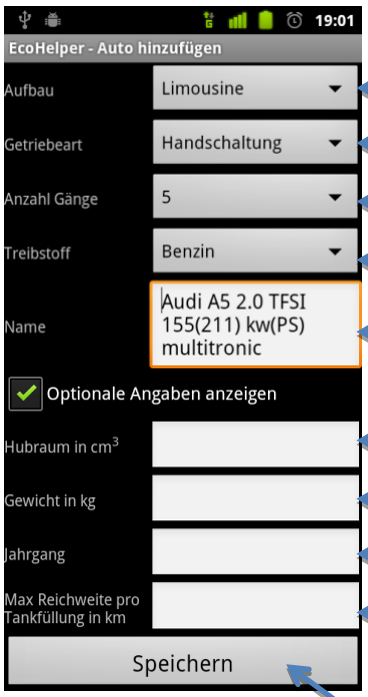
Abbildung 49: Progressbar während Laden von Varianten



Die Varianten werden in einer Liste angezeigt. Nach der Auswahl können Detailangaben getätigt werden.

Dieses Eingabefeld bietet die Möglichkeit, eine neue Variante hinzuzufügen.

Abbildung 50: Liste mit Varianten



Angabe des Aufbaus

Angabe der Getriebeart

Angabe der Anzahl Gänge

Angabe des Treibstoffs

Angabe des Namens

Angabe des Hubraums

Angabe des Gewichtes

Angabe des Jahrgangs

Angabe der maximalen Reichweite pro Tankfüllung in km

Optionale Angaben

Auto speichern

Abbildung 51: Angabe von Autodetails

CarDetailActivity

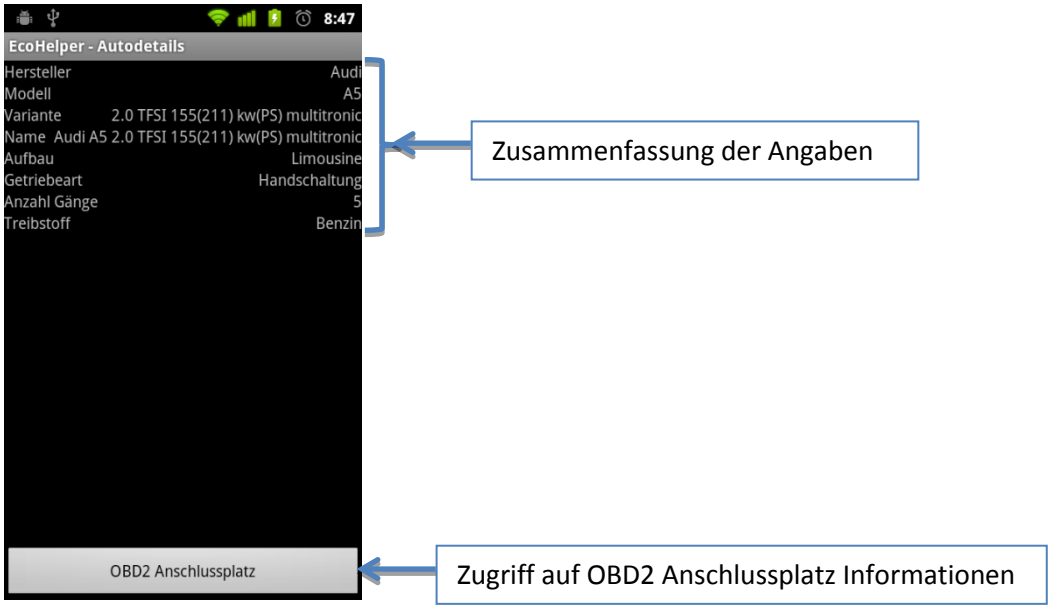


Abbildung 52: Erfasste Autodetails

ObdInfoActivity

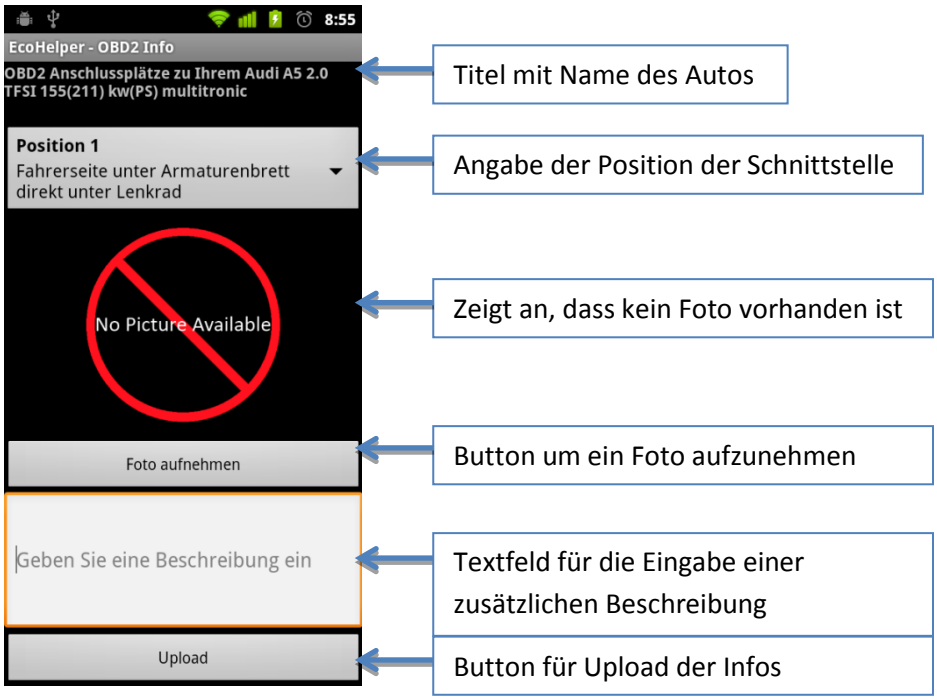


Abbildung 53: Erfassen von OBD Anschlussplatz

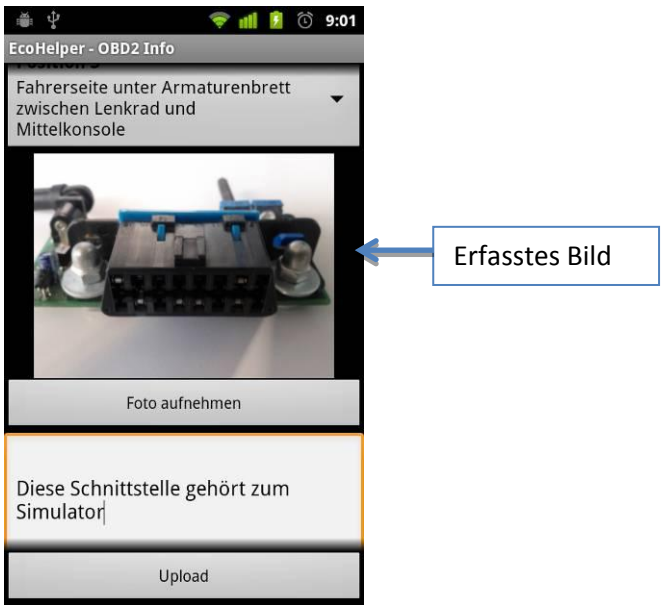


Abbildung 54: Erfassen von OBD2 Anschlussplatz

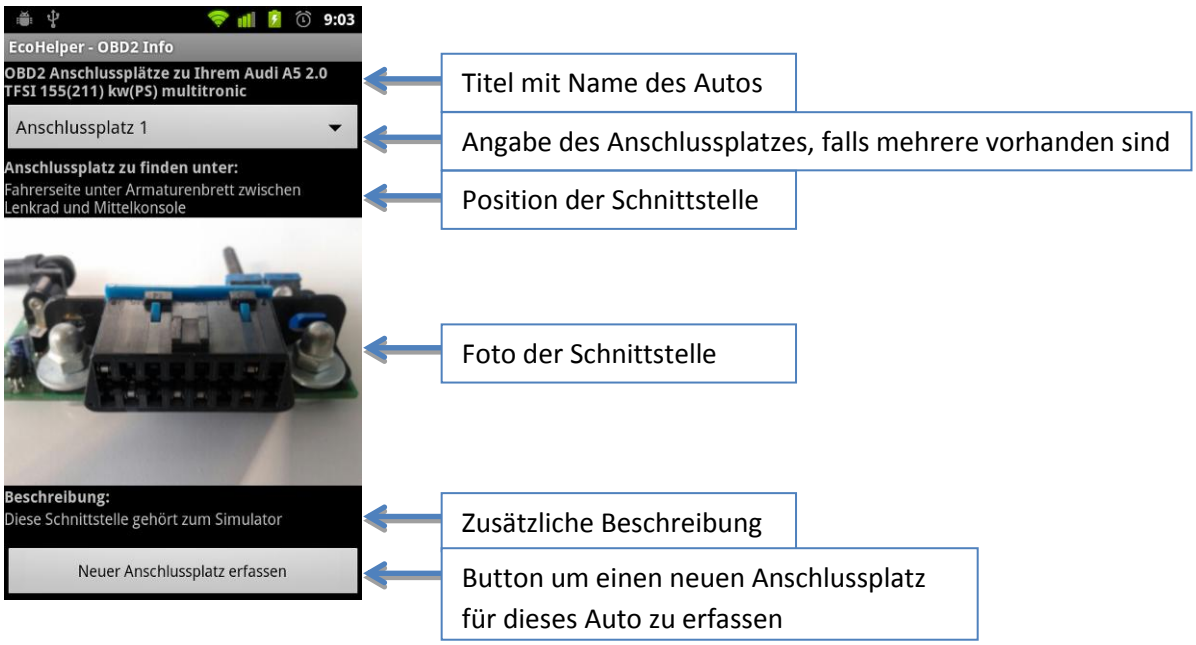


Abbildung 55: Erfasster OBD2 Anschlussplatz

DriverManagementActivity

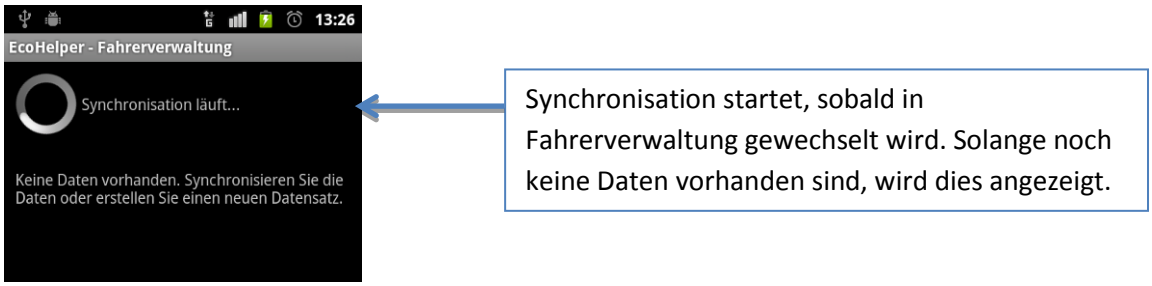


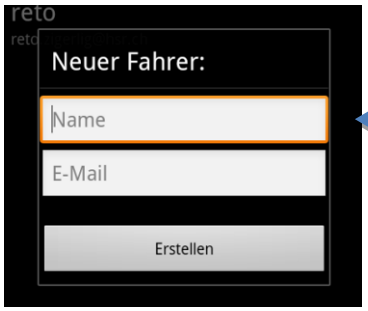
Abbildung 56: : Progressbar während Synchronisation von Fahrern

Durch Klick auf dieses Listenelement kann die Synchronisation manuell gestartet werden.

Sobald die Fahrer synchronisiert wurden, werden diese in der Liste angezeigt.

Um einen neuen Fahrer hinzuzufügen kann über das Menü auf „Fahrer hinzufügen“ geklickt werden.

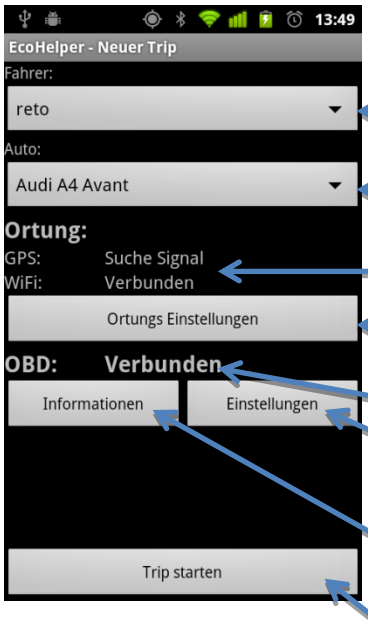
Abbildung 57: Liste mit Fahrern



Erfassung von neuem Fahrer. Angabe von Name und gültiger E-Mail Adresse

Abbildung 58: Angaben zu neuem Fahrer

NewTripActivity



Auswahl von Fahrer

Auswahl von Auto

Status von GPS- und WiFi Ortung

Zugriff auf Android Einstellungen für Standort & Sicherheit

Status von OBD2 Adapter Verbindung

Zugriff auf EcoHelper Einstellungen, um Bluetooth zu konfigurieren

Zugriff auf OBD2 Anschlussplatz Informationen

Button um Tripaufzeichnung zu starten

Abbildung 59: Neuer Trip starten

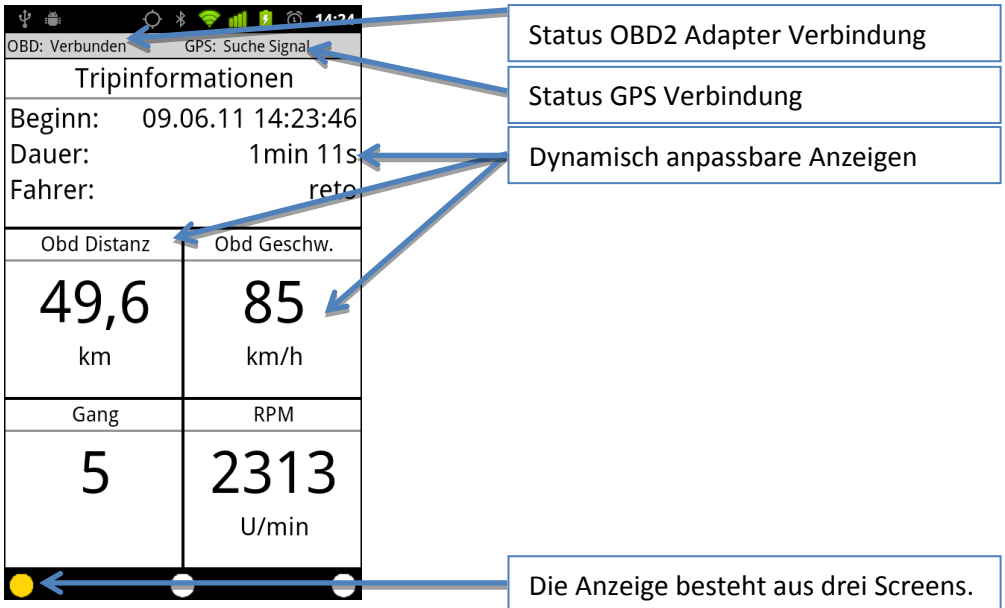


Abbildung 60: Ansicht während dem Trip

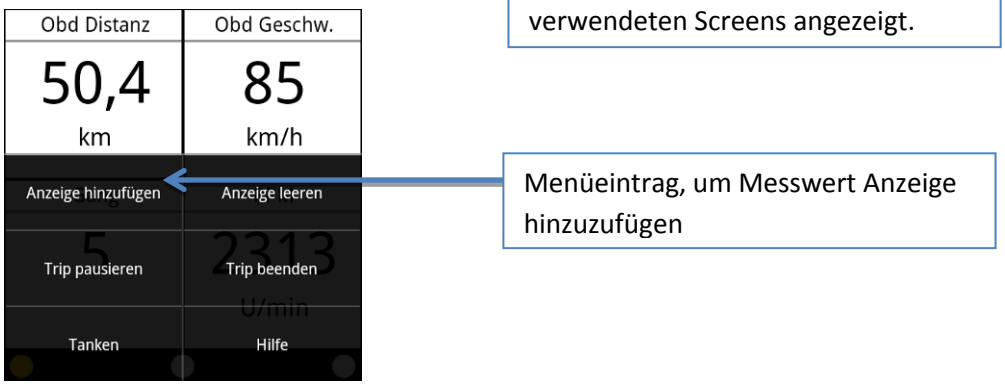


Abbildung 61: Menü während Trip

Abbildung 62: Anzeige während Trip hinzufügen

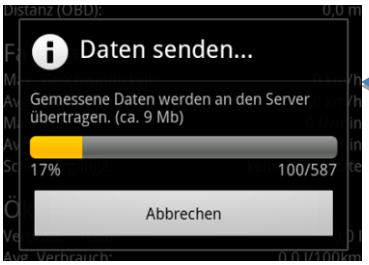
TripDetailActivity

Trip-Detail Angaben. Falls es für einen Wert zu wenige Messwerte gibt, wird dies angezeigt.

Falls noch Tripdaten lokal vorhanden sind, können diese über diesen Button an den Server übertragen werden.

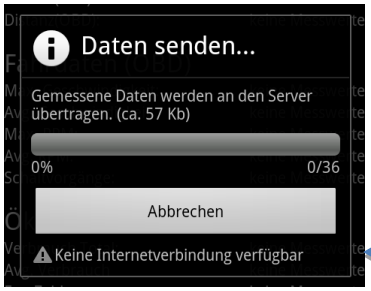
Wenn während dem Trip Ortungsdaten aufgezeichnet wurden, kann der Trip über diesen Button auf Google Maps angezeigt werden.

Abbildung 63: Ansicht der Tripdetails



Progressbar für Übertagung der Tripdaten an den Server. Mittels Abbrechen kann dieser Vorgang unterbrochen werden.

Abbildung 64: Progressbar während Upload von Tripdaten



Wenn während dem Upload keine Internetverbindung vorhanden ist, wird das angezeigt.

Abbildung 65: Keine Internet-verbindung während Upload

TripManagementActivity

Abbildung 66: Tripverwaltung mit gefahrenen Trips

TripsCompareActivity

Abbildung 67: Auswahl um Trips zu vergleichen

RefuelManagementActivity

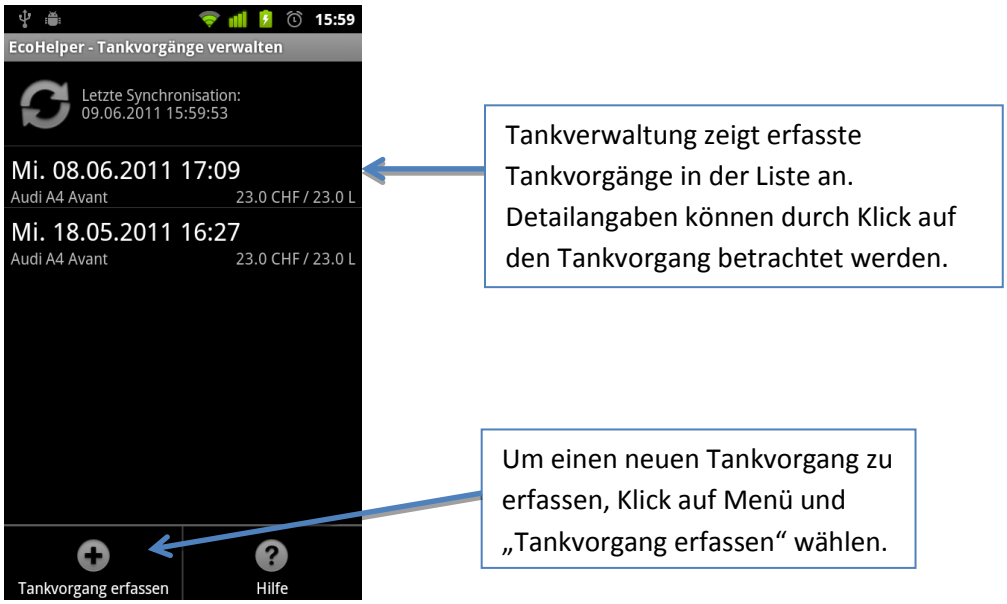


Abbildung 68: Tankverwaltung mit erfassten Tankvorgängen

AddRefuelActivity

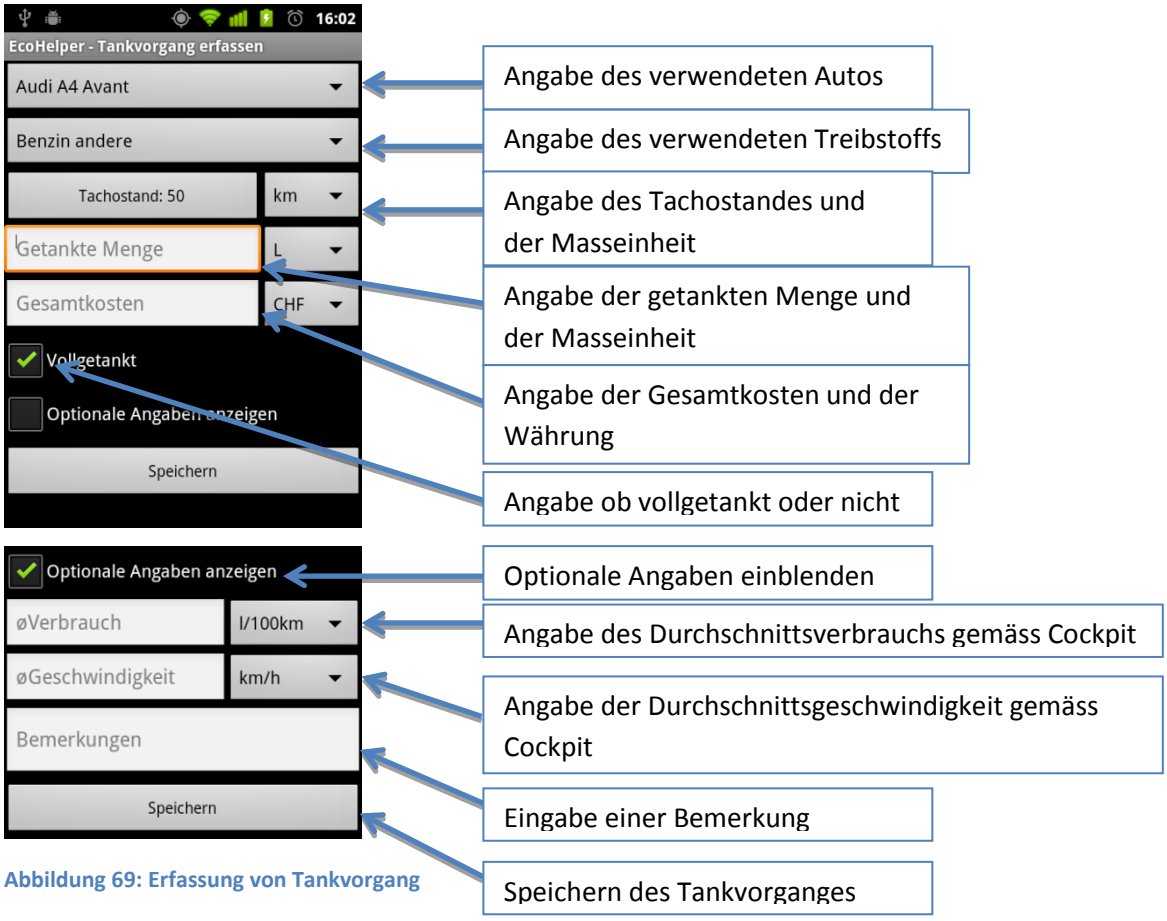


Abbildung 69: Erfassung von Tankvorgang

RefuelManagementDetailActivity

Abbildung 70: Details zu Tankvorgang

PreferencesActivity

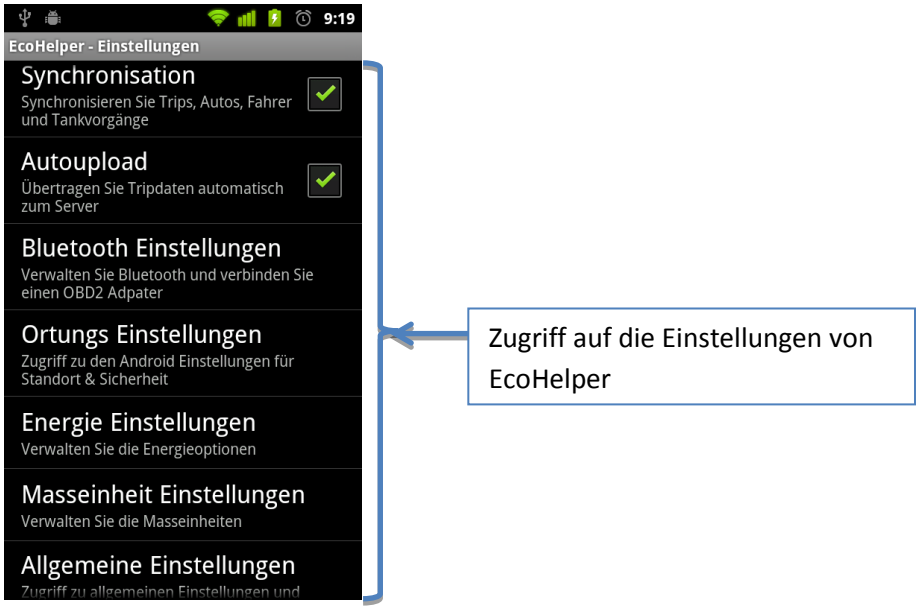


Abbildung 71: Ansicht der Einstellungen

12.9.3. Activity Diagramm

Zeigt den Fluss des Benutzers durch die Activities. Vorwärts-Navigationen werden durch schwarze Pfeile dargestellt. Wenn der Benutzer den Back-Button betätigt wird dies durch einen grüner Pfeil dargestellt.

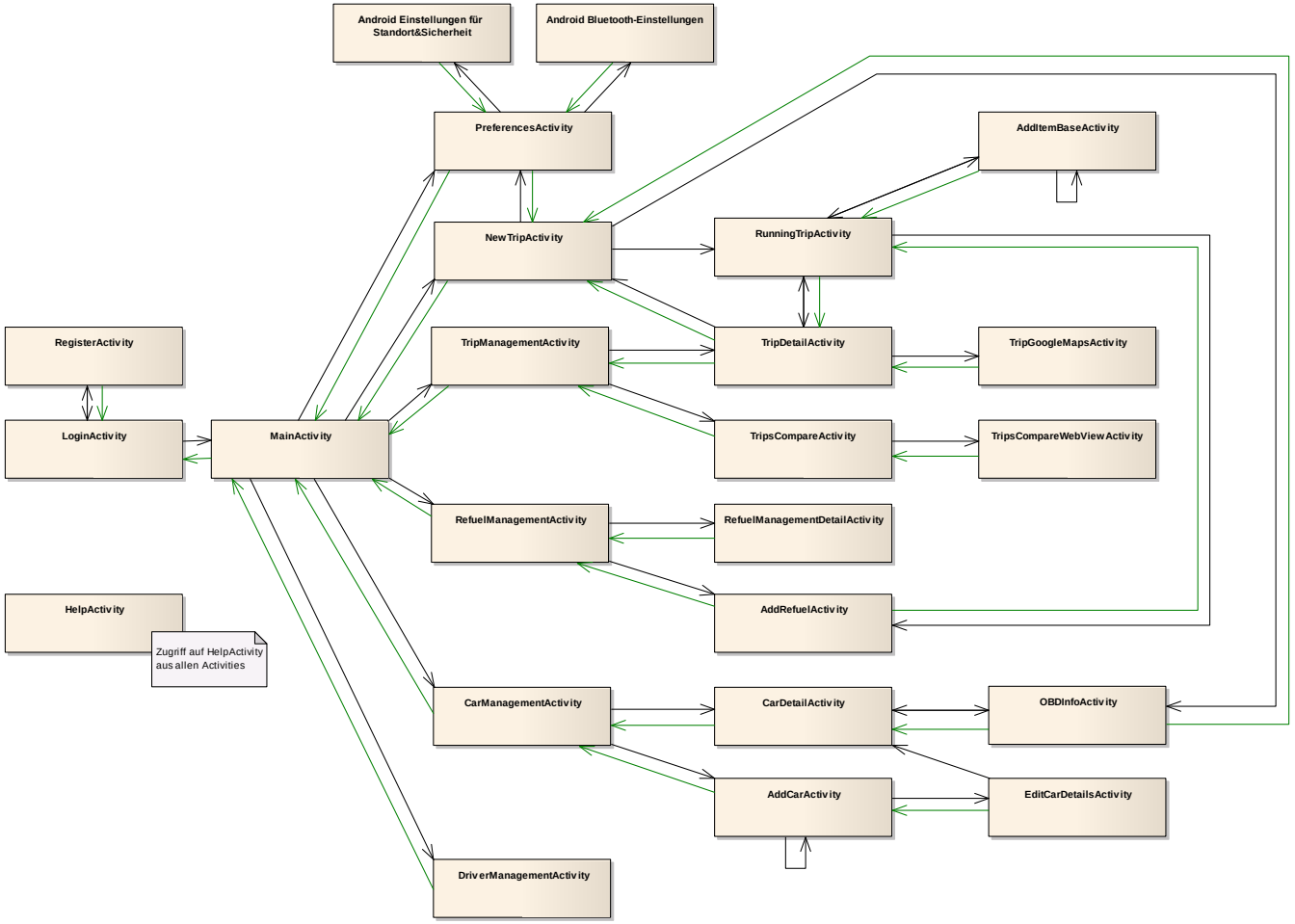


Abbildung 72: Activity Diagramm

12.10. Projektmanagement

12.10.1. Arbeitsaufteilung

Wir haben zu Beginn des Projekts eine grobe Arbeitsaufteilung durchgeführt. Sämtliche Projektteilnehmer müssen sich jedoch in der gesamten Anwendung auskennen.

Reto Zigerlig:	OBD-Information Einstellungen Oberfläche
Raphael Nagy:	Kommunikation mit Server Synchronisation Oberfläche
Stefan Schöb:	Datenaufzeichnung Datenanzeige während Trip

12.10.2. Entwicklungsprozess

Das EcoHelper-Projekt wurde nach einer angepassten Version des RUP entwickelt. Dabei wurden nicht benötigte Elemente weggelassen, die Grobstruktur jedoch beibehalten.

Die gesamte Entwicklung haben wir in die vier Phasen aufgeteilt:

Phase	Tätigkeit
Inception	Kennenlernen des EcoHelper-Projekts mit bestehender Infrastruktur, Prinzip des Systems und früheren Projekten.
Elaboration	Frühere Projekte genau analysiert. ▪ Welche Punkte sind für uns wichtig? ▪ Was muss besser gemacht werden? ▪ Was für neue Features wären interessant? Noch unbekannte Technologien kennenlernen und durch kleine Testprogramme ausprobieren (z.B. Bluetooth on Android)
Construction	Implementation des Programms
Transition	▪ Test und Verfeinerung der Anwendung. ▪ Bericht und Administrative Dokumente erstellt

Tabelle 21: Entwicklungsprozess

Die Phasen haben wir zur groben Einteilung des aktuellen Entwicklungsstands eingeführt. Wir haben uns nicht strikt daran gehalten. D.h. dass sich teils Phasen überschnitten haben. Weiter haben wir innerhalb der einzelnen Phasen keine Iterationen durchgeführt. Grund dafür ist, dass wir in einem so kleinen Team und der kurzen Zeit darin keinen Mehrwert eruieren konnten.

12.10.3. Zeitplan

Während des gesamten Projekts haben wir mit Redmine die Arbeitszeit erfasst. Dies war ein erster Versuch mit dieser Software.

Bei Projektbeginn wurden Tasks definiert und eine Schätzung vorgenommen. Im folgenden Diagramm sind die Haupttasks nach geschätzter und effektiv benötigter Zeit ausgewertet.
Die darunter definierten Subtasks wurden nicht genauer ausgewertet. Dies hat zwei Gründe:

1. Subtasks wurden zu Beginn zu ungenau definiert, wodurch beim Eintragen meist nicht klar war, zu welchem Subtask die Zeit gebucht werden soll.

2. Redmine war als Tool zur Zeiterfassung ungeeignet, wodurch meist mehrere Arbeiten zu einem Task zusammengefasst wurden (z.B. Arbeiten an GUI, Datenübertragung und Datenaufzeichnung wurden im Task Implementation rapportiert)

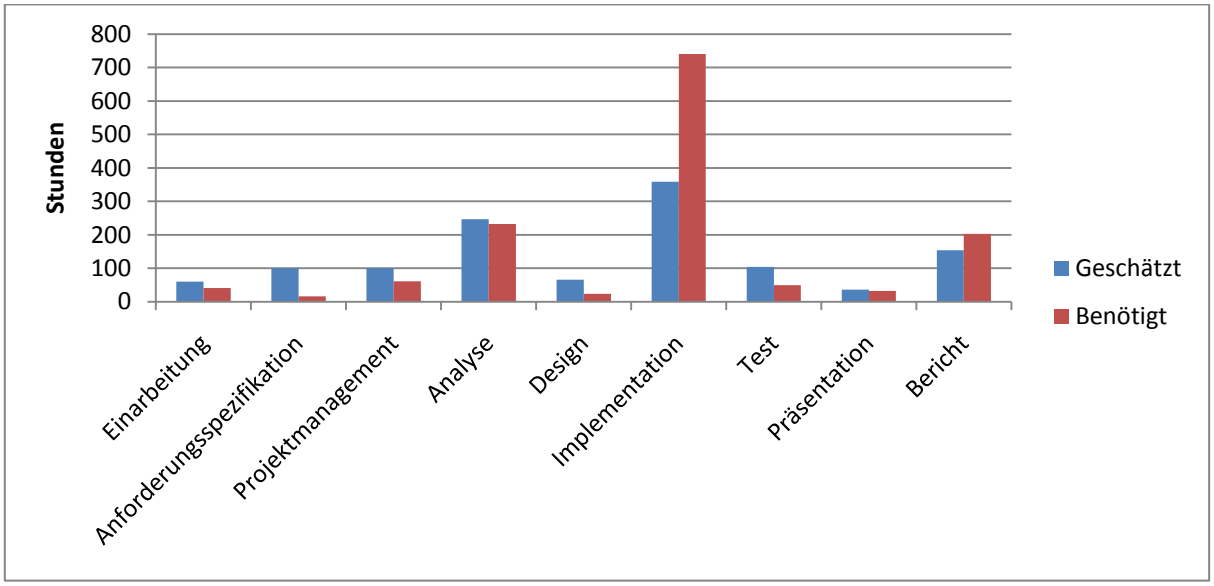


Abbildung 73: Zeitauswertung nach Haupttasks

Bis auf drei Ausnahmen wurde die Zeit gut einschätzen. Warum die drei Tasks nicht mit der Schätzung übereinstimmen, soll kurz erläutert werden:

Anforderungsspezifikation
Einerseits wurde zu viel Zeit eingerechnet. Andererseits ist die Anforderung während der Analyse und Implementation langsam herangewachsen. Entsprechend wurde die dafür verwendete Zeit diesem Task zugewiesen.

Implementation
Für die Implementation wurde zu wenig Zeit eingerechnet. Grund dafür war, dass die Anforderungen zu Beginn nicht genau definiert waren. Da zu den Aufgaben laufend neue Ideen zur Funktionalität und Oberfläche hinzukamen, wuchs entsprechend auch der benötigte Zeitaufwand.

Test
Geplant waren ausführliche Tests. Leider überstieg die Implementation das Zeitbudget in einem so grossen Masse, dass bei den Tests Abstriche gemacht werden musste.

12.10.4. Risikoanalyse

Risiko	Auswirkung	Massnahmen zur Vermeidung/Verminderung	max. Schaden[h]	Eintrittswahrscheinlichkeit	gewichteter Schaden [h]	Vorgehen bei Eintreffen
OBD-Parameter können nicht ermittelt werden	Anwendung wäre an sich unnütz, da nur noch mit GPS und Sensoren-Daten gearbeitet werden könnte	Exakte Analyse wie auf die OBD-Schnittstelle zugegriffen werden kann	200	5%	10	Saubere Analyse warum nicht möglich. Besprechung mit Projektbetreuer über weiteres Vorgehen der Arbeit
Projekt scheitert.	Keine ETCS Punkte und die BA muss wiederholt werden.	Besprechungen und Feedback des Betreuers	1080	1%	1.08	Offene Besprechungen
Kein Test Auto verfügbar	Es können keine Testfahren durchgeführt werden. Anwendung kann nicht mit Echtdaten geprüft werden	-	10	5%	0.5	Besprechung mit Betreuer, Simulator verwenden
Technische Einschränkung durch Android	Gewünschtes Feature kann nicht umgesetzt werden	Genaue Analyse der Android-Möglichkeiten	60	30%	18	Besprechen mit Betreuer über weiteres Vorgehen
			1350	**41 %**	**29.58**	

Tabelle 22: Risikoanalyse

12.10.5. Meilensteine

Meilenstein	Beschreibung	Datum
MS1	**Anforderungsspezifikation** ▪ Festlegen was die zu erstellende EcoHelper-Anwendung erfüllen muss. ▪ Welche Features sollen implementiert werden.	21.03.2011
MS2	**Softwarearchitektur / Schnittstellen definiert** ▪ Definieren einer groben Software-Struktur für die Hauptkomponenten. ○ Datenbankanbindung ○ Serverkommunikation	25.03.2011
MS3	**Bericht Part I** ▪ Erste Variante des Berichtes mit den Themen der Analyse und der Einführung	21.04.2011
MS4	**Prototyp I** ▪ Features der Priorität 1 wurden implementiert.	29.04.2011
MS5	**Prototyp II** ▪ Features der Priorität 1 wurde stabilisiert ▪ Features der Priorität 2 wurden implementiert.	20.05.2011
MS6	**Release Candidate I** ▪ Features der Priorität 1 und 2 wurden stabilisiert. ▪ Falls Zeit vorhanden Features der Priorität 3 implementiert	03.06.2011
MS7	**Finale Version** ▪ Die Anwendung läuft stabil und enthält alle gewünschten Features	10.06.2011
MS8	**Endabgabe** ▪ Version der Anwendung veröffentlicht ▪ Bericht fertiggestellt ▪ Poster fertiggestellt	17.06.2011

Tabelle 23: Meilensteine

12.10.6. Featurelist

Wir haben für die Entwicklung die zu entwickelnden Features nach deren Wichtigkeit in drei verschiedene Prioritätsstufen unterteilt.

Priorität 1

Features sind ein „must have“ und müssen implementiert werden.

- Login-Mechanismus
- Hauptscreen
- Setting-System (guter Zugriffsmechanismus auf die Einstellungen)
- DB-System auf dem Handy (Zugriffsschicht)
- Clientseitige Serverschnittstelle
- Trip erfassen
- Fahrerverwaltung
- Autoverwaltung
- Trip: dynamische Ansicht mit Screens
- Trip: Zahlenwert (Anzeige) (jetzt, Maximum, Durchschnitt)
- Trip: Online-Modus mit Upload
- Gangerkennung
- Datenerfassung: Bluetooth
- Datenerfassung: Sensoren
- Datenerfassung: GPS
- Schnittstellenfunktionalitäten

Priorität 2

Features werden nicht im ersten Release implementiert, sind aber für die finale Version ein „must have“.

- Registration
- Tankverwaltung
- Trip-Analyse (detaillierte Anzeige)
- Hilfe-Seiten
- Trip: Pause
- Trip: Diagramm (Anzeige) → wird nicht realisiert
- Trip: Offline-Modus
- OBD-Einbauhinweise
- Gefahrene Strecke auf Google-Maps anzeigen
- Trip: Element: Optimaler Schaltzeitpunkt → wird nicht realisiert
- GSM Parameter aufzeichnen

Priorität 3

Features die nur implementiert werden wenn Zeit übrig bleibt.

- Trip: weitere grafische Elemente (Anzeige)
- Trip: Aufnahme Video → nicht realisiert
- Trip: Aufnahme Bild → nicht realisiert

12.11. Systemtest

ID	Aktion	erwartetes Ergebnis	Resultat	Kommentar
T1	Start der Anwendung	Login Screen erscheint	ok	
T2	Click auf Registrieren	Registration Screen erscheint	ok	
T3	Click auf Registrieren ohne Angaben	Meldung, dass keine Mail Adresse angegeben wurde. Markierung des E-Mail Feldes mit roter Farbe.	ok	
T4	Eingabe einer ungültigen Mailadresse, anschliessend Klick auf Registrieren	Meldung, dass eine ungültige Mailadresse angegeben wurde	ok	
T5	Eingabe gültiger Mailadresse, Klick auf Registrieren	Meldung, dass kein Passwort eingegeben wurde. Markierung des Passwort Feldes mit roter Farbe. E-Mail Feld ist wieder in Standard Farbe	ok	
T6	Eingabe von der Ziffer 1 als Passwort und Passwort wiederholen. Klick auf Registrieren	Meldung, dass das Passwort zu kurz ist und dass mindestens 6 Zeichen benötigt werden	ok	
T7	Eingabe Passwort 123456 und bei Passwort wiederholen 654321. Klick auf Registrieren	Meldung, dass Passwörter nicht übereinstimmen & Markierung der Passwortfelder mit roter Farbe.	ok	
T8	Zwei Mal Eingabe von Passwort 123456 und Klick auf Registrieren. Anschliessend Klick auf OK.	Progress bar erscheint mit Meldung, dass Registrationsvorgang am Laufen ist. Anschliessend Meldung, dass Registration erfolgreich war und dass man die E-Mails für die Aktivierung prüfen soll. Nach Klick auf OK wird auf den Login Screen gewechselt.	ok	Hier darf Meldung nicht mit "Fehler" beginnen, ansonsten ist der User verwirrt, ob es nun geklappt hat oder ob ein Fehler aufgetreten ist!
T9	Eingabe von E-Mailadresse anschliessend Klick auf Login	Meldung, dass die Datenschutzbestimmungen akzeptiert werden müssen.	ok	
T10	Eingabe von E-Mailadresse, Klick auf Checkbox "Ich akzeptiere die Datenschutzbestimmungen" anschliessend Klick auf Login	Meldung, dass Passwort noch nicht eingegeben wurde	ok	
T11	Eingabe von E-Mailadresse und Passwort ohne	Meldung, dass die Datenschutzbestimmungen akzeptiert werden müssen.	ok	

	Datenschutzbestimmungen zu akzeptieren. Anschliessend Klick auf Login			
T12	Klick auf Datenschutzbestimmungen	Datenschutzbestimmungen werden geöffnet	nok!	1. Link ist schwierig zu treffen 2. Seite noch nicht vorhanden
T13	Eingabe von E-Mailadresse und Passwort inkl. Datenschutzbestimmungen zu akzeptieren. Anschliessend Klick auf Login	Progress bar mit Loginvorgang wird angezeigt, Login wird erfolgreich durchgeführt, Wechsel zu MainScreen	ok	
T14	Klick auf Neuer Trip	Meldung, dass zuerst ein Auto erfasst werden muss	ok	Meldung darf nicht lauten: "Bitte zuerst einen Fahrer und ein Auto erfassen" -> "Bitte erfassen Sie zuerst ein Auto" da der User ja schon als Fahrer angelegt wurde
T15	Klick auf Autoverwaltung	Autoverwaltung wird geöffnet, Synchronisation wird gestartet. Meldung dass zuerst ein Auto erstellt werden soll erscheint.	ok	Meldung ändern in: "Keine Daten vorhanden. Synchronisieren Sie die Daten oder erstellen Sie ein neues Auto über das Menü."
T16	Klick auf Menü, Auto hinzufügen	Wechsel in Auto hinzufügen Screen. Meldung erscheint, dass die Hersteller geladen werden. Sobald Hersteller verfügbar sind, werden diese in einer Liste angezeigt. Zuunterst erscheint ein Inputfeld um einen neuen Hersteller hinzuzufügen.	ok	
T17	Klick auf Smart	Wechsel zu Modellen von Smart. Meldung erscheint, dass die Modelle geladen werden. Sobald Modelle verfügbar sind, werden diese in einer Liste angezeigt. Zuunterst erscheint ein Inputfeld um einen neuen Modell hinzuzufügen.	ok	
T18	Klick auf fortwo	Wechsel zu Varianten von fortwo. Meldung erscheint, dass die Varianten geladen werden. Sobald Varianten verfügbar sind, werden diese in einer Liste angezeigt. Zuunterst erscheint ein Inputfeld um einen neue Variante hinzuzufügen.	ok	
T19	Klick auf coupé	Wechsel zu Detailangaben zum Auto.	ok	

T20	Klick auf Speichern	Meldung, dass zuerst der Aufbau gewählt werden muss	ok
T21	Auswahl von Aufbau, anschliessend Klick auf Speichern	Meldung, dass Getriebeart gewählt werden muss	ok
T22	Auswahl von Getriebeart Handschaltung, anschliessend Klick auf Speichern	Meldung, dass Anzahl Gänge angegeben werden muss	ok
T23	Auswahl von Getriebeart Automat, anschliessend Klick auf Speichern	Meldung, dass Treibstoff ausgewählt werden muss	ok
T24	Auswahl von Getriebeart Handschaltung, Anzahl Gänge 5, anschliessend Klick auf Speichern	Meldung, dass Treibstoff ausgewählt werden muss	ok
T25	Klick auf Optionale Angaben anzeigen	Hubraum, Gewicht, Jahrgang und Maximale Reichweite pro Tankfüllung werden angezeigt	ok
T26	Klick auf Optionale Angaben anzeigen	Optionale Angaben werden wieder ausgeblendet	ok
T27	Auswahl von Treibstoff Benzin, anschliessend Klick auf Speichern	Meldung, dass das Auto erstellt wird. Sobald Vorgang abgeschlossen ist, wird die Autodetails-Ansicht mit den erfassten Werten angezeigt. Falls ein Wert nicht angegeben wurde, wird dieser auch nicht in der Zusammenfassung angezeigt	ok
T28	Klick auf OBD2 Anschlussplatz	Wechsel zu OBD2 Informationen. Meldung, dass Daten geladen werden. Falls bereits OBD2 Informationen zu diesem Auto auf dem Server vorhanden sind, werden diese angezeigt. Dies beinhaltet die Angabe des Anschlussplatzes, Anzeige eines Bildes und die Anzeige einer Beschreibung. Zuunterst erscheint ein Button, mit dem man neue Anschlussplätze erfassen kann. Falls noch keine Daten vorhanden sind, kann eine Position gewählt werden, es wird ein "Kein Bild" Bild, ein Button "Foto aufnehmen" wie auch ein "Upload" Button und ein Textfeld für die Eingabe einer Beschreibung angezeigt.	ok

T29	Klick auf Foto aufnehmen	Android Kamera Ansicht wird gestartet	ok
T30	Foto aufnehmen und bestätigen	Wechsel in OBD2 Informationen. Foto wird nun angezeigt.	ok
T31	Beschreibung eingeben und auf Upload klicken	Meldung, dass Upload läuft. Wechsel in Autodetails	ok
T32	Klick auf OBD2 Anschlussplatz	Gerade erstellter Anschlussplatz wird mit Position, Bild und Beschreibung angezeigt.	ok
T33	Auf Mainscreen, Klick auf Neuer Trip	Meldung, dass zuerst ein Fahrer erstellt oder synchronisiert werden muss	ok
T34	Klick auf Fahrerverwaltung	Fahrerverwaltung wird geöffnet, Synchronisation wird gestartet. Standardfahrer (User) wird in Liste angezeigt	ok
T35	Klick auf Menü, Fahrer hinzufügen	Dialog um Name und Email anzugeben	ok
T36	Name und E-Mail eingeben, klick auf Erstellen	Fahrer wird erstellt und in Liste angezeigt	ok
T37	Longklick auf erstellten Fahrer, Klick auf Ändern, Name ändern, anschliessend Klick auf Ändern	Es erscheint Dialog um Name und Email zu ändern. Nach Änderung des Namens und Klick auf Ändern wird Fahrer mit neuem Namen angezeigt	ok
T38	Longklick auf geänderten Fahrer, Klick auf Löschen	Fahrer wird gelöscht und aus Liste entfernt	ok
T39	Auf Mainscreen, Klick auf Neuer Trip	Ansicht für neuen Trip wird geöffnet. Als Fahrer ist der Standardfahrer angegeben. Als Auto ist der Smart fortwo coupé angegeben. Unter Ortung werden die Status von GPS und WiFi Ortung angezeigt. Beide sind auf Deaktiviert. Unter OBD befindet sich der Status der Verbindung zur OBD Schnittstelle, dieser ist auf Bluetooth deaktiviert.	ok
T40	Klick auf Ortungs Einstellungen	Android Einstellungen für Standort & Sicherheit wird geöffnet	ok
T41	Klick auf Drahtlosnetzwerke und GPS-Satelliten, mit Backbutton zurück	Status von GPS wird auf Suche Signal gewechselt, falls noch kein GPS Signal gefunden wurde. WiFi ist nun Verbunden	ok

T42	Klick auf Informationen, anschliessend wieder zurück mit Backbutton	OBD2 Anschlussplatzinformationen werden angezeigt	ok	Informationen in OBD2 Anschlussplatz umbenennen?
T43	Klick auf Einstellungen, Bluetooth Einstellungen, Bluetooth an	EcoHelper Einstellungen werden geöffnet, Bluetooth Berechtigungsanfrage wird gestartet	ok	
T44	Klick auf Ja	Bluetooth wird aktiviert	ok	
T45	Klick auf Bluetooth Geräte, Nach Geräten scannen	In der Liste befinden sich keine gepaarten oder nicht gepaarten Geräte. Es wird nach Bluetooth Geräten in der Umgebung gesucht. ElmCan-BT wird unter „Nicht gepaarte Geräte“ angezeigt	ok	
T46	Klick auf ElmCan-BT, Eingabe des Pairing Schlüssels, ok	Pairingrequest wird aufgerufen, Aufforderung zur Eingabe des Pairing Schlüssels wird angezeigt. Nach Bestätigung wird ElmCan-BT als gewähltes Gerät angezeigt	ok	
T47	zwei Mal Klick auf Backbutton	Anzeige Neuer Trip erstellen wird angezeigt. Status von OBD ist nun Verbunden	ok	
T48	Klick auf Trip starten	RunningTripActivity wird angezeigt.	ok	
T49	Klick auf Anzeige hinzufügen, Einzelparameteranzeige, OBD2, Geschwindigkeit, Klein	OBD2 Geschwindigkeitsanzeige wird dem Screen hinzugefügt.	ok	
T50	Longclick auf Geschwindigkeitsanzeige, verschieben auf Papierkorb, Element loslassen	Anzeigeelement wird entfernt	ok	
T51	Klick auf Menü, Tanken, Tankangaben eingeben und auf Speichern klicken	Anzeige Tankvorgang erfassen wird geöffnet. Auto ist bereits ausgewählt, kann nicht geändert werden. Weitere Angaben können getätigt werden. Tankvorgang wird gespeichert. RunningTripActivity wird wieder angezeigt	ok	
T52	Klick auf Menü, Trip beenden	Trip-Detail Ansicht wird geöffnet. Tripdetails werden angezeigt, Button Upload zum Server steht für Übertragung der Daten zur Verfügung. Da kein GPS Signal vorhanden war, ist Button Trip in Google Maps anzeigen	ok	

		deaktiviert.	
T53	Klick auf Backbutton	Ansicht Neuer Trip erscheint	ok
T54	Auf Mainscreen, Klick auf Tankverwaltung.	Tankverwaltung wird geöffnet und synchronisiert	ok
T55	Klick auf Menü, Tankvorgang erfassen	Ansicht Tankvorgang erfassen wird geöffnet. Auto, Benzin, Tachostand und Getankte Menge, Gesamtkosten können angegeben werden. Ebenfalls können Masseinheiten gewählt werden. Checkbox für Vollgetankt ist standardmässig aktiviert, Optionale Angaben standardmässig deaktiviert.	ok
T56	Klick auf Optionale Angaben anzeigen	Eingabe für Durchschn. Verbrauch, Geschwindigkeit und Bemerkung wird angezeigt.	ok
T57	Klick auf Speichern	Tankvorgang wird gespeichert und in Liste angezeigt	ok
T58	Longclick auf Tankvorgang, Ändern, Speichern	Ansicht Tankvorgang ändern wird geöffnet. Es können die gleichen Angaben wie bei Tankvorgang erfassen geändert und angegeben werden. Nach Klick auf Speichern wird wieder in die Tankverwaltung gewechselt	ok
T59	Longclick auf Tankvorgang, Löschen	Tankvorgang wird aus Liste gelöscht, Synchronisation wird durchgeführt	ok

Tabelle 24: Systemtest

12.12. Sitzungsprotokolle

Die Sitzungsprotokolle befinden sich zur genaueren Betrachtung auf der CD.