

Wintersemester 2014/2015

# Projektarbeit

# **HFUwerdMobil!**

**Logistik und Supply Chain Management**

WIB 4

Prof. Dr. Jochen Baier

an der Fakultät Wirtschaftsinformatik

der Hochschule Furtwangen University

vorgelegt von

Matthias Feichtmeier, Mat.Nr. 244288

Veronika Kinzel, Mat.Nr. 240064

Susanne Roos, Mat.Nr. 244298

**Tobias Straub, Mat.Nr. 245634 (Teamleitung)**

eingereicht am: 22. Januar 2015



# Inhaltsverzeichnis

Abbildungsverzeichnis .....	V
Abkürzungsverzeichnis .....	VII
1. Einführung .....	9
1.1 Aufgabenstellung.....	9
1.2 Stakeholder .....	11
2. Bedienungsanleitung.....	12
2.1 Installation der App auf dem Smartphone .....	12
2.2 Funktionen .....	14
2.2.1 „Route wählen“ .....	14
2.2.2 „Route beenden“ .....	15
2.2.3 „Routendetails“ .....	16
2.2.4 „zuletzt gefahrene Routen“ .....	17
2.2.5 „Einstellungen“ .....	17
2.2.6 Informationsseiten .....	18
2.2.7 „Über“ .....	19
2.2.8 „Impressum“,.....	19
2.2.9 „Hilfe“ .....	20
3. Randbedingungen .....	21
3.1 Technische Randbedingungen .....	21
3.2 Organisatorische Randbedingungen .....	22
3.3 Konventionen.....	24
4. Versionskontrollsystem.....	24
4.1 Auswahlentscheidung.....	24
4.2 Bare-Repository Hosting .....	26
5. Kommunikation .....	28
5.1 Kommunikation intern.....	28

5.2	Kommunikation mit anderen Gruppen .....	29
5.3	Kommunikation mit Auftraggebern .....	30
6.	Anforderungsmanagement .....	30
6.1	User Stories.....	31
6.2	Product- Backlog .....	32
7.	Wireframing .....	34
8.	RESTful API.....	35
8.1	Spezifikation RESTful API (final) .....	36
9.	Dokumentation Softwaresystem .....	41
9.1	Speichermanagement.....	41
9.1.1	Shared Preferences .....	42
9.1.2	SQLite Datenbank .....	42
9.2	Double-Opt-In Verfahren .....	44
9.3	Verbesserungsvorschläge .....	44
10.	Fazit und Ausblick .....	44
11.	Quellenverzeichnis .....	47
12.	Anhänge (Versionshistorie - RESTful API Spezifikation, Quellcode, Javadoc, UML Diagramme).....	50
12.1	RESTful API - Versionsgeschichte .....	51
12.2	Javadoc, Quellcode und UML-Diagramme .....	75

## **Abbildungsverzeichnis**

Abbildung 1: Projekt HFUwerdMobil!

Abbildung 2: Interne und externe Stakeholder

Abbildung 3: Screen „Startbildschirm“

Abbildung 4: Screen „Anmeldung“

Abbildung 5: Screen „Route wählen“

Abbildung 6: Screen „Aufzeichnung und Route beenden“

Abbildung 7: Screen „Routendetails“

Abbildung 8: Screen „Einstellungen“

Abbildung 9: Screen „Über“

Abbildung 10: Screen „Impressum“

Abbildung 11: Verwendung Android Plattform Versionen

Abbildung 12: „Kopieren-Ändern-Zusammenfassen“ Teil 1

Abbildung 13: „Kopieren-Ändern-Zusammenfassen“ Teil 2

Abbildung 14: HFUwerdMobil! Repository auf GitHub

Abbildung 15: Projektaufbau der Gruppen

Abbildung 16: User Stories Seite 1, Status: Beginn der Entwicklung

Abbildung 17: User Stories Seite 2, Status: Beginn der Entwicklung

Abbildung 18: User Stories Seite 1, Status: Beginn der Entwicklung

Abbildung 19: Product Backlog, Seite1, Status: Beginn der Entwicklung

Abbildung 20: Product Backlog, Seite2, Status: Beginn der Entwicklung

Abbildung 21: Product Backlog, Seite3, Status: Beginn der Entwicklung

Abbildung 22: Übersicht Wireframes



## **Abkürzungsverzeichnis**

ADT	Android Developer Tools
API	Programmierschnittstelle (engl. Application programming interface)
APK	Android Package
App	Applikation
CORBA	Common Object Request Broker Architecture
CRUD	Create Read Update Delete
FAQ	Frequently Asked Questions
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IDE	Integrierte Entwicklungsumgebung
OBD	On-Board-Diagnose
REST	Representational State Transfer
RPC	Remote Procedure Call
SD-Karte	Secure Digital Memory Card
SOAP	Simple Object Access Protocol
UML	Unified Modeling Language
XP	Extreme Programming



# 1. Einführung

## 1.1 Aufgabenstellung

Die App HFUwerdMobil! ist Teil eines fächerübergreifenden Semesterprojekts des Studiengangs WIB4. Die Applikation wird von Studenten des Profilfachbereichs Entwicklung betrieblicher Anwendungssysteme für das Projekt des Pflichtfachs Logistik und Supply Chain Management entwickelt und stellt gleichzeitig deren praktische Prüfungsleistung für das Fach Mobile Applikationen dar. Die App ist das Kernstück eines großen Projekts des Studiengangs zum Thema Mobilität.

Das Fehlen eines Bahnhofs in Furtwangen und die damit verbundene schlechte Anbindung an öffentliche Verkehrsmittel des Nah- und Fernverkehrs machen Mobilität für Studierende der Hochschule Furtwangen University zu einem zentralen Thema. Nicht nur der Ruf nach mehr Parkmöglichkeiten wurde immer wieder laut, auch Car-Sharing-Angebote und Mitfahrgelegenheiten erfreuen sich großer Beliebtheit. In Anbetracht dieser Situation soll mit HFUwerdMobil! die Mobilität der Studierenden, nicht nur an der Hochschule Furtwangen, sondern auch an den beiden Standorten Schwenningen und Tuttlingen gefördert werden.

Mit HFUwerdMobil! haben Studierende, wie auch Mitarbeiter und Professoren der HFU die Möglichkeit ihre Fahrtwege zu den Standorten der HFU bzw. zu ihrem Heimat- oder Zielort aufzuzeichnen. Neben Funktionalitäten wie der Anbindmöglichkeit an einen OBD2-Adapter und dem damit verbundenen Auslesen der von ihm übertragenden Fahrzeugdaten, kann mit der App die gefahrene Strecke visualisiert werden und eine Anzeige und Auswahl der zuletzt gefahrenen Routen getroffen werden.

Im Kontext des gesamten Projekts bedeutet dies, dass die App als Kernstück fungiert. Durch Ihre Anbindung an eine auf dem Server liegende Datenbank können sowohl Fahrerdaten, als auch GPS-Daten gespeichert werden.

Dem potentiellen Mitfahrer, welcher ebenfalls über einen HFU Account verfügen muss, wird nun die Möglichkeit geboten die Mitfahrgelegenheiten live auf der Webseite von HFUwerdMobil! unter <http://www.hfuwerdmobil.de/> einzusehen. Er kann nun via Mobiltelefon Kontakt mit dem Fahrer aufnehmen und idealerweise mitfahren.

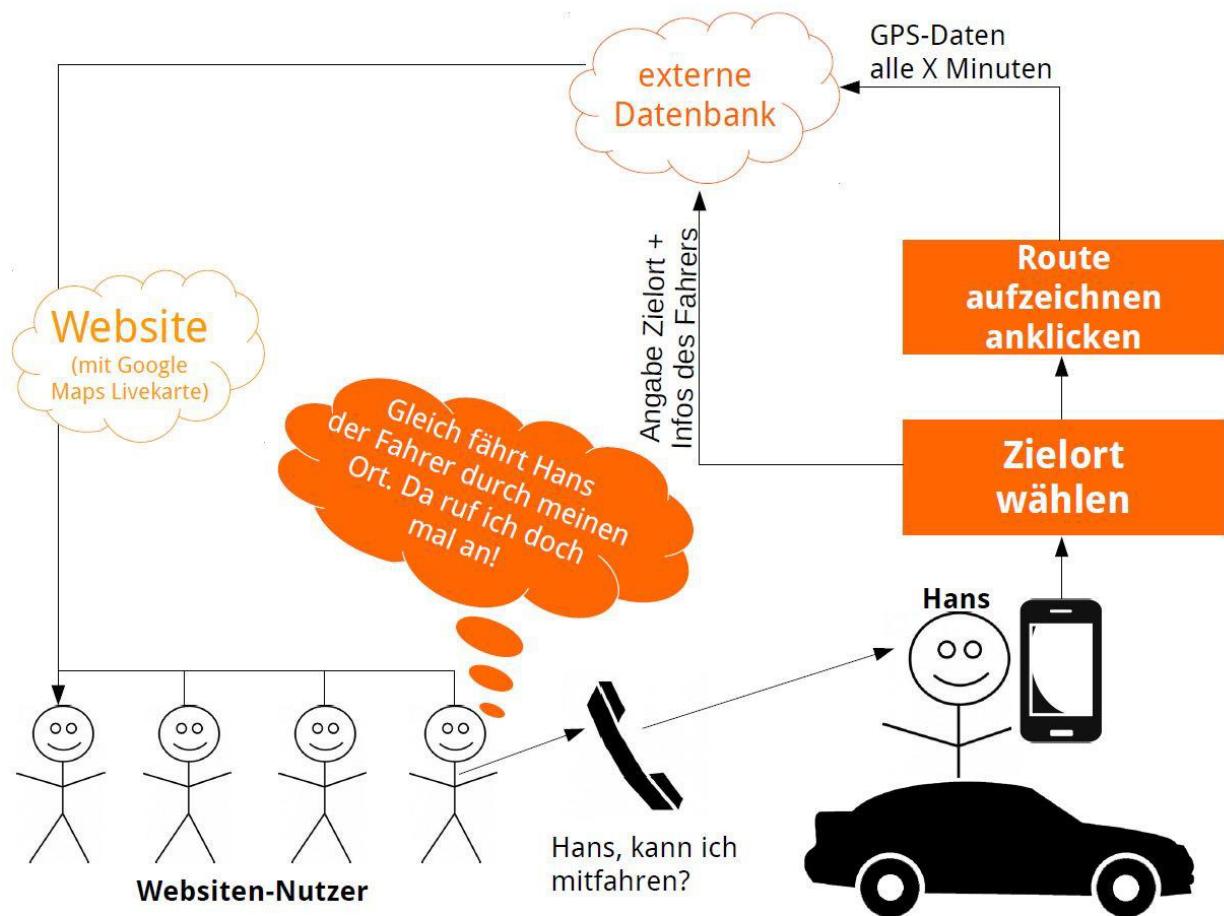


Abbildung 1: Projekt HFUwerdMobil!

## 1.2 Stakeholder

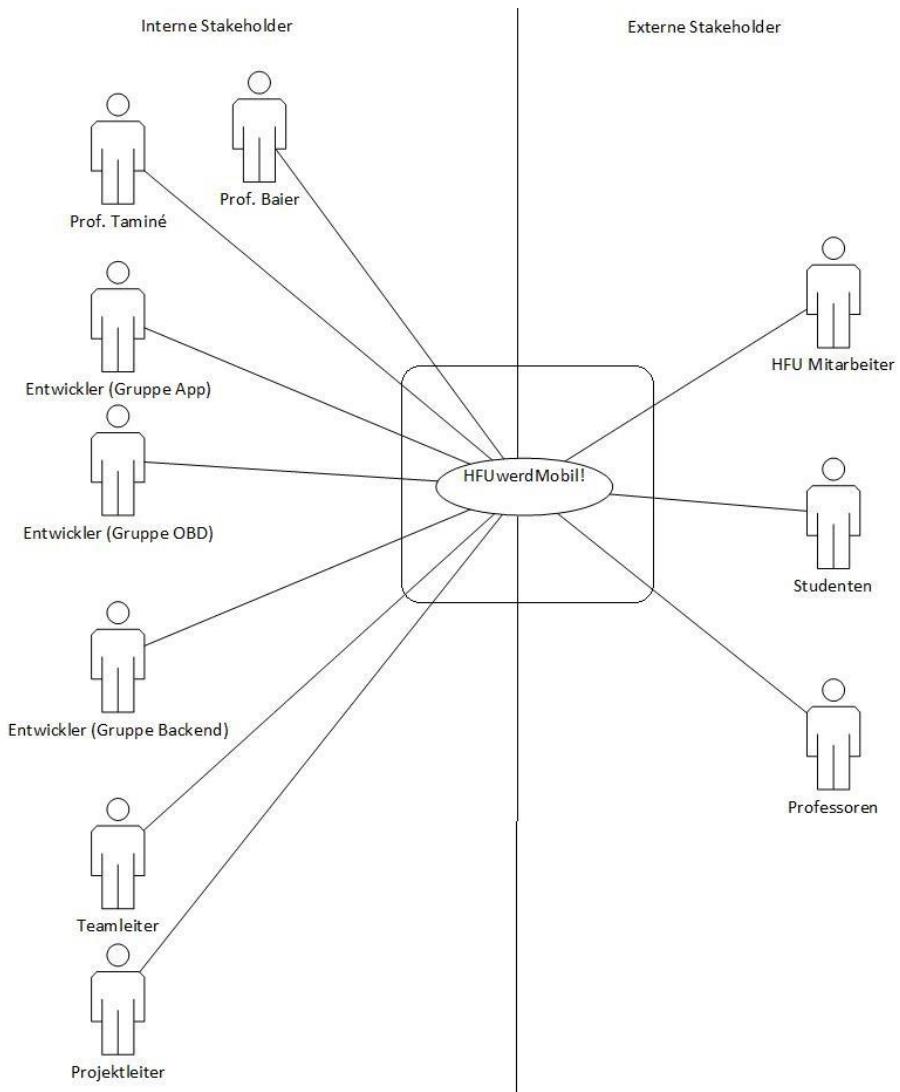


Abbildung 2: Interne und externe Stakeholder

## **2. Bedienungsanleitung**

### **2.1 Installation der App auf dem Smartphone**

Bei der App HFUwerdMobil! handelt es sich um eine für Android Geräte konzipierte Anwendung. Die Installation erfolgt momentan (Stand: 22.01.2015) indem die App in Form einer APK-Datei auf das Smartphone geladen wird. Über verschiedene Wege, beispielsweise via USB, E-Mail, Download etc. kann die App auf das Smartphone geladen werden.

Grundvoraussetzung für den APK Download ist, dass das entsprechende Android-Gerät das Öffnen unbekannter Quellen erlaubt. Diese Einstellungen können auf dem Android-Gerät unter "Einstellungen – Anwendungen", unter der Rubrik "Unbekannte Quellen aktivieren" gesetzt werden. Ab Android 4.0 befindet sich diese Option unter "Einstellungen - Sicherheit".<sup>1</sup>

Wird die App heruntergeladen ist eine direkte Installation im Anschluss möglich.

Bei einem Kopieren der APK-Datei auf das Android-Gerät sollte darauf geachtet werden die App vorzugsweise auf die SD-Karte des Smartphones zu kopieren. Bei manchen Geräten kann es vorkommen, dass nicht manuell in das Verzeichnis der SD-Karte beziehungsweise in den Ordner Downloads navigiert werden kann. In diesem Fall ist es notwendig eine App als Dateimanager zu installieren. Gute Erfahrungen wurden mit der App Datei Manager (File Manager) gemacht. Sie ist kostenlos im Google Play Store verfügbar.<sup>2</sup>

Nach erfolgreicher Navigation in den entsprechenden Dateiorndner auf dem Smartphone kann die App mithilfe des APK Files installiert werden.

Die App HFUwerdMobil! ist für Studierende der Hochschule Furtwangen University konzipiert. Bei der ersten Anmeldung fragt die App ob ein HFU Account vorhanden ist oder nicht. Falls das der Fall ist wird der Nutzer aufgefordert seine HFU E-Mailadresse einzugeben.

Falls kein HFU Account vorhanden ist prüft die App ob ein Google Account vorhanden ist und fügt diesen dann als User-E-Mailadresse hinzu. Ist kein Google Account vorhanden wird der Nutzer zur Eingabe einer validen E-Mailadresse aufgefordert.

Zudem wird bei Erstbenutzung der App die Mobilfunknummer des phone device ausgelesen. Sollte dieses Auslesen nicht erfolgreich sein, wird der Nutzer zur Eingabe einer gültigen Mobilfunknummer aufgefordert.

---

<sup>1</sup> [http://www.chip.de/artikel/APK-installieren-Android-App-aufs-Handy-bringen\\_47069033.html](http://www.chip.de/artikel/APK-installieren-Android-App-aufs-Handy-bringen_47069033.html)

<sup>2</sup> <https://play.google.com/store/apps/details?id=com.rhmsoft.fm&hl=de>

Bei jedem weiteren Start der App wird im Hintergrund automatisch geprüft, ob sich die in den Benutzereinstellungen hinterlegte E-Mailadresse geändert hat. Sollte dies der Fall sein, werden die geänderten Daten entsprechend in den Einstellungen der App hinterlegt.

Bei jedem weiteren Start der Anwendung wird der Nutzer aufgefordert sich entweder als HFU-Angehöriger mit seiner HFU-Emailadresse anzumelden oder die Option „Mit externer Email-Adresse anmelden“ zu wählen. Da die App HFUwerdMobil! schwerpunktmäßig für Studierende, Professoren und Mitarbeiter der Hochschule Furtwangen konzipiert wurde, steht die volle Funktionalität der Anwendung nur für HFU Angehörige zur Verfügung. Welche Funktionalitäten die App bietet wird unter „2.2 Funktionen“ im Detail beschrieben.



Abbildung 3: Screen „Startbildschirm“



Abbildung 4: Screen „Anmeldung“

## 2.2 Funktionen

### 2.2.1 „Route wählen“

Nach erfolgreicher Anmeldung auf der Startseite gelangt der Nutzer zuerst auf die Seite „Einstellungen“. Hier wird er aufgefordert seine Nutzerdaten einzugeben.

Danach gelangt er zu der Seite „Route wählen“. Das erste Feld in der Anzeige ist der aktuelle mittels GPS ermittelte Standort des Nutzers. Er dient als Ausgangslage für seine Fahrt. Im zweiten Feld kann nun eine Auswahl zum Ziel getroffen werden. Dies geschieht mittels einer Dropdownliste, in welcher zwischen folgenden Optionen gewählt werden kann: HFU Furtwangen, HFU Villingen-Schwenningen, HFU Tuttlingen und „nach Hause“.

Mit „Route starten“ beginnt die App nun mit der Aufzeichnung der gefahrenen Route. Im Hintergrund werden kontinuierlich die aktuellen GPS-Koordinaten ermittelt und zur späteren Darstellung der Route in der internen und der serverseitigen Datenbank abgespeichert. Zudem gibt es eine Liste der zuletzt gefahrenen Routen (vgl. 2.2.4).

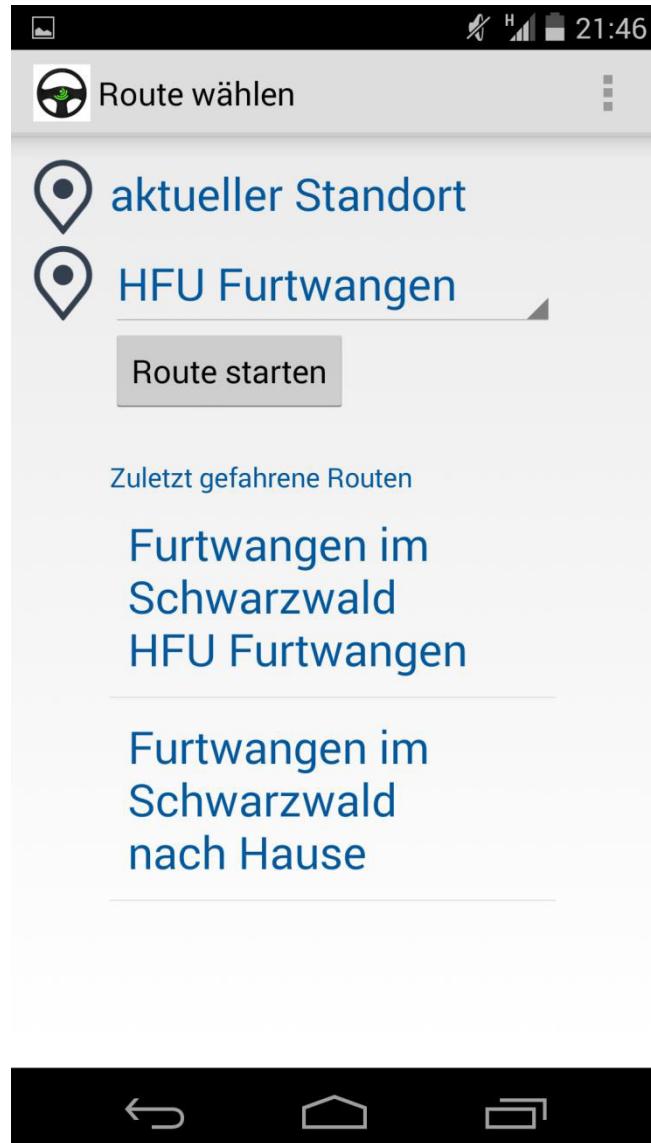


Abbildung 5: Screen "Route wählen"

### 2.2.2 „Route beenden“

Die Aufzeichnung der Fahrtstrecke terminieren: Die Route wird beendet wenn der Nutzer in den nächsten Umkreis der entsprechend gewählten Hochschule gelangt ist. In diesem Fall erkennt die App automatisch, dass das Ziel erreicht wurde.

Sofern der Heimatort des Fahrers das Ziel ist, muss der Nutzer die App manuell durch Wählen der Option „Route beenden“ stoppen.

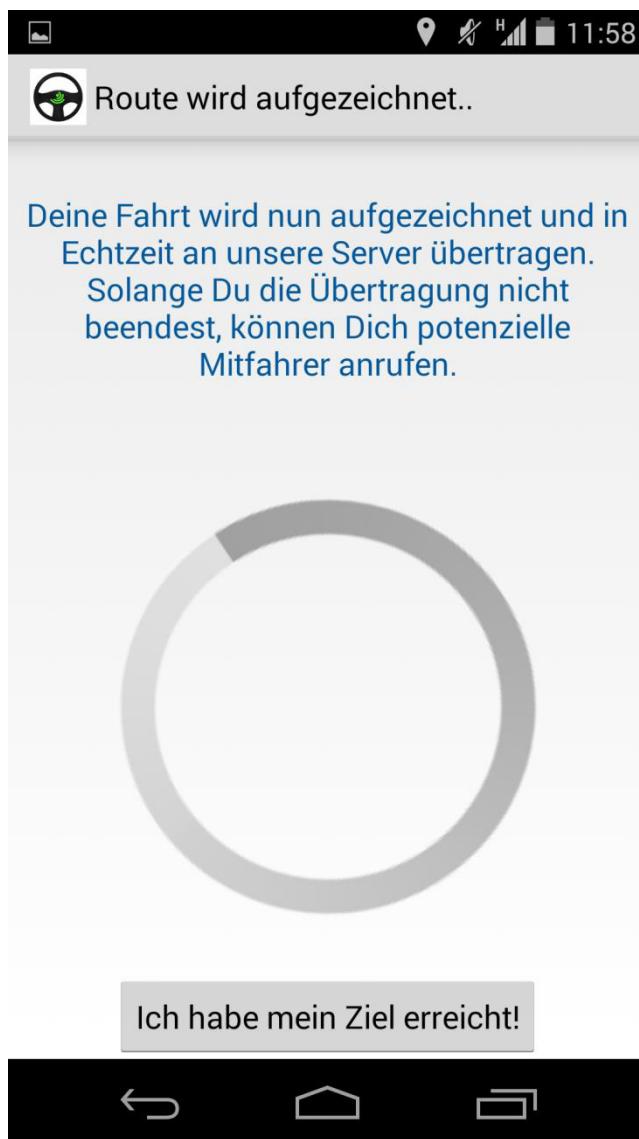


Abbildung 6: Screen „Aufzeichnung und Route beenden“

### 2.2.3 „Routendetails“

Nachdem die Aufzeichnung der Route durch Click auf den Button zum Beenden der Aufzeichnung (s. 2.2.2) beendet wurde, werden die Routendetails aufgerufen und dem Nutzer nun abschließend ein Google Maps Kartenausschnitt mit der zurückgelegten Route angezeigt.

Start- und Endpunkt der Route sind mit Markern markiert, zudem ist die gefahrene Strecke farblich gekennzeichnet.

Dies wird erreicht mittels einer polyline, welche die gesendeten GPS-Punkte verbindet. Die ermittelten Daten des OBD2-Adapters werden -sofern dieser während der Fahrt aktiviert war- unterhalb des Kartenausschnitts angezeigt.

Dem App-Nutzer werden die Fahrtzeit, Fahrstrecke, Durchschnittsgeschwindigkeit und der Verbrauch angezeigt.



Abbildung 7: Screen „Routendetails“

#### 2.2.4 „zuletzt gefahrene Routen“

Unter „Route wählen“ (vgl. 2.2.1) erfolgt zudem eine Auflistung der zuletzt gefahrenen Routen. Diese werden diese dem Nutzer in Textform mit Start und Ziel aufgelistet. Durch Klick auf eine dort aufgeführte Route werden die zugehörigen Routendetails aufgerufen und angezeigt (vgl. 2.2.3).

#### 2.2.5 „Einstellungen“

Über das Menü der App gelangt der Nutzer zu den individuellen Benutzereinstellungen der App. Beim ersten Start der Anwendung werden Vorname, Nachname und Postleitzahl des Heimatorts

abgefragt und hinterlegt. Möchte der Nutzer diese nun ändern, kann er dies über den Menüpunkt „Einstellungen“ ändern. Des Weiteren kann manuell eine andere Kontaktnummer als die aus dem Smartphone ausgelesene Telefonnummer hinterlegt werden. Diese dient zur Kontaktaufnahme im Rahmen der Community-Funktionalitäten der App.

Zudem besteht die Möglichkeit die Verbindung zum OBD Adapter, also die Bluetooth-Schnittstelle zum On-Board-Device, auszuschalten. Dies ist empfehlenswert wenn die App eigenständig, ohne den OBD-Adapter funktionieren soll.

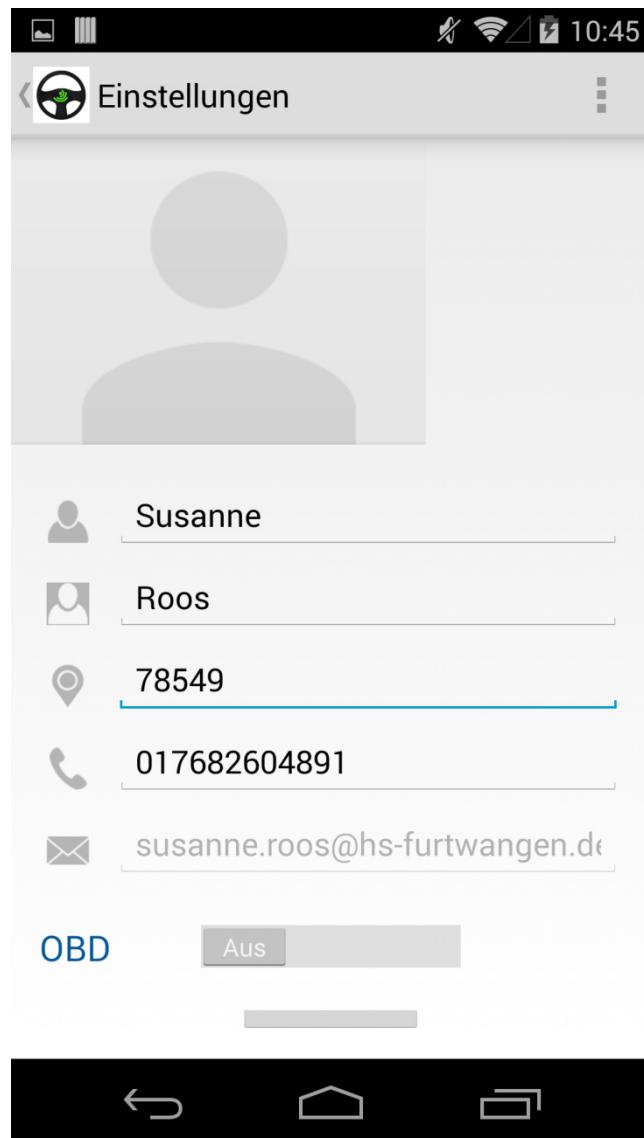


Abbildung 8: Screen „Einstellungen“

## 2.2.6 Informationsseiten

Unter den drei folgenden Rubriken soll dem Nutzer von HFUwerdMobil! ein Einblick in allgemeine Informationen zur App sowie Kontaktdaten und Hilfeseiten gewährt werden.

## 2.2.7 „Über“

Hintergrundinformationen zur App HFUwerdMobil! wie etwa das Logo der Applikation und Fotos des Entwicklerteams sind unter der Rubrik „Über“ aufgeführt.

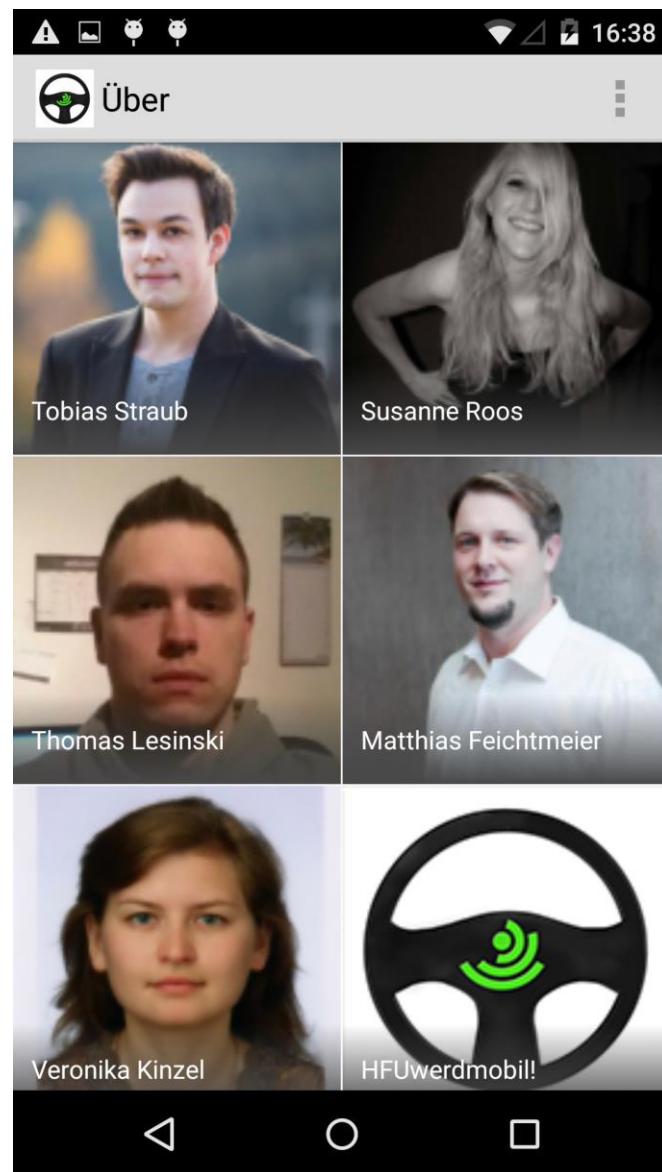


Abbildung 9: Screen "Über"

## 2.2.8 „Impressum“,

Im Impressum finden sich Name und Anschrift des Dienstanbieters sowie die Angaben zum Haftungsausschluss und Quellenangaben zu verwendeten Grafiken.

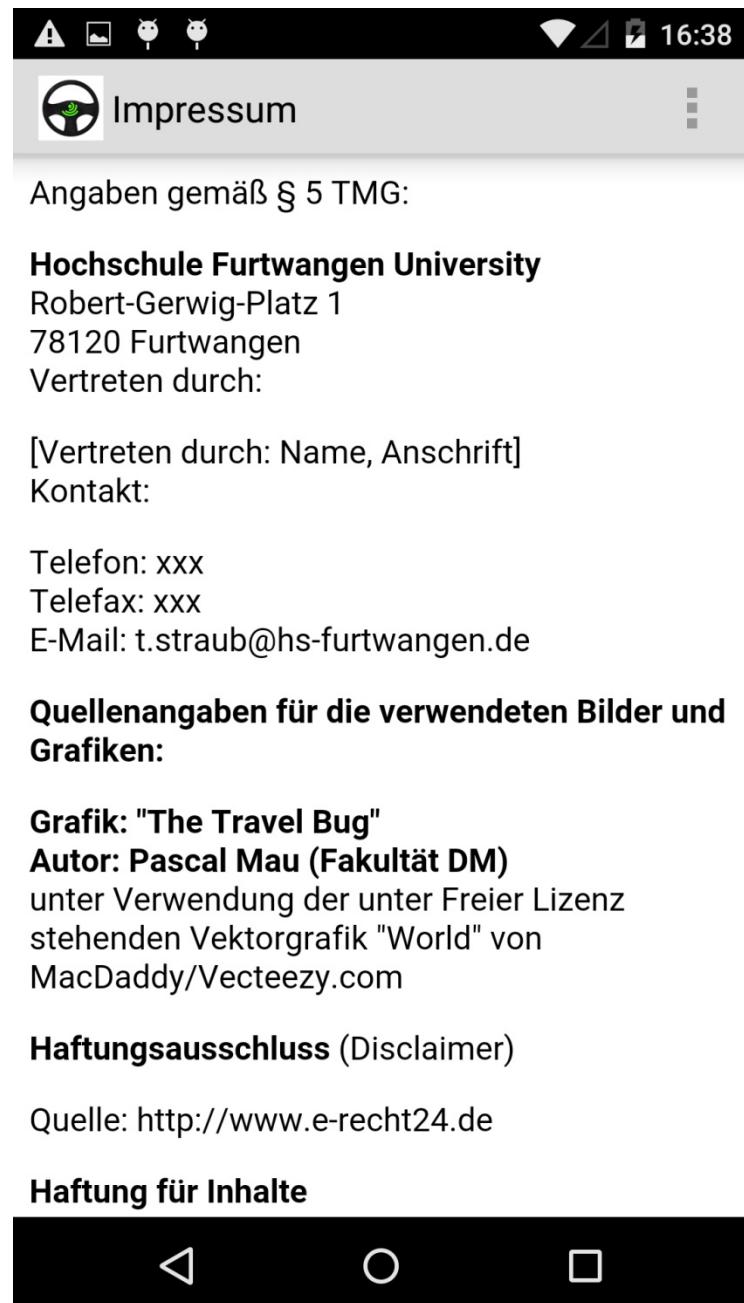


Abbildung 10: Screen "Impressum"

## 2.2.9 „Hilfe“

Unter „Hilfe“ werden die FAQs zusammengefasst, um dem Nutzer einen Überblick über die Funktionalitäten der App zu verschaffen und die Bedienung der Anwendung zu erleichtern.

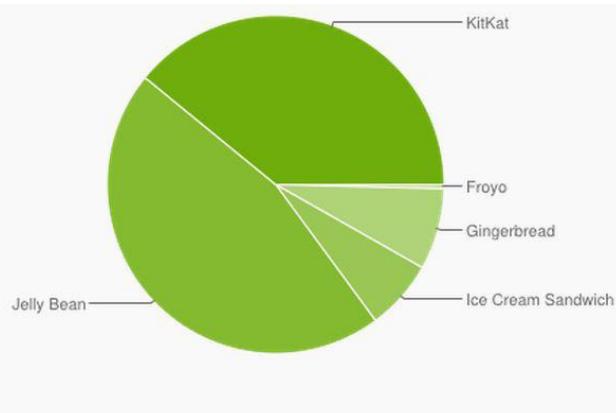
## 3. Randbedingungen

### 3.1 Technische Randbedingungen

Die App muss auf einem Android-Gerät betrieben werden. Unterstützte Versionen sind Android 4.0 (Ice Cream Sandwich) bis Android 5.0 (Lollipop). Da dieses Semesterprojekt künftig weiter ausgebaut werden soll, sei es durch Hinzufügen weiterer Funktionalitäten, Schnittstellen oder zur Wartung, steht für das Team im Vordergrund die App in der neuesten Android Version 5.0 (Lollipop) zu entwickeln. Dass bei Lollipop eine Abwärtskompatibilität besteht, stellt sich als großer Vorteil heraus. So können auch ältere Android Geräte ab Version 4.0 die App im vollen Funktionsumfang nutzen. Einzige Einschränkungen liegen im Layout, die Darstellung variiert innerhalb der verschiedenen Android Versionen ein wenig, stellt aber keinerlei Hindernis in puncto Benutzbarkeit dar.

Gemäß einer aktuellen Statistik von Android zu der Nutzung von Android Plattformen ist Jelly Bean derzeit die am meist genutzte Plattform.<sup>3</sup> Die HFUwerdMobil App ist kompatibel mit diesen Android Versionen und stellt somit für den Großteil der HFU Studenten eine in vollem Umfang nutzbare Anwendung dar.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.7%
4.1.x	Jelly Bean	16	19.2%
4.2.x		17	20.3%
4.3		18	6.5%
4.4	KitKat	19	39.1%



*Data collected during a 7-day period ending on January 5, 2015.*

*Any versions with less than 0.1% distribution are not shown.*

Abbildung 11: Verwendung Android Plattform Versionen

Quelle: <https://developer.android.com/about/dashboards/index.html>

<sup>3</sup> <https://developer.android.com/about/dashboards/index.html>

Als integrierte Entwicklungsumgebung wurde Eclipse mit dem Android Developer Tools (ADT)- Plug-In verwendet, die verwendete Programmiersprache ist Java. Darüber hinaus kam das Google Play Framework zum Einsatz.

Die Android Developer Tools bieten eine Vielzahl von Funktionen in der Eclipse Entwicklungsumgebung:<sup>4</sup>

- schnelles Erstellen von neuen Android Projekten
- Entwicklung einer Benutzeroberfläche für eine Anwendung unterstützt durch ein grafisches Tool
- Hinzufügen von packages aus den Android Framework APIs (Schnittstellen)
- Debugging der Android Anwendung mithilfe der Android SDK (Software Development Kit) Tools
- Exportfunktion der .apk Dateien der Android Anwendung zur Weitergabe der App

## 3.2 Organisatorische Randbedingungen

Im Folgenden sollen Teamstruktur, Zeitplan, Vorgehensmodell und Entwicklungswerkzeuge, sowie verwendete Softwares und Tools dieses Semesterprojekts vorgestellt werden.

- Team:

Das Team besteht aus dem Teamleiter, Tobias Straub, sowie zwei weiteren vollwertigen Mitgliedern: Matthias Feichtmeier und Susanne Roos. Ein weiteres Teammitglied nimmt seitens der Vorlesung Logistik teil: Veronika Kinzel. Als sechstes Teammitglied wirkt Thomas Lesinski für die Vorlesung Entwicklung betrieblicher Anwendungssysteme an dem Semesterprojekt mit.

- Zeitplan:

- Beginn der Entwicklung: Oktober 2014
  - erster lauffähiger Prototyp am 4. Dezember 2014
  - Fertigstellung am 22. Januar 2015 (Abgabe Semesterprojekt)

- Vorgehensmodell

Bei dem Projekt HFUwerdMobil! wurde, soweit möglich, das agile Software Development Framework SCRUM verwendet. Hierbei handelt es sich um ein nicht lineares, itera-

---

<sup>4</sup> <http://developer.android.com/tools/sdk/eclipse-adt.html>

tives und inkrementelles Rahmenwerk. Mit dessen Hilfe können Teams komplexe Aufgabenstellungen, mit dem Ziel produktiv und kreativ Produkte mit dem höchstmöglichen Wert auszuliefern, angehen.<sup>5</sup>

Vor allem die Vorteile der agilen Vorgehensweise machen das Modell für dieses Projekt sehr interessant. Das Team arbeitet eigenverantwortlich, der Fokus liegt in erster Linie auf lauffähigem Code, weniger auf ausführlicher Dokumentation. Zudem liegt der Fokus auf einer hohen Flexibilität gegenüber neuen Anforderungen. Die Teamgröße von sechs Mitgliedern eignet sich hervorragend für diese agile Methode.

- Entwicklungswerzeuge

Erste Entwürfe der App erfolgen mit Stift und Papier.

Die Product Backlog Items und User Stories werden mit Excel festgehalten. Anschließende UML-Diagramme entstehen mit MS Visio bzw. Enterprise Architect. Aufgaben und Arbeitsergebnisse werden in GitHub unter der Rubrik „Issues“ gesammelt und ggf. diskutiert. Die Java-Quelltexte sowie Javadoc (als Dokumentation) werden in Eclipse erstellt.

- Prototyping

Der Zweck an einen Prototyp ist es Anforderungen und mögliche Lösungsideen zu klären. Ein Prototyp soll nicht nur lauffähig sein, sondern auch ausgewählte Aspekte des Zielsystems realisieren und vor allem vom Auftraggeber geprüft werden. Eine Präsentation des Prototyps erfolgte Ende November. Dieser erste Prototyp stellte sich als lauffähige Lösung heraus und bildet eine gute Basis für die Weiterentwicklung. Dementsprechend wurde dieser Prozess hin zum Zielsystem, also einer funktionstüchtigen Anwendung mit allen geforderten Funktionalitäten, weiterentwickelt.

- Konfigurations- und Versionsverwaltung

Github (siehe 4. Versionskontrollsystem)

---

<sup>5</sup> Schwaber / Sutherland, SCRUM Guide 2011

### **3.3 Konventionen**

Innerhalb des Java Quellcodes der HFUwerdMobil! App werden Kodierrichtlinien gemäß der Java Coding Conventions von Oracle eingehalten.<sup>6</sup>

Innerhalb der Dokumentation (UML-Diagramme, Texte) werden Dinge (Komponenten, Schnittstellen etc.) Deutsch benannt. Im Java-Quelltext hingegen werden englische Bezeichner für Klassen, Methoden, Attribute etc. verwendet.

## **4. Versionskontrollsystem**

Unter einem Versionskontrollsystem oder auch Revisionsmodellsystem versteht man ein System, das inkrementelle Versionen von Dateien oder auch Verzeichnissen über einen gewissen Zeitraum verfolgt. Zum einen erlaubt es die Nachverfolgung verschiedener Versionen einer Datei eines Anwenders, zum anderen die Untersuchung von Änderungen, die zu jeder dieser Versionen getätigt wurden.<sup>7</sup> Des Weiteren können Modifizierungen einer Datei wieder revidiert werden und ebendiese einem Zeitpunkt sowie einer Person zugeordnet werden. Diese Art von System unterscheidet sich nicht nur grundlegend in der Anwendung, sondern auch in der Architektur von normalen Datensicherungssystemen. Versionskontrollsysteme empfehlen sich vor allem für den Einsatz in verteilten Projekten – dies kann sich sowohl auf die örtliche Verteilung wie auch ein Projekt mit Verteilung auf verschiedenen Personen beziehen.<sup>8</sup>

Diesbezüglich ein besonders wichtiger Vorteil eines solchen Systems stellt die Realisierung von kollaboriertem Bearbeiten und Teilen von Dateien dar. Verschiedene Versionskontrollsysteme verwenden unterschiedliche Strategien um dies zu bewerkstelligen.<sup>9</sup>

### **4.1 Auswahlentscheidung**

Im Rahmen dieses Semesterprojektes treten zwei Herausforderungen auf: Einerseits die Teamarbeit als relativ große Gruppe mit sechs Teammitgliedern und andererseits die Zusammenarbeit an verteilten Orten. Um diese Herausforderungen bestmöglich zu bewältigen wird Git als Versionskontrollsystem eingesetzt.

---

<sup>6</sup> <http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html>

<sup>7</sup> <http://svnbook.red-bean.com/de/1.6/svn.basic.version-control-basics.html>

<sup>8</sup> <http://www.itwissen.info/definition/lexikon/Versionskontrolle-version-control-system-VCS.html>

<sup>9</sup> <http://svnbook.red-bean.com/de/1.6/svn.basic.version-control-basics.html>

Arbeiten mehrere Nutzer gleichzeitig an derselben Datei kann es zu Problemen kommen. Git benutzt ein „Kopien-Ändern-Zusammenfassen-Modell“. Diese Technologie ermöglicht eine Verbindung jedes Clients aus der Summe der Anwender mit dem Archiv des Projekts (Repository). Nach Verbindungsauflauf wird eine persönliche Arbeitskopie für jeden Client erzeugt. Gleichzeitig und unabhängig voneinander arbeiten die Anwender nun an ihrem privaten Kopien. Hierbei spielt es keine Rolle, ob der Client on- oder offline ist. Schließlich findet ein Zusammenfügen aller Einzelkopien zu einer aktuellen, neuen Version statt.<sup>10</sup>

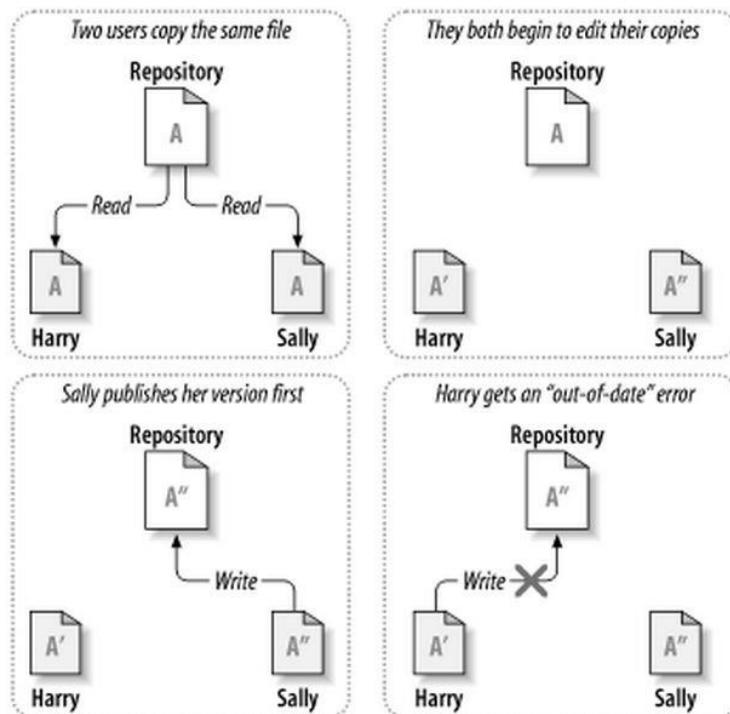


Abbildung 12: "Kopieren-Ändern-Zusammenfassen" Teil 1

Quelle: <http://svnbook.red-bean.com/de/1.6/svn.basic.version-control-basics.html>

---

<sup>10</sup> <http://svnbook.red-bean.com/de/1.6/svn.basic.version-control-basics.html>

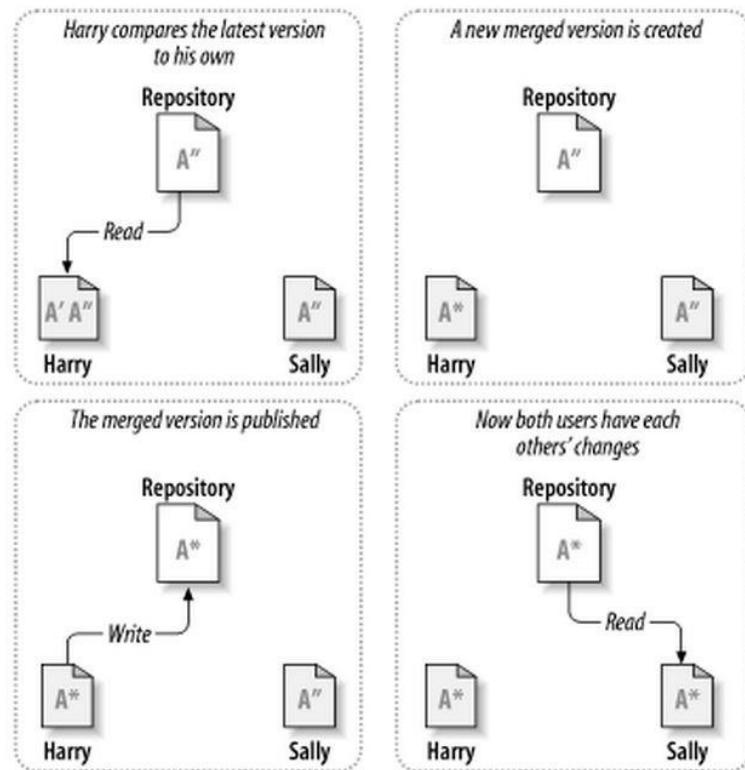


Abbildung13: "Kopieren-Ändern-Zusammenfassen" Teil 2

Quelle: <http://svnbook.red-bean.com/de/1.6/svn.basic.version-control-basics.html>

Git gehört zu den verteilten Versionskontrollsystmen. Das bedeutet, dass es kein zentrales Archiv (Repository) hat, sondern jeder Anwender über ein vollwertiges lokales Repository verfügt, in dem Änderungen durchgeführt und gespeichert werden. Nur wenn ein Abgleich mit Entwicklungszweige (Branches) in anderen Archiven stattfindet führt Git einen Netzwerkbefehl aus.<sup>11</sup> Jeder Anwender kann also lokal für sich an einem beliebigen Ort arbeiten und nur für ein Zusammenfügen der verschiedenen Versionen oder ein Download einer Projektkopie ist ein Netzzugriff nötig.

## 4.2 Bare-Repository Hosting

Im Gegensatz zu vielen anderen verteilten Versionskontrollsystmen kommt Git ohne zentralen Server aus.<sup>12</sup> Um Open-Source-Projekte zu verwalten wird jedoch der Einsatz eines Servers notwendig. Eine Möglichkeit ist das Einrichten eines eigenen Git-Servers, die andere Variante ist die Nutzung eines Hosting-Anbieters für Git. Dies hat den Vorteil, dass die Konfiguration

<sup>11</sup> <http://jaxenter.de/artikel/Git-ein-wahrer-Teamplayer-164409>

<sup>12</sup> <https://faq.hosteurope.de/?cpid=17810>

schneller durchgeführt werden kann und keine Wartung und Überwachung wie etwa bei einem eigenen Server durchgeführt werden muss.

GitHub ist die derzeit größte Open-Source Git Hosting Plattform und bietet sowohl öffentliche, als auch private Hosting-Optionen. Somit können also auf einfache Art und Weise Open-Source-Projekte und proprietärer Code gemeinsam auf einer Plattform verwaltet werden.<sup>13</sup>

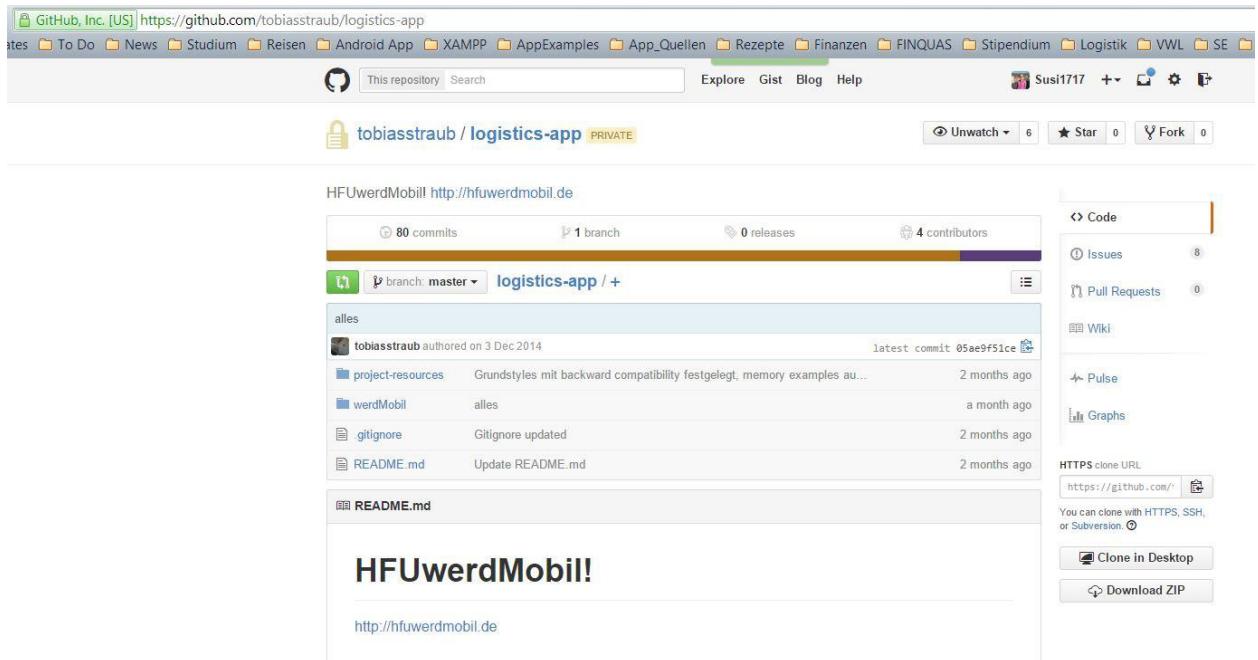
Vor allem bei der Arbeit in verteilten Teams, sobald wie bei diesem Semesterprojekt mehrere Entwickler von verschiedenen Computern aus gemeinsam an einem Projekt arbeiten, wird ein sogenanntes Bare Git Repository benötigt. Es handelt sich hierbei um ein zu Beginn leeres Archiv (Repository) mit dessen Hilfe man die Entwicklung in verteilten Teams koordinieren kann. Das Bare Repository liegt auf einem Host, in diesem Fall bei dem Anbieter GitHub, der als zentraler Server fungiert. Dieses Repository kann nun von allen an der Entwicklung beteiligten Nutzern eingesehen und geteilt werden. Dieses Archiv dient als Quelle für die lokalen Entwicklungszweige (Branches), aus welchem die Nutzer die neuste Version des Archivs vom Server in das lokale Verzeichnis speichern können. Zudem werden in diesem zentralen Bare Git Repository Änderungen gebündelt und zusammengefügt, welche die Nutzer in ihren lokalen Verzeichnissen vornehmen und dann in das Archiv auf dem zentralen Server hochladen.<sup>14</sup>

Im Rahmen dieses Semesterprojektes wurde vom Teamleiter ein Repository für die App HFU-  
werdmobil! erstellt (siehe: <http://bit.ly/hfuwerdmobil-repo>). Alle beteiligten Teammitglieder nutzen das Archiv um sowohl Quellcode, als auch Textdateien, Spezifikationen, Images, Nachrichten und Entwürfe etc. auszutauschen.

---

<sup>13</sup> <http://git-scm.com/book/de/v1/Git-auf-dem-Server-Git-Hosting>

<sup>14</sup> <http://www.saintsjd.com/2011/01/what-is-a-bare-git-repository/>



**Abbildung 14: HFUwerdMobil! Repository auf GitHub**

Quelle: <http://bit.ly/hfuwerdmobil-repo> (nur als eingeladenes Mitglied des Git-Repositories abrufbar)

## 5. Kommunikation

Aufgrund der Tatsache, dass die App als Kern eines großen Gesamtprojektes zu sehen ist, stellt Kommunikation ein zentrales Thema dar und ist in ihrer Wichtigkeit unabdingbar.

Differenzieren kann man hierbei nach der internen Kommunikation innerhalb der sechs Teammitglieder umfassenden Gruppe, sowie Kommunikation nach außen. Dies umfasst sowohl den Kontakt zu anderen Gruppen, sowie den zu Auftraggebern.

### 5.1 Kommunikation intern

Die Kommunikation innerhalb des Teams findet über diverse Kanäle statt.

Programmiertechnische Aspekte werden über das Versionsverwaltungstool Git und dessen Plattform GitHub geregelt. Hierbei werden zum einen sogenannte Issues (zu lösende Aufgaben) erstellt, verwaltet und kommuniziert. Darüber hinaus wird programmiert Quellcode hochgeladen und zusammengefügt und die Dokumentation entsprechend aktualisiert.

Allgemeine Themen, Organisatorisches und Termine werden in einer geschlossenen Facebookgruppe kommuniziert. Da jedes Teammitglied über ein Smartphone mit mobilem Internet verfügt stellt dies den einfachsten Weg dar, Teaminterna schnell und einfach auszutauschen.

Des Weiteren wird die Cloud-Software Dropbox verwendet um einfache Dateien allen Teammitgliedern zur Verfügung zu stellen. Jedes Teammitglied verfügte bereits über einen entsprechenden Account bei dem Anbieter Dropbox. Somit stellt ein geteilter Ordner, auf den die Mitglieder zu jedem Zeitpunkt zugreifen können, eine einfache und gänzlich unkomplizierte Lösung dar. Ausgetauscht werden über die Cloud-Software beispielsweise einfache Dateien wie Bilder, Screenshots, Formatvorlagen, Diagramme und Logos, sowie Präsentationen und Excel-Dateien.

## 5.2 Kommunikation mit anderen Gruppen

Die App-Gruppe steht zentral innerhalb des Gesamtprojektes zwischen den Gruppen OBD und Server. Das oberste Glied über der Server-Gruppe ist Frontend-technisch die Webseiten-Gruppe.

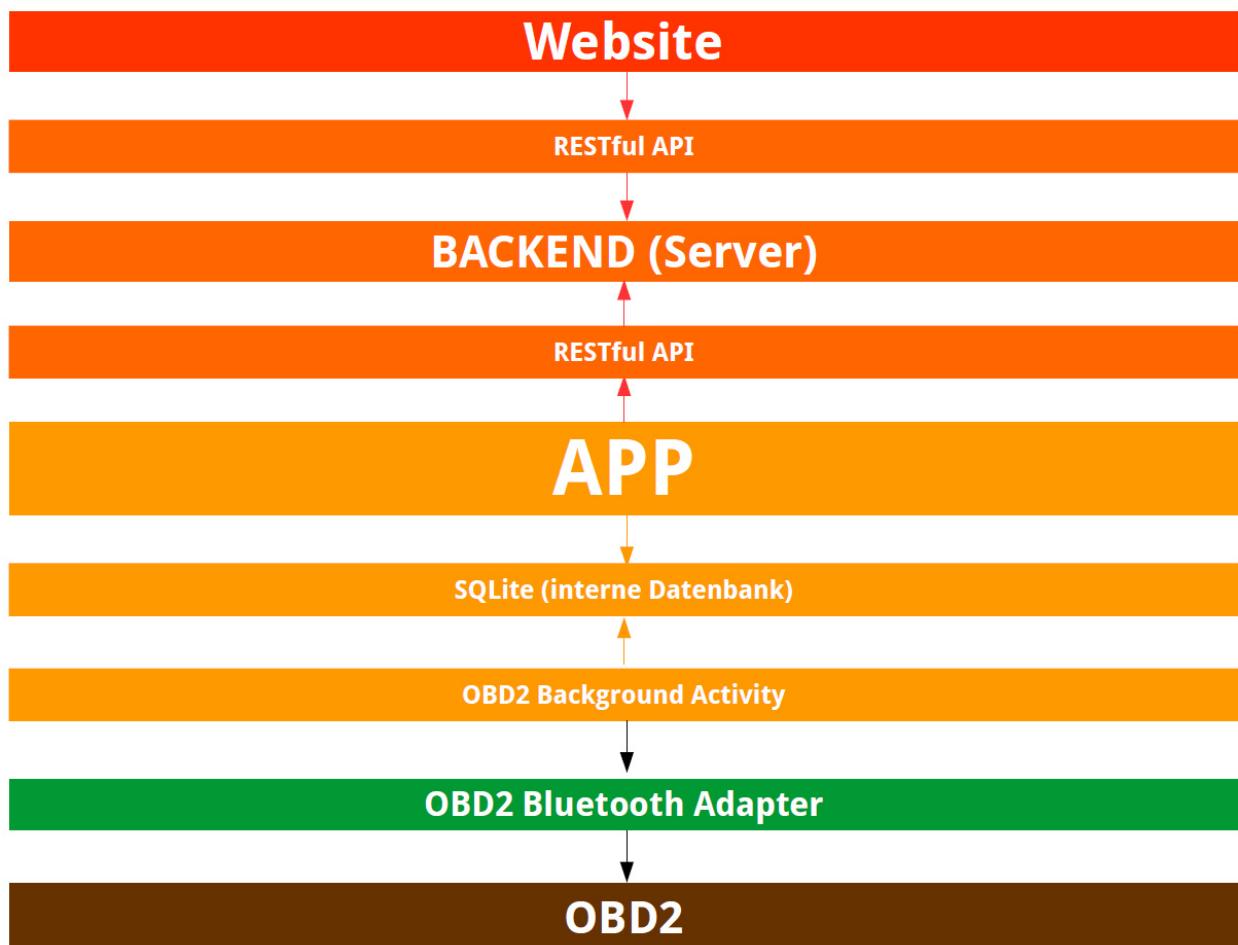


Abbildung 15: Projektaufbau der Gruppen

Um eine gute und konsistente Zusammenarbeit zu gewährleisten bedarf es einem von allen Teamleitern akzeptiertem Kommunikationsmittel. Hier wurde sich durch Unterstützung der Projektleiter auf eine Kommunikation via Facebook geeinigt. Über eine entsprechende Gruppe er-

folgen Terminvereinbarungen zu regelmäßigen Treffen und ein Austausch über den Status Quo der einzelnen Gruppen etc.

Der Austausch von Daten wie etwas Präsentationsfolien erfolgt über die E-Mail Plattform der HFU. Größere Dateien werden allen Teamleitern sowie den Projektleitern über einen gemeinsamen Ordner in Dropbox zu Verfügung gestellt.

### 5.3 Kommunikation mit Auftraggebern

Auftraggeber im Rahmen dieses Semesterprojektes im Bereich Logistik und Supply Chain Management ist in erster Linie Professor Jochen Baier. Er definiert von funktioneller Seite die Anforderungen an das Projekt und entscheidet über Projektaufbau sowie die saubere und einwandfreie Einbindung aller Schnittstellen.

Softwareseitig fungiert Professor Oliver Taminé als Auftraggeber. Er legt fest, welche Anforderungen programmierseitig an das Projekt gestellt werden.

Die Kommunikation mit beiden Auftraggebern erfolgt in erster Linie in der entsprechenden Vorlesung in persönlichen Gesprächen. Falls ein längerer Zeitrahmen nötig ist, wird ein Termin außerhalb der Vorlesung vereinbart.

Darüber hinaus findet ein Austausch über relevante Themen, Probleme und zu klärende Sachverhalte via E-Mail statt.

## 6. Anforderungsmanagement

Das Anforderungsmanagement spielt in jedem Softwareentwicklungsprojekt eine gewichtige Rolle. Anforderung bzw. engl. requirement ist dabei eine Aussage über eine zu erbringende Eigenschaft/Leistung eines Softwareproduktes. In unserem Projekt entschieden wir uns für einen Ansatz der agilen Softwareentwicklung (in Anlehnung an Scrum, XP, etc.) die sogenannte User-Story. Diese bewusst kurz formulierten Anforderungen an die Software entstanden nach der Vorlage „Als [Rolle] möchte ich [Ziel], um [Nutzen]“. Zudem nahmen wir eine erste Priorisierung vor (Wichtigkeit) und setzten den Status zunächst auf „To-Do“. Das Product-Backlog nach Scrum-Vorbild ist sodann eine geordnete Auflistung der product requirements. Nachfolgend werden unter 6.1 und 6.2 die vom Team erstellten User-Stories und der Product-Backlog zu Beginn der Entwicklungsphase dargestellt. Sämtliche Stories/Tasks sind (Stand Januar 2015) mit Abgabe der Arbeit freilich auf den Status „Done“ und somit erledigt gesetzt.

## 6.1 User Stories

User Stories

ID	Als...	möchte ich...	um...	Bemerkungen	Wichtigkeit	Status
3	App-Nutzer	beim Start der App auswählen können, eine Route aufzuzeichnen	meine Fahrstrecke aufzuzeichnen		1	ToDo
4	App-Nutzer	beim Start der App auswählen können, meine bereits aufgezeichneten Routen anzuseigen	eine Übersicht der bereits aufgezeichneten Fahrstrecken zu erhalten		1	ToDo
6	App-Nutzer	dass Aufzeichnen der Route beenden können	wieder zum Startbildschirm zurückzukehren		1	ToDo
8	App-Nutzer	nach der Auswahl einer bestimmten Route in 'Meine bereits aufgezeichnete Routen' die ausgewählte Route in einer Google Maps Karte angzeigt bekommen	meine Route nachvollziehen zu können		1	ToDo
13	App-Nutzer	die App in deutscher Sprache nutzen können	die App in meiner Muttersprache zu verwenden		1	ToDo
5	App-Nutzer	nach der Auswahl von 'Route aufzeichnen' eine Rückmeldung erhalten	zu wissen dass die App ab jetzt meine Fahrstrecke aufzeichnet		2	ToDo
15	App-Nutzer	mit einem Klick zum Impressum der App gelangen	zu wissen, dass die App Betreiber deutsches Recht (Impressumspflicht) einhalten		2	ToDo

Abb.16 : User Stories Seite 1, Status: Beginn der Entwicklung

User Stories

1	App-Nutzer	dass nach dem Anklicken von 'Route aufzeichnen', Bluetooth automatisch aktiviert wird	Bluetooth nicht manuell aktivieren zu müssen	Möglicherweise wurde Bluetooth schon vorher aktiviert	3	ToDo
2	App-Nutzer	dass nach dem Beenden der App-Tasks, Bluetooth wieder deaktiviert wird	den Akku des App-Nutzers zu schonen		3	ToDo
9	App-Nutzer	nach der Auswahl einer bestimmten Route in 'Meine Routen' für die ausgewählte Route in Textform die gefahrenen Kilometer erhalten	zu wissen wie viele Kilometer diese Route beträgt		3	ToDo
11	App-Nutzer	nach der Auswahl einer bestimmten Route in 'Meine Routen' für die ausgewählte Route in Textform die durchschnittlich gefahrene Geschwindigkeit erhalten	mein Fahrverhalten in Zukunft möglicherweise ökonomischer zu beeinflussen		3	ToDo
12	App-Nutzer	nach der Auswahl einer bestimmten Route in 'Meine Routen' für die ausgewählte Route in Textform den durchschnittlichen Verbrauch erhalten	mein Fahrverhalten in Zukunft möglicherweise ökonomischer zu beeinflussen		3	ToDo
10	App-Nutzer	nach der Auswahl einer bestimmten Route in 'Meine Routen' für die ausgewählte Route in Textform die Auflistung aller Orte erhalten, durch die diese Route führt	zu wissen welche Orte ich durchfahren habe		4	ToDo

Abb.17 : User Stories Seite 2, Status: Beginn der Entwicklung

**Abb.18 : User Stories Seite 3, Status: Beginn der Entwicklung**

## 6.2 Product- Backlog

Backlog				
Item	Story ID	Story / Task	Zugewiesen	Status
				Prio
1	0	Allgemeines		in Progress
2	0.1	Einrichtung & Konfiguration Versionskontrollsysteem (Git)	ts	Done
3	0.2	Namensgebung App	sr	in Progress
4	0.3	Laucher Icon konzipieren und umsetzen	tl	in Progress
5	0.4	UI Theme konzipieren und umsetzen	ts	in Progress
6	0.5	Blankes Android anlegen	ts	in Progress
7	0.6	Wireframes der UI	sr, mf, ef, vk, tl, ts	in Progress
8	0.7	Manifest Permissions kontinuierlich überprüfen	n/a	ToDo
9		Als App-Nutzer möchte ich die App in deutscher Sprache nutzen können um die App in meiner Muttersprache 13 zu verwenden		in Progress
10		Andauernde Überprüfung der String Ressourcen auf die Einhaltung deutscher Begrifflichkeiten (inklusive 13.1 Grammatik und Rechtschreibung)	tl	in Progress
11		Als App-Nutzer möchte ich globale Einstellungen an einem Ort in der App durchführen können, um 18 Einstellungen nicht in Activities verstreut zu handeln		in Progress
12		18.1 Anlegen der Activity + Erweiterung der UI um zur neuen Activity zu gelangen	tl	in Progress
13		Als App-Nutzer möchte ich beim erstmaligen Start der App meine persönlichen Daten angeben können, um sie 17 nicht bei jeder Route angeben zu müssen		in Progress
14	17.1	Definition von persönlichen Daten (Welche Daten?)	ts	in Review
15	17.2	SQLite Schema der persönlichen Daten definieren und realisieren	ts	in Review
16	17.4	„Setting“-Activity aus Story#18 um UI-Elemente für die Eingabe der persönlichen Daten erweitern	n/a	ToDo
17		Logik zur Prüfung „Hat der User die App zum ersten mal gestartet?“ implementieren und im benannten Falle, 17.5 zur „Setting“-Activity weiterleiten	n/a	ToDo
18		17.6 Daten dauerhaft in SQLite persistieren	ts	in Review
19		Als App-Nutzer möchte ich in den Settings der App angegeben können, ob ich die OBD-Daten auslesen möchte oder nicht, um im Falle eines nicht vorhandenen OBD-Adapters der App keine unnötigen 19 Schwierigkeiten zu bereiten		ToDo
20	19.1	SQLite Schema aus Story#17 erweitern	n/a	ToDo
21	19.2	„Setting“-Activity aus Story#18 um UI-Elemente für die on/off-Einstellung des OBD erweitern	n/a	ToDo
22	19.3	Zustand dauerhaft in SQLite persistieren	n/a	ToDo
23		Als App-Nutzer möchte ich nach dem Start der App auswählen können, eine Route aufzuzeichnen um meine 3 Fahrtstrecke aufzuzeichnen		in Progress
24	3.1	Anlegen der Activity + Erweiterung der UI um zur neuen Activity zu gelangen (mit Angabe des Zielortes)	n/a	ToDo

**Abb.19 : Product Backlog Seite 1, Status: Beginn der Entwicklung**

Backlog					
25	SQLite Schema für GPS-Daten, sowie Start und Ziel definieren (Start ist immer aktuelle Position, Ziel kann nur 3.2 Furtwangen, Schwenningen, Tuttlingen, Heimatort sein) und realisieren	ts	in Review	2	
26	3.3 Prüfen, ob GPS aktiviert und ggf. starten	n/a	ToDo	2	
27	3.4 Logik implementieren, damit kontinuierlich GPS-Daten in SQLite persistiert werden	n/a	ToDo	2	
28	3.5 Prüfen, ob OBD-Daten gelesen werden sollen (> Settings) und ggf. Bluetooth aktivieren	n/a	ToDo	2	
29	3.6 OBD2 Teilprogramm implementieren und als Background-Activity starten (nur wenn OBD-Daten auch gelesen werden sollen => Settings)	n/a	ToDo	2	
30	Als App-Nutzer möchte ich nach der Auswahl von 'Route aufzeichnen' eine Rückmeldung erhalten um zu wissen, dass die App ab jetzt meine Fahrtstrecke aufzeichnet		ToDo	2	
31	5.1 Erweiterung der UI aus Resultat der Story#3	n/a	ToDo	2	
32	Als App-Nutzer möchte ich dass Aufzeichnen der Route beenden und eine Informationsübersicht erhalten um wieder über Parameter der Fahrt informiert zu werden		ToDo	2	
33	6.1 Anlegen der Activity + Erweiterung der UI um zur neuen Activity zu gelangen	n/a	ToDo	2	
34	6.2 Ggf. OBD Background Activity canceln	n/a	ToDo	2	
35	6.3 Informationen der Fahrt ausgeben (Route und ggf. OBD-Daten)	n/a	ToDo	2	
36	Als App-Nutzer möchte ich mit einem Klick zum Impressum der App gelangen um zu wissen, dass die App-Betreiber deutsches Recht (Impressumspflicht) einhalten		ToDo	3	
37	15.1 Impressum Activity erzeugen, UI gestalten und mit entsprechendem Content befüllen	n/a	ToDo	3	
38	20 Als App-Nutzer möchte ich meine erfasssten Daten publizieren um Mitfahrer zu gewinnen		in Progress	4	
39	20.1 REST API Endpunkt festlegen und an Backend Gruppe übergeben	ts	in Review	4	
40	Definition, welche Daten und zu welchen Zeitintervallen dem Backend via REST Endpoint zur Verfügung gestellt werden	ts	in Review	4	
41	20.3 stellen Persönliche Daten des App-Nutzers nach Registrierung via REST Endpoint dem Backend zur Verfügung	ts	in Review	4	
42	20.4 Nach dem Ändern der Settings, müssen Nutzer Daten via REST Endpoint beim Backend aktualisiert werden	ts	in Review	4	
43	Während der Routen-Aufzeichnung muss in definiertem Intervall GPS-Daten via REST Endpoint dem Backend zur Verfügung gestellt werden	ts	in Review	4	
44	Nach der Routen-Aufzeichnung müssen bestimmte OBD-Daten via REST Endpoint dem Backend zur Verfügung gestellt werden	ts	in Progress	4	
45	Als App-Nutzer möchte ich nach dem Start der App auswählen können, meine bereits aufgezeichneten Routen auszuwählen um eine Übersicht der bereits aufgezeichneten Fahrtstrecken zu erhalten		ToDo	5	
46	4.1 Anlegen der Activity + Erweiterung der UI um zur neuen Activity zu gelangen	tl	ToDo	5	
47	UI erweitern durch Auflistung aller in der internen Datenbank befindlichen aufgezeichneten Routen des Benutzers	tl	ToDo	5	

Abb.20 : Product Backlog Seite 2, Status: Beginn der Entwicklung

Backlog					
48	Als App-Nutzer möchte ich nach der Auswahl einer bestimmten Route in 'Meine bereits aufgezeichneten Routen' die ausgewählte Route in einer Google Maps Karte angezeigt bekommen um meine Route 8 nachzuvollziehen zu können		ToDo	5	
49	8.1 Google Maps API Key erzeugen	n/a	ToDo	5	
50	8.2 UI erweitern, durch das Einbinden der Google Map + Route in Map zeichnen	n/a	ToDo	5	
51	Als App-Nutzer möchte ich nach der Auswahl einer bestimmten Route in 'Meine Routen' für die ausgewählte 9 Route in Textform die gefahrenen Kilometer erhalten um zu wissen wie viele Kilometer diese Route beträgt		ToDo	6	
52	9.1 Auslesen und ggf. Berechnung der entsprechenden Daten aus der SQLite Datenbank	n/a	ToDo	6	
53	9.2 UI entsprechend erweitern	n/a	ToDo	6	
54	Als App-Nutzer möchte ich nach der Auswahl einer bestimmten Route in 'Meine Routen' für die ausgewählte 11 möglicherweise ökonomischer zu beeinflussen Route in Textform die durchschnittliche Geschwindigkeit erhalten um mein Fahrverhalten in Zukunft		ToDo	6	
55	11.1 Auslesen und ggf. Berechnung der entsprechenden Daten aus der SQLite Datenbank	n/a	ToDo	6	
56	11.2 UI entsprechend erweitern	n/a	ToDo	6	
57	Als App-Nutzer möchte ich nach der Auswahl einer bestimmten Route in 'Meine Routen' für die ausgewählte 12 ökonomischer zu beeinflussen Route in Textform den durchschnittlichen Verbrauch erhalten um mein Fahrverhalten in Zukunft möglicherweise		ToDo	6	
58	12.1 Auslesen und ggf. Berechnung der entsprechenden Daten aus der SQLite Datenbank	n/a	ToDo	6	
59	12.2 UI entsprechend erweitern	n/a	ToDo	6	
60	Als App-Nutzer möchte ich nach der Auswahl einer bestimmten Route in 'Meine Routen' für die ausgewählte 10 durchfahren habe Route in Textform die Auflistung aller Orte erhalten, durch die diese Route führt, um zu wissen welche Orte ich		ToDo	7	
61	10.1 Anlegen der Activity + Erweiterung der UI um zu einer neuen Activity zu gelangen	n/a	ToDo	7	
62	10.2 Auslesen und ggf. Berechnung der entsprechenden Daten aus der SQLite Datenbank	n/a	ToDo	7	
63	16 Als App-Nutzer möchte ich wissen, wer die App entwickelt hat, um Danke zu sagen		ToDo	8	
64	16.1 Team Activity erzeugen, UI gestalten und mit entsprechendem Content befüllen	n/a	ToDo	8	
65	Als App-Nutzer möchte ich auf eine Hilfeseite gelangen, wenn ich Hilfe benötige um eine Lösung für mein 14 Problem zu finden		ToDo	9	
66	14.1 Hilfe Activity erzeugen, UI gestalten und mit entsprechendem Content befüllen	n/a	ToDo	9	

Abb.21 : Product Backlog Seite 3, Status: Beginn der Entwicklung

## 7. Wireframing

Für den ersten konzeptionellen Entwurf der verschiedenen Activities und ihrer zugehörigen Screens inklusive Texteingabefeldern, Buttons, Kartenfragmenten etc. verwendet wird das Verfahren des sogenannten Wireframings. Der Ausdruck Wireframe, welcher in etwa mit „Drahtgerüst“ übersetzt werden kann, steht hier für einen visuellen Prototyp, ein frühes Conceptual Design unserer Applikation. Das Layout und der Funktionalitätsumfang einiger dieser Layouts finden sich nahezu identisch umgesetzt in der fertigen Applikation wieder (z.B. Wireframes Einstellungen und Routendetails, vgl. Abb. 22).

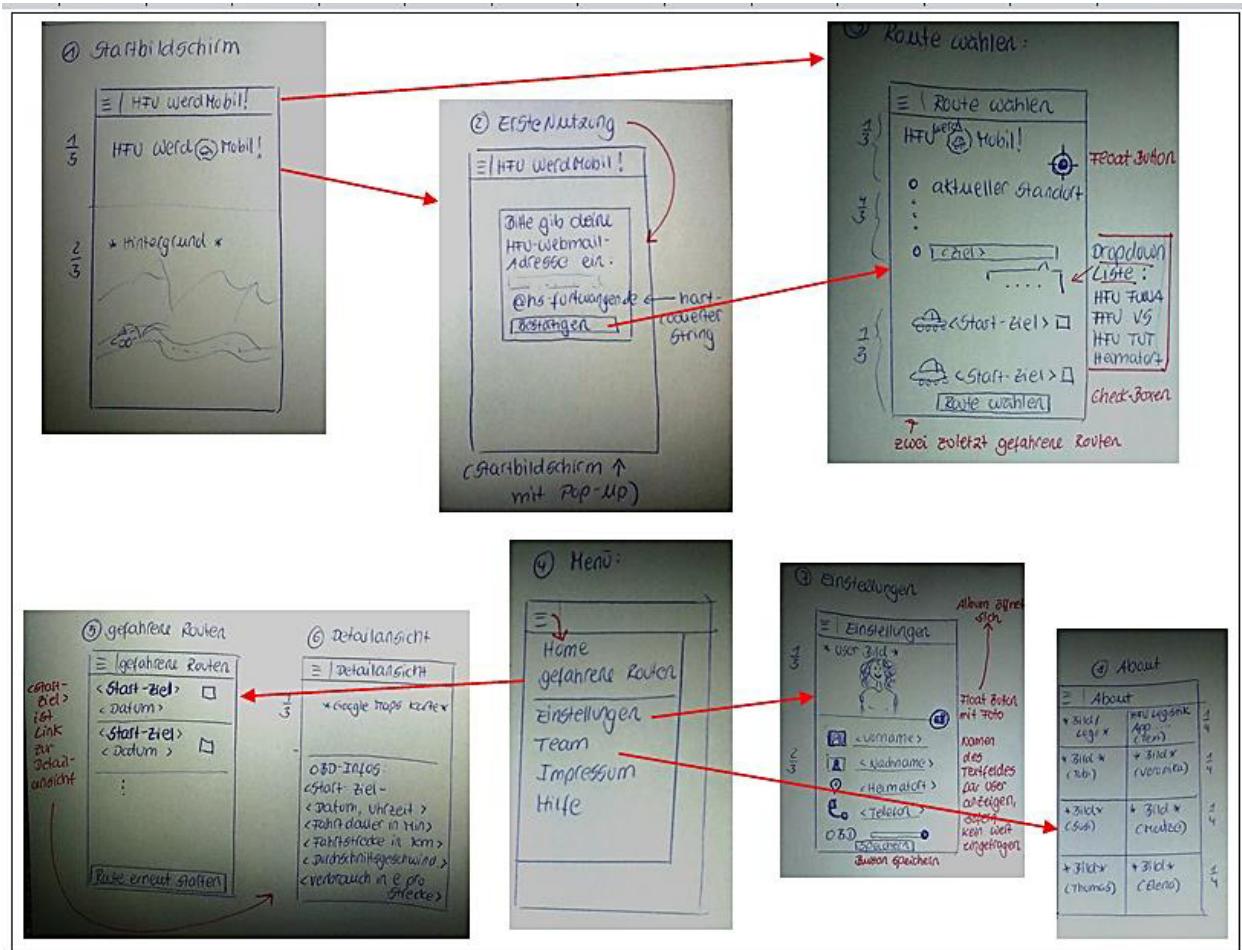


Abbildung 22: Übersicht Wireframes

## 8. RESTful API

Representational State Transfer (REST) ist ein Architekturpattern bzw. –stil für die Entwicklung skalierbarer Webservices, der aus Richtlinien und best practices besteht.

Die Idee hinter REST ist, dass man statt komplexer Mechanismen wie CORBA, RPC oder SOAP schlicht HTTP für Calls zwischen Maschinen einsetzt. RESTful Applikationen nutzen HTTP um Daten zu posten (create, update), zu lesen (z.B. Queries) und zu löschen. Somit benutzt REST HTTP für sämtliche CRUD-Operationen.<sup>15</sup>

Für die Kommunikation zwischen Applikation und Backend, sowie zwischen Backend- und Frontend wurde der Einsatz einer RESTful API beschlossen. Da es sich bei REST nicht um einen Standard handelt, es also z.B. keine w3c recommendation dazu gibt, ist eine eigene Spezifikation für das Projekt notwendig. Diese Spezifikation der RESTful API für HFUwerdMobil! wurde vom Teamleiter der Applikationsgruppe Tobias Straub geschrieben und allen Gruppen zur Verfügung gestellt.

---

<sup>15</sup> Elkstein, “Learn REST – a tutorial, 1. What is REST?”

## 8.1 Spezifikation RESTful API (final)

### **HFUwerdMobil!** Spezifikation RESTful API

**Autor & Ansprechpartner**  
Tobias Straub  
[t.straub@hs-furtwangen.de](mailto:t.straub@hs-furtwangen.de)

Internal Version: 3.3

External Version: 1.0

---

#### **Authentifizierung**

Bei GET-Requests: Durch Anhängen des Token an den Parameter ?token=

Bei POST- und PUT-Requests: Das Token befindet sich im JSON-Objekt unter "token"

Token Berechnung: sha256({user\_email}+{API\_KEY}+{API\_ENDPUNKT}+{JSON\_OBJEKT})

#### Anmerkungen:

- Bildung des Hashes durch die Hashfunktion „SHA-2“ mittels SHA-256
- {user\_email} ist beim Aufruf des API Endpunktes zur Änderung der Nutzerdaten bereits die neue E-Mail (falls geändert), ansonsten die bisherige (nicht geänderte)
- {API\_KEY} lautet: `/u8teH}@A_&&sQ%3DUsbW/m88Wk,K` (API\_KEY muss „geheim“ bleiben)
- {API\_ENDPUNKT} ist gesamter Endpunkt, z.B. /user/max.mustermann@hs-furtwangen.de/route/873
- {JSON\_OBJEKT} ist das gesamte JSON-Objekt OHNE führende und OHNE endende geschweifte Klammer mit token Bezeichner einschließlich Doppelpunkt aber ohne Value, bei GET-Requests hat das JSON-Objekt den Wert "" (Leerstring)  
Beispiel:  
`"user_email": "max.mustermann@gmail.com",  
"firstname": "Max",  
"lastname": "Mustermann",  
"mobile_number": "49776523456",  
"hometown_zip": "78098",  
"token":`
- Stimmt gelieferter Token nicht mit dem auf serverseite berechneten Token überein, muss der Request immer mit dem HTTP Response 401 abgelehnt werden

---

#### **Verifizierungscode anfordern**

API-Endpoint: /verify/{user\_email} /\* z.B. max.mustermann@hs-furtwangen.de \*/  
HTTP-Method: GET

Erwartete Rückgabe:

- HTTP Response Code 200 (ok) + JSON-Objekt (mit Beispielinhalt):  
`{  
 "verify_code":123456  
}`

Anmerkung: Es wird ein zufälliger, sechs-stelliger Verifizierungscode generiert und an {user\_email} verschickt, mit der Aufforderung, diesen Verifizierungscode in der App einzugeben. Zusätzlich wird dieser als JSON-Objekt dem Request zurückgegeben.

---

#### **App Nutzer anlegen**

API-Endpoint: /user/new

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_email": "max.mustermann@gmail.com",  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098",  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
  - HTTP Response Code 409 (conflict): {user\_email} schon vorhanden
  - HTTP Response Code 401 (unauthorized): ungültiger Token
- 

#### **App Nutzer ändern**

API-Endpoint: /user/{{old\_user\_email}} /\* z.B. max.mustermann@gmail.com \*/

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_email": "max@mustermann.com", /* Bei keiner Änderung der E-Mail Adresse =>  
    old_user_email */  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098",  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
  - HTTP Response Code 424 (failed dependency): {{old\_user\_email}} nicht vorhanden
  - HTTP Response Code 401 (unauthorized): ungültiger Token
- 

#### **App Nutzer Daten abrufen**

API-Endpoint: /user/{user\_email}

HTTP-Method: GET

Erwartete Rückgabe:

- HTTP Response Code 200 (ok) + JSON-Objekt (mit Beispielinhalt):

```
{  
    "user_email": "max.mustermann@gmail.com",  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
- HTTP Response Code 424 (failed dependency): {user\_email} nicht gefunden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

#### Bei Beginn einer Routen Aufzeichnung (User klickt auf Route aufzeichnen)

API-Endpoint: /user/{user\_email}/route/new

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "route_id": "1cf2d0dfdf09e8e11421338948",  
    "timestamp": "123456789",  
    "start_point": {  
        "latitude": "37.422005",  
        "longitude": "-122.084095"  
    },  
    "end_point": "1", /* 0: User Heimatort; 1: Furtwangen; 2: Schwenningen; 3: Tuttlingen */  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete HTTP-Rückgabewerte

- HTTP Response Code 200 (ok)
- HTTP Response Code 409 (conflict): {route\_id} schon vorhanden
- HTTP Response Code 424 (failed dependency): {user\_email} nicht vorhanden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

#### Während einer Routen Aufzeichnung

API-Endpoint: /user/{user\_email}/route/{route\_id}

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "positions": [  
        {  
            "timestamp": "12569537329",  
            "lat": 37.422005,  
            "lon": -122.084095  
        }  
    ]  
}
```

```

        "latitude": "38.123456",
        "longitude": "-47.654321"
    },
    {
        "timestamp": "12569564253",
        "latitude": "42.123456",
        "longitude": "-66.654321"
    },
    ( ......... )
],
"token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"
}

```

Erwartete HTTP-Rückgabewerte

- HTTP Response Code 200 (ok)
- HTTP Response Code 424 (failed dependency): {route\_id} nicht gefunden
- HTTP Response Code 401 (unauthorized): ungültiger Token

Anmerkung: Alle X-Minuten wird ein Satz von GPS-Daten an den API-Endpunkt versendet.  
Nach weiteren X-Minuten wird ein neuer Satz von GPS-Daten an den API-Endpunkt  
gesendet. Der Vorgang wiederholt sich solange, bis die Routen Aufzeichnung beendet  
wird (dafür wird ein entsprechender API-Endpunkt aufgerufen, siehe weiter unten)

---

#### **Nach Ende der Routen Aufzeichnung (User klickt auf Route aufzeichnen beenden)**

API-Endpoint: /user/{user\_email}/route/end

HTTP-Method: POST

Gelieferte Daten (mit Beispieldaten):

```

{
    "route_id": "123456",
    "timestamp": "123456789",
    "total_duration_seconds": "1500",
    "total_kilometers": "25",
    "obd_info": { /* Sofern OBD aktiv war, ansonsten werden die OBD Values mit null gefüllt */
    },
    "average_speed": "70",
    "average_consumption": "3"
},
"token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"
}

```

Erwartete HTTP-Rückgabewerte

- HTTP Response Code 200 (ok)
  - HTTP Response Code 409 (conflict): Route wurde schon beendet
  - HTTP Response Code 424 (failed dependency): {route\_id} nicht vorhanden
  - HTTP Response Code 401 (unauthorized): ungültiger Token
- 

#### **Anmerkungen**

- `{user_email}` kann als Primärschlüssel verwendet werden
- `{route_id}` ist nur für den jeweiligen App-Nutzer eindeutig (nicht global), aber auf allen seinen Android Geräten
- Beim Start der Route (nach Aufruf von `/user/{user_email}/route/new`), beim Zufügen von Routen Daten (nach Aufruf von `/user/{user_email}/route/end`) und beim Beenden der Route (nach Aufruf von `/user/{user_email}/route/end`) muss die Websites-Gruppe über das Update informiert werden (Ping)! - Das kann ähnlich wie das die API-Designer bei Facebook implementiert haben, realisiert werden (siehe: <https://developers.facebook.com/docs/graph-api/real-time-updates/v2.2>). Oder einfacher als hardcodierte Ping URL. Mit dem Ping, werden die benötigten Daten per HTTP-GET als JSON mitgeliefert.

## **9. Dokumentation Softwaresystem**

### **9.1 Speichermanagement**

Vor allem Datenbankanwendungen, welche in Softwaresystemen verwendet werden, lassen die Datenmengen ansteigen, wodurch eine Erhöhung der Speicherkapazität nötig wird. Diese Speicherkapazitäten gilt es zu verwalten. Da diese Speichersysteme über diverse Berührungspunkte zu anderen Systemen und Netzwerken verfügen, gehen deren Aufgaben weit über die des Backups und Recoveries hinaus. Speichermanagement umfasst generell drei große Aufgabenbereiche: Backup-, Problem- und Change-Management. Bei Backup-Management sind inkrementelle und differenzierte Speicherung von großer Wichtigkeit sowie die Online-Speicherung bzw. Spiegelung. Eine Konfiguration der Speichereinheiten und die Verfolgung von Konfigurationsänderungen sind wesentliche Bestandteile des Change-Management. Des Weiteren fallen im Speichermanagement Aufgaben bezüglich der dynamischen Zuteilung, der Kapazitätsplanung und der Strukturierung der Datenarchive an.<sup>16</sup>

Bei Android nimmt das Speichermanagement eine besondere Rolle ein. Zwar verfügt das Android Betriebssystem über einen Linux-Kernel, hat jedoch in seiner Systemarchitektur einige Besonderheiten, die sich gravierend von anderen Linux-Versionen unterscheiden. Android ist eine von der Open Handset Alliance entwickelte Open-Source-Plattform. Diese ist ein breiter Zusammenschluss von Softwarefirmen, Netzbetreibern, Smartphone- und Chip-Herstellern; allen voran Google.

Ein normales Linux würde die CPU zu sehr belasten wodurch Anwendungen im Multitasking-Betrieb deutlich langsamer würden. Zudem würde sich dies deutlich bei der Akku-Leistung des Geräts bemerkbar machen. Aus diesen Gründen verfügt das Android-Betriebssystem über einen angepassten Linux-Kernel 2.6. Zudem wurde eine Vielzahl von Treibern und Bibliotheken angepasst oder sogar vollständig ersetzt. Auch hier setzt Android auf eine besondere Technologie, abweichend von Standard-Linux-Systemen: mit dem Power-Management wird das Linux-Kernel an das Android-System angepasst. Hintergrundbeleuchtung von Tastatur und Bildschirm müssen vom System ein- und ausgeschaltet werden können. Zudem muss das System den Akkustatus

---

<sup>16</sup> <http://www.itwissen.info/definition/lexikon/Speichermanagement-storage-management.html>

überwachen und bevor eine Stromversorgung zusammenbricht das Gerät zuverlässig herunterfahren.<sup>17</sup>

### 9.1.1 Shared Preferences

Android stellt mehrere Möglichkeiten zur Verfügung Applikationsdaten persistent zu speichern. Die Wahl der Lösung hängt von den spezifischen Bedürfnissen ab, nämlich ob die Daten privat für die eigene Applikation oder für andere Applikationen und den Anwender zugänglich sein sollten und wieviel Speicherplatz die Daten benötigen.

Neben der ebenfalls eingesetzten SQLite Datenbank (vgl. unter 8.1.2) gibt es noch Internal Storage (auf dem device memory), External Storage (auf einem shared external memory), Network Connection (auf einem webserver) und Shared Preferences (zur Speicherung von private primitive data in key-value Paaren). Die `sharedPreferences` Klasse der letztgenannten Technik, welche in der vorliegenden Applikation eingesetzt wurde, stellt ein Framework zur Verfügung, welches es erlaubt persistente Key-Value Paare primitiver Daten zu speichern und abzurufen. Diese Daten bleiben über User Sessions hinweg, und selbst wenn man die Applikation terminiert, bestehen. Im Rahmen des Speichermanagements von HFUwerdMobil! wurden SharedPreferences insbesondere eingesetzt, um die User-Daten persistent zu speichern.

### 9.1.2 SQLite Datenbank

SQLite ist eine Software Bibliothek, die eine gemeinfreie, eigenständige, serverunabhängige, transaktionale SQL Datenbank-Engine, deren Weiterentwicklung und Maintenance in Teilen von Mitgliedern des SQLite Consortiums wie der Mozilla Foundation, Bentley und Bloomberg getragen wird.<sup>18</sup> Wie bereits erwähnt (vlg. 8.1.1) bietet Android mehrere Wege Anwender und Applikationsdaten zu speichern. Da es sich bei SQLite um eine light weight database handelt, macht ihr Einsatz gerade auf mobilen Endgeräten mit relativ begrenzten Speicherkapazitäten Sinn.

Wie die größeren relationalen auf SQL basierenden Datenbanken bietet SQLite eine gewohnte SQL-Syntax. Ein weiterer Vorteil ist der Verbrauch von einem nur sehr geringen Speicherplatz zur Laufzeit.

---

<sup>17</sup> <http://www.heise.de/ct/artikel/Innenansichten-1176816.html>

<sup>18</sup> <http://www.sqlite.org/>

Das package `android.database.sqlite` enthält die SQLite database management Klassen, die die Applikation benutzt um Ihre eigene private Datenbank zu verwalten. In HFU-werdMobil! werden die SQLite database Management Klassen in den Klassen `DataAccessHandler.java` und `DataAccessHelper.java` verwendet. Während die Klasse `DataAccessHandler.java` den Datenzugriff zur und von der internen SQLite-Datenbank der App handelt, ist die Klasse `DataAccessHelper.java` ihre Hilfsklasse und definiert die Struktur für die interne Datenbank zur Speicherung App-interner Daten.

`DataAccessHandler.java` stellt dabei entsprechende Methoden zur Verfügung, die es erleichtern auf bestimmte Daten zuzugreifen, sowie bestimmte Daten in die Datenbank einzutragen.

`DataAccessHelper.java` stellt weiterhin, unterstützt durch die Klasse `android.database.sqlite.SQLiteOpenHelper`, Methoden zur Erzeugung sowie zum Update der Datenbank bereit.

Serverunabhängig bedeutet, dass bei SQLite kein Datenbankserver benötigt wird; die Einbindung in die Anwendung geschieht auf einfache Weise durch Hinzufügen der Bibliothek. In der Anwendung HFUwermDobil! wird eine SQLite Datenbank zum Ablegen der internen Daten angelegt. Die umfasst folgende Werte:

Routendaten:

- Routen-ID
- Timestamp (Uhrzeit und Datum)
- Startpunkt (Latitude und Longitude)
- Endpunkt

GPS-Daten:

- GPS-ID
- Routen-ID
- Timestamp (Uhrzeit und Datum)
- Latitude
- Longitude

OBD-Daten:

- OBD-ID
- Routen-ID
- Aktuelle Geschwindigkeit
- Aktueller Verbrauch

Die aus der SQLite zu der externen Datenbank übertragenden Werte sind in der RESTful API näher spezifiziert.

## 9.2 Double-Opt-In Verfahren

Das Double-Opt-In Verfahren aus dem Permission Marketing wird eingesetzt um die Email-Adresse des Nutzers zu verifizieren. Zu diesem Zweck wird eine Email-Nachricht mit einem Verifizierungscode an die angegebene Email-Adresse verschickt. Derartige Email-Nachrichten werden auch als DOI-Mail (Double-Opt-In-Mail) bezeichnet. Ihr Zweck ist es eine Bestätigung des tatsächlichen Inhabers der Email-Adresse zu erhalten und missbräuchliche Anmeldungen mit fremden Email-Adressen zu verhindern. Der Nutzer wird dann dazu aufgefordert den erhaltenen Verifizierungscode einzugeben. Nach erfolgter Eingabe ist er registriert und kann die Applikation HFUwerdmobil! verwenden.

## 9.3 Verbesserungsvorschläge

Nachfolgend liefert das initiale Entwicklerteam noch folgende Verbesserungsvorschläge für zukünftige Weiterentwicklung und Maintenance:

- Fremdschlüsselabhängigkeiten in SQLite einbauen um eine noch sichere Datenintegrität zu gewährleisten
- Datenbankschema => timestamp von int nach float updaten und entsprechende Konstruktoren und Methoden anpassen
- Bei RoutenDetails noch Fahrzeit und Fahrstrecke in km mit aufnehmen
- Primary Key (wie z.B. routeID als long definieren)
- OrtsGPS aus OpenStreetMap ziehen
- User Daten immer zusammenhängend als Objekt liefert, um Datenintegrität zu gewährleisten und bei der Synchronisation redundante Requests zu verhindern

## 10. Fazit und Ausblick

Das Semesterprojekt HFUwerdMobil! soll allen Mitgliedern der Hochschule Furtwangen, unabhängig von Fachbereich oder Standort die Möglichkeit bieten aktiv an der Förderung der nachhaltigen Mobilität von Studenten und Mitarbeitern mitzuwirken. Sei es selbst als Fahrer zu agie-

ren und anderen HFU Mitgliedern eine Mitfahrglegenheit anzubieten oder sich als potentieller Mitfahrer bei einer Fahrgemeinschaft einzuklinken.

Dabei steht nicht nur die bessere Mobilität im Vordergrund. Natürlich bringen Fahrgemeinschaften zahlreiche weitere Vorteile mit sich: eine Schonung der Umwelt durch höhere Auslastung der Fahrzeuge und damit einem geringeren CO<sub>2</sub>-Ausstoß pro Person im Fahrzeug. Zudem wird auf einer gemeinsamen Fahrt die Kommunikation der Studierenden im Auto miteinander gefördert und bietet Raum für interessante Gespräche und ein gegenseitiges Kennenlernen. Durch die besondere Form der Live-Übertragung der Fahrgemeinschaften auf der Webseite ist ein hohes Maß an Spontaneität möglich. Zu erwarten ist, dass sich durch regelmäßige Anbieter von Mitfahrglegenheiten sicherlich eine gewisse Routine einpendeln wird, so dass HFU Mitglieder, die häufig dieselben Strecken fahren bzw. mitfahren, zueinander finden werden.

Die HFUwerdMobil! App als Kern des großen Projektes ist zudem ein interessantes Projekt, das viel Spielraum für weitere Optimierungen und Ausbaumöglichkeiten bildet.

Da Sicherheit im Netz eine zunehmend größere Rolle spielt, nimmt auch der Datenschutz gerade bei persönlichen Daten eine immer höhere Wichtigkeit ein. Insbesondere beim Thema Verschlüsselung der zu übertragenden Daten, sei es Nutzerdaten mit Namen, Handynummer, E-Mailadresse und Wohnort, wie auch bei Nutzer-ID usw. könnte mit weiteren Projekten angesetzt werden.

Nach einer gewissen Nutzungsdauer ist eine Auswertung der Bewertung der Applikation im Google Playstore sinnvoll. Auch über die Kommentarfunktion im Playstore wird sich herausstellen um welche Funktionalitäten die Applikation gegebenenfalls erweitert werden könnte. Hier ist auch ein Feedback der App-Anwender nützlich um die FAQ Seite entsprechend zu erweitern.

Ein weiterer wichtiger Punkt stellt die Maintenance der Software dar.

Wartung kommt in den verschiedensten Bereichen der Softwareentwicklung zum Tragen: Nicht nur in Anbetracht neuerer Android-Versionen, auf die eventuell upgedatet werden muss, sondern auch beim Beseitigen von Fehlern, Bugs oder bei der Nutzungsoptimierung.

Hier macht es vor allem bei größeren Änderungen oder Updates Sinn, eine neue Version der App zu veröffentlichen. Entweder könnte dies in Form von Updates für bestehende, sich schon in Verwendung befindliche Versionen von HfUwerdMobil! geschehen oder als Beta-Release der App.

Die im Rahmen des Semesterprojekts entwickelte App bietet eine Vielzahl von Funktionen und Aufgabenbereichen, sei es Oberflächengestaltung, im Hintergrund verwendete, interne wie extern genutzte Datenbanken, Schnittstelleneinbindung oder die dynamische Darstellung der gefahreneren Route in Google Maps.

Im Rahmen der innovativen, nachhaltigen Zielsetzung des Gesamtprojekts zur Förderung und zum Ausbau studentischer Mobilität, stellt die App eine anspruchsvolle technische Lösung mit Zukunftspotenzial dar. Künftige Semester haben die Möglichkeit hier anzusetzen, den Gedanken weiter zu verfolgen und das Projekt HFUwerdMobil! weiter voranzutreiben.

## 11. Quellenverzeichnis

Android Developers, “ADT Plugin Release Notes”  
<http://developer.android.com/tools/sdk/eclipse-adt.html>  
abgerufen am: 10.01.2015

Android Developers, “Platform Versions”  
<https://developer.android.com/about/dashboards/index.html>  
abgerufen am: 10.01.2015

Arc42, „arc42 architecture template“  
<http://arc42.de/arc42-downloads/arc42-template-v40-DE.pdf>  
abgerufen am: 20.11.2014

Arno Becker, „Innenansichten – Die Architektur von Android“ in c’t: Know how  
<http://www.heise.de/ct/artikel/Innenansichten-1176816.html>  
abgerufen am: 10.01.2015

Ben Collins-Sussman et al., „Grundlagen der Versionskontrolle“  
<http://svnbook.red-bean.com/de/1.6/svn.basic.version-control-basics.html>  
abgerufen am: 22.12.2014

Chip online, “APK installieren: Android-App aufs Handy bringen - Download auch ohne Android Store”  
[http://www.chip.de/artikel/APK-installieren-Android-App-auf-Handy-bringen\\_47069033.html](http://www.chip.de/artikel/APK-installieren-Android-App-auf-Handy-bringen_47069033.html)  
abgerufen am: 13.01.2015

Elkstein, M., “Learn REST – a tutorial, 1. What is REST?”  
<http://rest.elkstein.org/>  
abgerufen am: 14.01.2015

Google Play Store: App Datei Manager (File Manager)  
<https://play.google.com/store/apps/details?id=com.rhmsoft.fm&hl=de>  
abgerufen am: 13.01.2015

Host Europe, „Webhosting - Git Versionsverwaltung“

<https://faq.hosteurope.de/?cpid=17810>

abgerufen am: 22.12.2014

IT Wissen, “Versionskontrolle”

<http://www.itwissen.info/definition/lexikon/Versionskontrolle-version-control-system-VCS.html>

abgerufen am: 22.12.2014

IT Wissen, “Speichermanagement”

<http://www.itwissen.info/definition/lexikon/Speichermanagement-storage-management.html>

abgerufen am: 22.12.2014

Jeff Sutherland and Ken Schwaber, „The official SCRUM guide“

<http://www.scrumguides.org/>

abgerufen am: 22.12.2014

Jon Saints, „What is a bare git repository?“

<http://www.saintsjd.com/2011/01/what-is-a-bare-git-repository/>

abgerufen am: 22.12.2014

Oracle, „Code Conventions for the Java TM Programming Language“

<http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html>

abgerufen am: 23.12.2014

Scott Chacon and Ben Straub, “Pro Git”

<http://git-scm.com/book/en/v2>

abgerufen am: 22.12.2014

SQLite, „sqlite.org“

<http://www.sqlite.org/>

abgerufen am: 22.12.2014

Walid El Sayed Aly, „Git in Kombination mit anderen Systemen: Git - ein wahrer Teamplayer“

<http://jaxenter.de/artikel/Git-ein-wahrer-Teamplayer-164409>

abgerufen am: 22.12.2014

## **12. Anhänge (Versionshistorie - RESTful API Spezifikation, Quellcode, Javadoc, UML Diagramme)**

## 12.1 RESTful API - Versionsgeschichte

### HFUwerdMobil! Spezifikation RESTful API

#### Autor & Ansprechpartner

Tobias Straub

[t.straub@hs-furtwangen.de](mailto:t.straub@hs-furtwangen.de)

Internal Version: 3.2

External Version: 1.0

---

#### Authentifizierung

Bei GET-Requests: Durch Anhängen des Token an den Parameter ?token=

Bei POST- und PUT-Requests: Das Token befindet sich im JSON-Objekt unter "token"

Token Berechnung: sha256({user\_email}+{API\_KEY}+{API\_ENDPUNKT}+{JSON\_OBJEKT})

#### Anmerkungen:

- Bildung des Hashes durch die Hashfunktion „SHA-2“ mittels SHA-256
- {user\_email} ist beim Aufruf des API Endpunktes zur Änderung der Nutzerdaten bereits die neue E-Mail (falls geändert), ansonsten die bisherige (nicht geänderte)
- {API\_KEY} lautet: `/u8teH}@A_&&sQ%3DUsbW/m88Wk,K` (API\_KEY muss „geheim“ bleiben)
- {API\_ENDPUNKT} ist gesamter Endpunkt, z.B. /user/max.mustermann@hs-furtwangen.de/route/873
- {JSON\_OBJEKT} ist das gesamte JSON-Objekt OHNE führende und OHNE endende geschweifte Klammer mit token Bezeichner einschließlich Doppelpunkt aber ohne Value, bei GET-Requests hat das JSON-Objekt den Wert "" (Leerstring)

Beispiel:

```
"user_email": "max.mustermann@gmail.com",
"firstname": "Max",
"lastname": "Mustermann",
"mobile_number": "49776523456",
"hometown_zip": "78098",
"token":
```

- Stimmt gelieferter Token nicht mit dem auf serverseite berechneten Token überein, muss der Request immer mit dem HTTP Response 401 abgelehnt werden

---

#### Verifizierungscode anfordern

API-Endpoint: /verify/{user\_email} /\* z.B. max.mustermann@hs-furtwangen.de \*/  
HTTP-Method: GET

Erwartete Rückgabe:

- HTTP Response Code 200 (ok) + JSON-Objekt (mit Beispielinhalt):

```
{
  "verify_code":123456
}
```



**Anmerkung:** Es wird ein zufälliger, sechs-stelliger Verifizierungscode generiert und an {user\_email} verschickt, mit der Aufforderung, diesen Verifizierungscode in der App einzugeben. Zusätzlich wird dieser als JSON-Objekt dem Request zurückgegeben.

---

### **App Nutzer anlegen**

API-Endpoint: /user/new

HTTP-Method: POST

Gelieferte Daten (mit Beispieldaten):

```
{  
    "user_email": "max.mustermann@gmail.com",  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098",  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
- HTTP Response Code 409 (conflict): {user\_email} schon vorhanden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

### **App Nutzer ändern**

API-Endpoint: /user/{(old\_)user\_email} /\* z.B. max.mustermann@gmail.com \*/

HTTP-Method: PUT

Gelieferte Daten (mit Beispieldaten):

```
{  
    "user_email": "max@mustermann.com", /* Bei keiner Änderung der E-Mail Adresse =>  
    old_user_email */  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098",  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
- HTTP Response Code 424 (failed dependency): {(old\_)user\_email} nicht vorhanden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

### **App Nutzer Daten abrufen**

API-Endpoint: /user/{user\_email}

HTTP-Method: GET

Erwartete Rückgabe:

- HTTP Response Code 200 (ok) + JSON-Objekt (mit Beispielinhalt):

```
{  
  "user_email": "max.mustermann@gmail.com",  
  "firstname": "Max",  
  "lastname": "Mustermann",  
  "mobile_number": "49776523456",  
  "hometown_zip": "78098"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
- HTTP Response Code 424 (failed dependency): {user\_email} nicht gefunden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

#### Bei Beginn einer Routen Aufzeichnung (User klickt auf Route aufzeichnen)

API-Endpoint: /user/{user\_email}/route/new

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
  "route_id": "1cf2d0fdf09e8e11421338948",  
  "timestamp": "123456789",  
  "start_point": {  
    "latitude": "37.422005",  
    "longitude": "-122.084095"  
  },  
  "end_point": "1", /* 0: User Heimatort; 1: Furtwangen; 2: Schwenningen; 3: Tuttlingen */  
  "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete HTTP-Rückgabewerte

- HTTP Response Code 200 (ok)
- HTTP Response Code 409 (conflict): {route\_id} schon vorhanden
- HTTP Response Code 424 (failed dependency): {user\_email} nicht vorhanden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

#### Während einer Routen Aufzeichnung

API-Endpoint: /user/{user\_email}/route/{route\_id}

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
  "positions": [  
    {  
      "timestamp": "12569537329",  
      "lat": 37.422005,  
      "lon": -122.084095  
    }  
  ]  
}
```

```

        "latitude": "38.123456",
        "longitude": "-47.654321"
    },
    {
        "timestamp": "12569564253",
        "latitude": "42.123456",
        "longitude": "-66.654321"
    },
    ( ..... )
],
"token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"
}

```

Erwartete HTTP-Rückgabewerte

- HTTP Response Code 200 (ok)
- HTTP Response Code 424 (failed dependency): {route\_id} nicht gefunden
- HTTP Response Code 401 (unauthorized): ungültiger Token

Anmerkung: Alle X-Minuten wird ein Satz von GPS-Daten an den API-Endpunkt versendet. Nach weiteren X-Minuten wird ein neuer Satz von GPS-Daten an den API-Endpunkt gesendet. Der Vorgang wiederholt sich solange, bis die Routen Aufzeichnung beendet wird (dafür wird ein entsprechender API-Endpunkt aufgerufen, siehe weiter unten)

---

#### **Nach Ende der Routen Aufzeichnung (User klickt auf Route aufzeichnen beenden)**

API-Endpoint: /user/{user\_email}/route/end

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```

{
    "route_id": "123456",
    "timestamp": "123456789",
    "total_duration_seconds": "1500",
    "total_kilometers": "25",
    "obd_info": { /* Sofern OBD aktiv war, ansonsten werden die OBD Values mit null gefüllt */}
    "average_speed": "70",
    "average_consumption": "3"
},
"token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"
}

```

Erwartete HTTP-Rückgabewerte

- HTTP Response Code 200 (ok)
  - HTTP Response Code 409 (conflict): Route wurde schon beendet
  - HTTP Response Code 424 (failed dependency): {route\_id} nicht vorhanden
  - HTTP Response Code 401 (unauthorized): ungültiger Token
- 

#### **Anmerkungen**

- {user\_email} kann als Primärschlüssel verwendet werden
- {route\_id} ist nur für den jeweiligen App-Nutzer eindeutig (nicht global), aber auf allen seinen Android Geräten
- Beim Start der Route (nach Aufruf von /user/{user\_email}/route/new), beim Zufügen von Routen Daten (nach Aufruf von /user/{user\_email}/route/end) und beim Beenden der Route (nach Aufruf von /user/{user\_email}/route/end) muss die Websiten-Gruppe über das Update informiert werden (Ping)! - Das kann ähnlich wie das die API-Designer bei Facebook implementiert haben, realisiert werden (siehe: <https://developers.facebook.com/docs/graph-api/real-time-updates/v2.2>). Oder einfacher als hardcodierte Ping URL. Mit dem Ping, werden die benötigten Daten per HTTP-GET als JSON mitgeliefert.

# **HFUwerdMobil!**

## **Spezifikation RESTful API**

### **Autor & Ansprechpartner**

Tobias Straub

[t.straub@hs-furtwangen.de](mailto:t.straub@hs-furtwangen.de)

Internal Version: 3.1

External Version: 1.0

---

### **Authentifizierung**

Bei GET-Requests: Durch Anhängen des Token an den Parameter ?token=

Bei POST- und PUT-Requests: Das Token befindet sich im JSON-Objekt unter "token"

Token Berechnung: sha256({user\_email}+{API\_KEY}+{API\_ENDPUNKT}+{JSON\_OBJEKT})

### Anmerkungen:

- Bildung des Hashes durch die Hashfunktion „SHA-2“ mittels SHA-256
- {user\_email} ist beim Aufruf des API Endpunktes zur Änderung der Nutzerdaten bereits die neue E-Mail (falls geändert), ansonsten die bisherige (nicht geänderte)
- {API\_KEY} lautet: **/u8teH}@A\_&&sQ%3DUsbW/m88Wk,K** (API\_KEY muss „geheim“ bleiben)
- {API\_ENDPUNKT} ist gesamter Endpunkt, z.B. /user/max.mustermann@hs-furtwangen.de/route/873
- {JSON\_OBJEKT} ist das gesamte JSON-Objekt OHNE führende und OHNE endende geschweifte Klammer mit token Bezeichner einschließlich Doppelpunkt aber ohne Value, bei GET-Requests hat das JSON-Objekt den Wert "" (Leerstring)

Beispiel:

```
"user_email": "max.mustermann@gmail.com",
"firstname": "Max",
"lastname": "Mustermann",
"mobile_number": "49776523456",
"hometown_zip": "78098",
"token":
```

- Stimmt gelieferter Token nicht mit dem auf serverseite berechneten Token überein, muss der Request immer mit dem HTTP Response 401 abgelehnt werden
- 

### **Verifizierungscode anfordern**

API-Endpoint: /verify/{user\_email} /\* z.B. max.mustermann@hs-furtwangen.de \*/

HTTP-Method: GET

Erwartete Rückgabe:

- HTTP Response Code 200 (ok) + JSON-Objekt (mit Beispielinhalt):

```
{
  "verify_code": "123456"
}
```

**Anmerkung:** Es wird ein zufälliger, sechs-stelliger Verifizierungscode generiert und an {user\_email} verschickt, mit der Aufforderung, diesen Verifizierungscode in der App einzugeben. Zusätzlich wird dieser als JSON-Objekt dem Request zurückgegeben.

---

### **App Nutzer anlegen**

API-Endpoint: /user/new  
HTTP-Method: POST

Gelieferte Daten (mit Beispieldaten):

```
{  
    "user_email": "max.mustermann@gmail.com",  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098",  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
- HTTP Response Code 409 (conflict): {user\_email} schon vorhanden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

### **App Nutzer ändern**

API-Endpoint: /user/{(old\_)user\_email} /\* z.B. max.mustermann@gmail.com \*/  
HTTP-Method: PUT

Gelieferte Daten (mit Beispieldaten):

```
{  
    "user_email": "max@mustermann.com", /* Bei keiner Änderung der E-Mail Adresse =>  
    old_user_email */  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098",  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
- HTTP Response Code 424 (failed dependency): {(old\_)user\_email} nicht vorhanden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

### **App Nutzer Daten abrufen**

API-Endpoint: /user/{user\_email}

HTTP-Method: GET

Erwartete Rückgabe:

- HTTP Response Code 200 (ok) + JSON-Objekt (mit Beispielinhalt):
- ```
{  
    "user_email": "max.mustermann@gmail.com",  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
- HTTP Response Code 424 (failed dependency): {user\_email} nicht gefunden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

#### Bei Beginn einer Routen Aufzeichnung (User klickt auf Route aufzeichnen)

API-Endpoint: /user/{user\_email}/route/new

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "route_id": "123456",  
    "timestamp": "123456789",  
    "start_point": {  
        "latitude": "37.422005",  
        "longitude": "-122.084095"  
    },  
    "end_point": "1", /* 0: User Heimatort; 1: Furtwangen; 2: Schwenningen; 3: Tuttlingen */  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete HTTP-Rückgabewerte

- HTTP Response Code 200 (ok)
- HTTP Response Code 409 (conflict): {route\_id} schon vorhanden
- HTTP Response Code 424 (failed dependency): {user\_email} nicht vorhanden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

#### Während einer Routen Aufzeichnung

API-Endpoint: /user/{user\_email}/route/{route\_id}

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "positions": [  
        {  
            "timestamp": "12569537329",  
            "lat": 37.422005,  
            "lon": -122.084095  
        }  
    ]  
}
```

```

    "latitude": "38.123456",
    "longitude": "-47.654321"
},
{
    "timestamp": "12569564253",
    "latitude": "42.123456",
    "longitude": "-66.654321"
},
    ( ..... )
],
"token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"
}

```

#### Erwartete HTTP-Rückgabewerte

- HTTP Response Code 200 (ok)
- HTTP Response Code 424 (failed dependency): {route\_id} nicht gefunden
- HTTP Response Code 401 (unauthorized): ungültiger Token

Anmerkung: Alle X-Minuten wird ein Satz von GPS-Daten an den API-Endpunkt versendet. Nach weiteren X-Minuten wird ein neuer Satz von GPS-Daten an den API-Endpunkt gesendet. Der Vorgang wiederholt sich solange, bis die Routen Aufzeichnung beendet wird (dafür wird ein entsprechender API-Endpunkt aufgerufen, siehe weiter unten)

---

#### **Nach Ende der Routen Aufzeichnung (User klickt auf Route aufzeichnen beenden)**

API-Endpoint: /user/{user\_email}/route/end

HTTP-Method: POST

Gelieferte Daten (mit Beispieldaten):

```

{
    "route_id": "123456",
    "timestamp": "123456789",
    "total_duration_seconds": "1500",
    "total_kilometers": "25",
    "obd_info": { /* Sofern OBD aktiv war, ansonsten werden die OBD Values mit null gefüllt */
    },
    "average_speed": "70",
    "average_consumption": "3"
},
"token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"
}

```

#### Erwartete HTTP-Rückgabewerte

- HTTP Response Code 200 (ok)
  - HTTP Response Code 409 (conflict): Route wurde schon beendet
  - HTTP Response Code 424 (failed dependency): {route\_id} nicht vorhanden
  - HTTP Response Code 401 (unauthorized): ungültiger Token
- 

#### **Anmerkungen**

- {user\_email} kann als Primärschlüssel verwendet werden
- {route\_id} ist nur für den jeweiligen App-Nutzer eindeutig, nicht global
- Beim Start der Route (nach Aufruf von /user/{user\_email}/route/new), beim Zufügen von Routen Daten (nach Aufruf von /user/{user\_email}/route/end) und beim Beenden der Route (nach Aufruf von /user/{user\_email}/route/end) muss die Websites-Gruppe über das Update informiert werden (Ping)! - Das kann ähnlich wie das die API-Designer bei Facebook implementiert haben, realisiert werden (siehe: <https://developers.facebook.com/docs/graph-api/real-time-updates/v2.2>). Oder einfacher als hardcodierte Ping URL. Mit dem Ping, werden die benötigten Daten per HTTP-GET als JSON mitgeliefert.

## **HFU Logistik App**

### **Spezifikation RESTful API (seitens APP-Gruppe)**

#### **Ansprechpartner**

Tobias Straub

[t.straub@hs-furtwangen.de](mailto:t.straub@hs-furtwangen.de)

Version: 2.6

---

#### **Authentifizierung**

Bei GET-Requests: Durch Anhängen des Token an den Parameter ?token=

Bei POST- und PUT-Requests: Das Token befindet sich im JSON-Objekt unter "token"

Token Berechnung: {user\_email}+{API\_KEY}+{API-ENDPUNKT}+{JSON\_OBJEKT}

#### Anmerkungen:

- {user\_email} ist beim Aufruf des API Endpunktes zur Änderung der Nutzerdaten bereits die neue E-Mail (falls geändert), ansonsten die bisherige (nicht geänderte)
- {API\_KEY} lautet: [\\_u8teH}@A\\_&&sQ%3DUusbW/m88Wk,K](mailto:_u8teH}@A_&&sQ%3DUusbW/m88Wk,K) (API\_KEY muss „geheim“ bleiben)
- {API-ENDPUNKT} ist gesamter Endpunkt, z.B. /user/max.mustermann@hs-furtwangen.de/route/873
- {JSON\_OBJEKT} ist das gesamte JSON-Objekt OHNE führende und OHNE endende geschweifte Klammer, bei GET-Requests hat das JSON-Objekt den Wert "" (Leerstring)
- Stimmt gelieferter Token nicht mit dem auf serverseite berechneten Token überein, muss der Request immer mit dem HTTP Response 401 abgelehnt werden

---

#### **Verifizierungscode anfordern**

API-Endpoint: /verify/{user\_email} /\* z.B. max.mustermann@hs-furtwangen.de \*/

HTTP-Method: GET

#### Erwartete Rückgabe:

- HTTP Response Code 200 (ok) + JSON-Objekt (mit Beispielinhalt):

```
{  
    "verify_code": "123456"  
}
```

Anmerkung: Es wird ein zufälliger, sechs-stelliger Verifizierungscode generiert und an {user\_email} verschickt, mit der Aufforderung, diesen Verifizierungscode in der App einzugeben. Zusätzlich wird dieser als JSON-Objekt dem Request zurückgegeben.

---

#### **App Nutzer anlegen**

API-Endpoint: /user/new

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_email": "max.mustermann@gmail.com",  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098",  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
- HTTP Response Code 409 (conflict): {user\_email} schon vorhanden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

### App Nutzer ändern

API-Endpoint: /user/{(old\_)user\_email} /\* z.B. max.mustermann@gmail.com \*/

HTTP-Method: PUT

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_email": "max@mustermann.com", /* Bei keiner Änderung der E-Mail Adresse =>  
    old_user_email */  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098",  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

Erwartete Rückgabe:

- HTTP Response Code 200 (ok)
- HTTP Response Code 424 (failed dependency): {(old\_)user\_email} nicht vorhanden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

### Bei Beginn einer Routen Aufzeichnung (User klickt auf Route aufzeichnen)

API-Endpoint: /user/{user\_email}/route/new

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "route_id": 123456,  
    "timestamp": 123456789,  
    "start_point": {  
        "latitude": "37.422005",  
        "longitude": "-122.084095"  
    },  
    "end_point": "1", /* 0: User Heimatort; 1: Furtwangen; 2: Schwenningen; 3: Tuttlingen */  
    "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"
```

}

**Erwartete HTTP-Rückgabewerte**

- HTTP Response Code 200 (ok)
  - HTTP Response Code 409 (conflict): {route\_id} schon vorhanden
  - HTTP Response Code 424 (failed dependency): {user\_email} nicht vorhanden
  - HTTP Response Code 401 (unauthorized): ungültiger Token
- 

**Während einer Routen Aufzeichnung**

API-Endpoint: /user/{user\_email}/route/{route\_id}

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
  "positions": [  
    {  
      "timestamp": "12569537329",  
      "latitude": "38.123456",  
      "longitude": "-47.654321"  
    },  
    {  
      "timestamp": "12569564253",  
      "latitude": "42.123456",  
      "longitude": "-66.654321"  
    },  
    ( ..... )  
  "token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"  
}
```

**Erwartete HTTP-Rückgabewerte**

- HTTP Response Code 200 (ok)
- HTTP Response Code 424 (failed dependency): {route\_id} nicht gefunden
- HTTP Response Code 401 (unauthorized): ungültiger Token

Anmerkung: Alle X-Minuten wird ein Satz von GPS-Daten an den API-Endpunkt versendet.

Nach weiteren X-Minuten wird ein neuer Satz von GPS-Daten an den API-Endpunkt gesendet. Der Vorgang wiederholt sich solange, bis die Routen Aufzeichnung beendet wird (dafür wird ein entsprechender API-Endpunkt aufgerufen, siehe weiter unten)

---

**Nach Ende der Routen Aufzeichnung (User klickt auf Route aufzeichnen beenden)**

API-Endpoint: /user/{user\_email}/route/end

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
  "route_id": 123456,  
  "timestamp": 123456789
```

```
"total_duration_seconds": 1500
"obd_info": { /* Sofern OBD aktiv war, ansonsten werden diese Daten nicht geliefert */
    "total_kilometers": 25,
    "average_speed": 70,
    "consumption": 3
},
"token": "ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad"
}
```

Erwartete HTTP-Rückgabewerte

- HTTP Response Code 200 (ok)
- HTTP Response Code 409 (conflict): Route wurde schon beendet
- HTTP Response Code 424 (failed dependency): {route\_id} nicht vorhanden
- HTTP Response Code 401 (unauthorized): ungültiger Token

---

#### Anmerkungen

- {user\_email} kann als Primärschlüssel verwendet werden
- {route\_id} ist nur für den jeweiligen App-Nutzer eindeutig, nicht global
- In diesem Dokument sind ausschließlich Endpunkte seitens der APP-Gruppe definiert. Die Websites-Gruppe kann zum Lesen der Daten allerdings die gleichen API-Endpunkte verwenden (wird in der Praxis sogar so gemacht), sofern die HTTP-Methode **NICHT** über POST läuft, sondern bspw. (am besten) über GET.

## **HFU Logistik App**

### **Spezifikation RESTful API (seitens APP-Gruppe)**

#### **Ansprechpartner**

Tobias Straub

[t.straub@hs-furtwangen.de](mailto:t.straub@hs-furtwangen.de)

Version: 1.1

---

#### **Neuer App-Nutzer**

API-Endpoint: /user

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_email": "max.mustermann@gmail.com",  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78098"  
}
```

Erwartete HTTP-Rückgabewerte

200 – ok

409 – conflict (user\_mail schon vorhanden)

401 – unauthorized (Nicht eingeloggt)

---

#### **App-Nutzer ändert seine persönlichen Daten**

API-Endpoint: /user/{(old\_)user\_mail}

HTTP-Method: PUT

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_email": "max.xyz@gmail.com", /* maybe(!) new user mail */  
    "firstname": "Max",  
    "lastname": "Xyz",  
    "mobile_number": "49776523456",  
    "hometown_zip": "78088"  
}
```

Erwartete HTTP-Rückgabewerte

200 – ok

424 – failed dependency (user\_mail nicht gefunden)

401 – unauthorized (Nicht eingeloggt)

---

#### **Bei Beginn einer Routen Aufzeichnung (User klickt auf Route aufzeichnen)**

API-Endpoint: /route/new

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_email": "max.xyz@gmail.com",  
    "route_id": 123456,  
    "timestamp": 123456789,  
    "start_point": {  
        "latitude": "37.422005",  
        "longitude": "-122.084095"  
    },  
    "end_point": "1" /* 0: User Heimatort; 1: Furtwangen; 2: Schwenningen; 3: Tuttlingen */  
}
```

Erwartete HTTP-Rückgabewerte

200 – ok

409 – conflict (route\_id schon vorhanden)

424 – failed dependency (user\_email nicht vorhanden)

401 – unauthorized (Nicht eingeloggt)

---

### **Während einer Routen Aufzeichnung**

API-Endpoint: /route/{route\_id}

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "positions": [  
        {  
            "timestamp": "12569537329",  
            "latitude": "38.123456",  
            "longitude": "-47.654321"  
        },  
        {  
            "timestamp": "12569564253",  
            "latitude": "42.123456",  
            "longitude": "-66.654321"  
        },  
        ( ..... )  
    ]  
}
```

Erwartete HTTP-Rückgabewerte

200 – ok

424 – failed dependency (route\_id nicht gefunden)

401 – unauthorized (Nicht eingeloggt)

Anmerkung: Alle X-Minuten wird ein Satz von GPS-Daten an den API-Endpunkt versendet.

Nach weiteren X-Minuten wird ein neuer Satz von GPS-Daten an den API-Endpunkt gesendet. Der Vorgang wiederholt sich solange, bis die Routen Aufzeichnung beendet wird (dafür wird ein entsprechender API-Endpunkt aufgerufen, siehe weiter unten)

---

**Nach Ende der Routen Aufzeichnung (User klickt auf Route aufzeichnen beenden)**

API-Endpoint: /route/end

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "route_id": 123456,  
    "timestamp": 123456789  
    "total_duration_seconds": 1500  
    "obd_info": { /* Sofern OBD aktiv war, ansonsten werden diese Daten nicht geliefert */  
        "total_kilometers": 25,  
        "average_speed": 70,  
        "average_consumption": 3  
    }  
}
```

Erwartete HTTP-Rückgabewerte

200 – ok

409 – conflict (Route wurde schon beendet)

424 – failed dependency (route\_id nicht vorhanden)

401 – unauthorized (Nicht eingeloggt)

---

**Anmerkungen**

- user\_mail kann als Primärschlüssel verwendet werden
- route\_id ist nur für den jeweiligen App-Nutzer eindeutig, nicht global
- In diesem Dokument sind ausschließlich Endpunkte seitens der APP-Gruppe definiert. Die Websites-Gruppe kann zum Lesen der Daten allerdings die gleichen API-Endpunkte verwenden (wird in der Praxis sogar so gemacht), sofern die HTTP-Methode **NICHT** über POST läuft, sondern bspw. (am besten) über GET.
- RESTful Authentifizierung kann über OAuth 2.0 abgewickelt werden

## **HFU Logistik App**

### **Spezifikation RESTful API (seitens APP-Gruppe)**

#### **Ansprechpartner**

Tobias Straub

[t.straub@hs-furtwangen.de](mailto:t.straub@hs-furtwangen.de)

Version: 1.0

---

#### **Neuer App-Nutzer**

API-Endpoint: /user

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_mail": "max.mustermann@gmail.com",  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": 49776523456,  
    "hometown": "Berlin"  
}
```

Erwartete HTTP-Rückgabewerte

200 – ok

409 – conflict (user\_mail schon vorhanden)

401 – unauthorized (Nicht eingeloggt)

---

#### **App-Nutzer ändert seine persönlichen Daten**

API-Endpoint: /user/{(old\_user\_mail}

HTTP-Method: PUT

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_mail": "max.xyz@gmail.com", /* maybe(!) new user mail */  
    "firstname": "Max",  
    "lastname": "Xyz",  
    "mobile_number": 49776523456,  
    "hometown": "Stuttgart"  
}
```

Erwartete HTTP-Rückgabewerte

200 – ok

424 – failed dependency (user\_mail nicht gefunden)

401 – unauthorized (Nicht eingeloggt)

---

#### **Bei Beginn einer Routen Aufzeichnung (User klickt auf Route aufzeichnen)**

API-Endpoint: /route/new

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_mail": "max.xyz@gmail.com",  
    "route_id": 123456,  
    "timestamp": 123456789,  
    "start_point": {  
        "latitude": "37.422005",  
        "longitude": "-122.084095"  
    },  
    "end_point": "1" /* 0: User Heimatort; 1: Furtwangen; 2: Schwenningen; 3: Tuttlingen */  
}
```

Erwartete HTTP-Rückgabewerte

200 – ok

409 – conflict (route\_id schon vorhanden)

424 – failed dependency (user\_mail nicht vorhanden)

401 – unauthorized (Nicht eingeloggt)

---

### Während einer Routen Aufzeichnung

API-Endpoint: /route/{route\_id}

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "positions": [  
        {  
            "timestamp": "12569537329",  
            "latitude": "38.123456",  
            "longitude": "-47.654321"  
        },  
        {  
            "timestamp": "12569564253",  
            "latitude": "42.123456",  
            "longitude": "-66.654321"  
        },  
        ( ..... )  
    ]  
}
```

Erwartete HTTP-Rückgabewerte

200 – ok

424 – failed dependency (route\_id nicht gefunden)

401 – unauthorized (Nicht eingeloggt)

Anmerkung: Alle X-Minuten wird ein Satz von GPS-Daten an den API-Endpunkt versendet.

Nach weiteren X-Minuten wird ein neuer Satz von GPS-Daten an den API-Endpunkt gesendet. Der Vorgang wiederholt sich solange, bis die Routen Aufzeichnung beendet wird (dafür wird ein entsprechender API-Endpunkt aufgerufen, siehe weiter unten)

---

#### Nach Ende der Routen Aufzeichnung (User klickt auf Route aufzeichnen beenden)

API-Endpoint: /route/end

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "route_id": 123456,  
    "timestamp": 123456789  
    "total_duration_seconds": 1500  
    "obd_info": { /* Sofern OBD aktiv war, ansonsten werden diese Daten nicht geliefert */  
        "total_kilometers": 25,  
        "average_speed": 70,  
        "average_consumption": 3  
    }  
}
```

Erwartete HTTP-Rückgabewerte

200 – ok

409 – conflict (Route wurde schon beendet)

424 – failed dependency (route\_id nicht vorhanden)

401 – unauthorized (Nicht eingeloggt)

---

#### Anmerkungen

- user\_mail kann als Primärschlüssel verwendet werden
- route\_id ist nur für den jeweiligen App-Nutzer eindeutig, nicht global
- In diesem Dokument sind ausschließlich Endpunkte seitens der APP-Gruppe definiert. Die Websites-Gruppe kann zum Lesen der Daten allerdings die gleichen API-Endpunkte verwenden (wird in der Praxis sogar so gemacht), sofern die HTTP-Methode **NICHT** über POST läuft, sondern bspw. (am besten) über GET.
- RESTful Authentifizierung kann über OAuth 2.0 laufen

## **HFU Logistik App**

### **Spezifikation RESTful API (seitens APP-Gruppe)**

#### **Ansprechpartner**

Tobias Straub

[t.straub@hs-furtwangen.de](mailto:t.straub@hs-furtwangen.de)

Version: 0.9 (vorläufig)

---

#### **Neuer App-Nutzer**

API-Endpoint: /user

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_mail": "max.mustermann@gmail.com",  
    "firstname": "Max",  
    "lastname": "Mustermann",  
    "mobile_number": 49776523456,  
    "hometown": "Berlin"  
}
```

Erwarteter HTTP-Rückgabewert: 201

---

#### **App-Nutzer ändert seine persönlichen Daten**

API-Endpoint: /user/{old\_user\_mail}

HTTP-Method: PUT

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_mail": "max.xyz@gmail.com", /* new user mail */  
    "firstname": "Max",  
    "lastname": "Xyz",  
    "mobile_number": 49776523456,  
    "hometown": "Stuttgart"  
}
```

Erwarteter HTTP-Rückgabewert: 200

---

#### **Bei Beginn einer Routen Aufzeichnung (User klickt auf Route aufzeichnen)**

API-Endpoint: /route/new

HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "user_mail": "max.xyz@gmail.com",  
    "route_id": 123456,
```

```
"timestamp": 123456789,  
"start_point": {  
    "latitude": "37.422005",  
    "longitude": "-122.084095"  
},  
"end_point": "1" /* 0: User Heimatort; 1: Furtwangen; 2: Schwenningen; 3: Tuttlingen */  
}
```

Erwarteter HTTP-Rückgabewert: 201

---

#### **Während einer Routen Aufzeichnung**

API-Endpoint: /route/{route\_id}  
HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "positions": [  
        {  
            "timestamp": "12569537329",  
            "latitude": "38.123456",  
            "longitude": "-47.654321"  
        },  
        {  
            "timestamp": "12569564253",  
            "latitude": "42.123456",  
            "longitude": "-66.654321"  
        },  
        ( ..... )  
    ]  
}
```

Erwarteter HTTP-Rückgabewert: 201

Anmerkung: Alle X-Minuten wird ein Satz von GPS-Daten an den API-Endpunkt versendet.  
Nach weiteren X-Minuten wird ein neuer Satz von GPS-Daten an den API-Endpunkt  
gesendet. Der Vorgang wiederholt sich solange, bis die Routen Aufzeichnung beendet  
wird (dafür wird ein entsprechender API-Endpunkt aufgerufen, siehe weiter unten)

---

#### **Nach Ende der Routen Aufzeichnung (User klickt auf Route aufzeichnen beenden)**

API-Endpoint: /route/end  
HTTP-Method: POST

Gelieferte Daten (mit Beispielinhalten):

```
{  
    "route_id": 123456,  
    "timestamp": 123456789  
    "total_duration_seconds": 1500  
    "obd_info": { /* Sofern OBD aktiv war, ansonsten werden diese Daten nicht geliefert */  
    "total_kilometers": 25,  
}
```

```
        "average_speed": 70,  
        "average_consumption": 3  
    }  
}
```

Erwarteter HTTP-Rückgabewert: 201

---

#### Anmerkungen

- user\_mail kann als Primärschlüssel verwendet werden
- route\_id ist nur für den jeweiligen App-Nutzer eindeutig, nicht global
- In diesem Dokument sind ausschließlich Endpunkte seitens der APP-Gruppe definiert. Die Websites-Gruppe kann zum Lesen der Daten allerdings die gleichen API-Endpunkte verwenden (wird in der Praxis sogar so gemacht), sofern die HTTP-Methode **NICHT** über POST läuft, sondern bspw. (am besten) über GET.
- RESTful Authentifizierung kann über OAuth 2.0 laufen

## **12.2 Javadoc, Quellcode und UML-Diagramme**

javadoc abrufbar unter: <http://lab.tobiasstraub.com/hfuwerdmobil/>  
sowie auf der beiliegenden DVD.

```
1: package com.HFUwerdMobil.activity;
2:
3: import java.util.ArrayList;
4: import java.util.List;
5:
6: import android.app.Activity;
7: import android.content.Context;
8: import android.content.Intent;
9: import android.os.Bundle;
10: import android.view.LayoutInflater;
11: import android.view.Menu;
12: import android.view.MenuItem;
13: import android.view.View;
14: import android.view.ViewGroup;
15: import android.widget.BaseAdapter;
16: import android.widget.GridView;
17: import android.widget.ImageView;
18: import android.widget.TextView;
19:
20: import com.HFUwerdMobil.main.R;
21:
22: /**
23:  * Diese Klasse handelt die About Activity. Die About Activity enthÃ¤lt ein
24:  * Grid mit Fotos der Verantwortlichen Team-Member der App.
25: *
26: * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
27: *         Veronika Kinzel
28: * @version 1.0
29: */
30: public class AboutActivity extends Activity {
31:     /**
32:      * Zweidimensionale GridView
33:      */
34:     GridView gvMain;
35:
36:     /**
37:      * Erzeugt die View
38:      */
39:     @Override
40:     protected void onCreate(Bundle savedInstanceState) {
41:         super.onCreate(savedInstanceState);
42:         setContentView(R.layout.activity_about);
43:
44:         gvMain = (GridView) findViewById(R.id.gvMain);
45:         gvMain.setAdapter(new ImgAdapter(this));
46:     }
47:
48:     /**
49:      * Erzeugt das OptionmenÃ¼
```

```
50:         */
51:     @Override
52:     public boolean onCreateOptionsMenu(Menu menu) {
53:         getMenuInflater().inflate(R.menu.popupmenu, menu);
54:         return true;
55:     }
56:
57:     /**
58:      * Handelt das Optionmenü
59:      */
60:     @Override
61:     public boolean onOptionsItemSelected(MenuItem item) {
62:         switch (item.getItemId()) {
63:             case R.id.menu_home:
64:                 startActivity(new Intent(this, ChooseRouteActivity.class));
65:                 return true;
66:             case R.id.menu_settings:
67:                 startActivity(new Intent(this, SettingsActivity.class));
68:                 return true;
69:             case R.id.menu_about:
70:                 startActivity(new Intent(this, AboutActivity.class));
71:                 return true;
72:             case R.id.menu_imprint:
73:                 startActivity(new Intent(this, ImprintActivity.class));
74:                 return true;
75:             case R.id.menu_help:
76:                 startActivity(new Intent(this, HelpActivity.class));
77:                 return true;
78:             default:
79:                 super.onOptionsItemSelected(item);
80:             }
81:             return false;
82:         }
83:
84:         /**
85:          * Adapterklasse, welche für die Auslieferung der Images zuständig ist
86:          *
87:          * @author Tobias Straub, Susi Roos, Thomas Lesinski, Matthias Feichtmeier,
88:          *         Veronika Kinzel, Elena Fedorow
89:          * @version 1.0
90:          *
91:         */
92:         private class ImgAdapter extends BaseAdapter {
93:
94:             private List<Item> items = new ArrayList<Item>();
95:             private LayoutInflator inflater;
96:
97:             public ImgAdapter(Context context) {
98:                 inflater = LayoutInflator.from(context);
```

```
99:  
100:        items.add(new Item(getResources().getString(R.string.tobiasstraub), R.drawable.b3));  
101:        items.add(new Item(getResources().getString(R.string.susanneroops), R.drawable.b4));  
102:        items.add(new Item(getResources().getString(R.string.thomaslesinski), R.drawable.b8));  
103:        items.add(new Item(getResources().getString(R.string.matthiasfeichtmeier), R.drawable.b6));  
104:        items.add(new Item(getResources().getString(R.string.veronikakinzel), R.drawable.b5));  
105:        items.add(new Item(getResources().getString(R.string.hfuwerdmobillogo), R.drawable.ic_launcher)  
,;  
106:    }  
107:  
108:  
109:    /**  
110:     * Gibt die Anzahl der Items in der Arrayliste zurÃ¼ck  
111:     */  
112:    @Override  
113:    public int getCount() {  
114:        return items.size();  
115:    }  
116:  
117:    /**  
118:     * Gibt das Item an einer bestimmten Position in der Arraylist zurück  
119:     *  
120:     * @param position  
121:     *          Position des Elements, welches zurückgegeben werden soll  
122:     */  
123:    @Override  
124:    public Object getItem(int position) {  
125:        return items.get(position);  
126:    }  
127:  
128:    /**  
129:     * Gibt die Bild ID des Item an einer bestimmten Position in der  
130:     * Arraylist zurück  
131:     *  
132:     * @param position  
133:     *          Position des Elements, welches die Bild Id zurückgeben  
134:     *          soll  
135:     */  
136:    @Override  
137:    public long getItemId(int position) {  
138:        return items.get(position).drawableId;  
139:    }  
140:  
141:    /**  
142:     * Gibt die View zurück  
143:     */  
144:    @Override  
145:    public View getView(int position, View convertView, ViewGroup parent) {  
146:        View v = convertView;
```

```
147:             ImageView picture;
148:             TextView name;
149:
150:             if (v == null) {
151:                 v = inflater.inflate(R.layout.item, parent, false);
152:                 v.setTag(R.id.picture, v.findViewById(R.id.picture));
153:                 v.setTag(R.id.text, v.findViewById(R.id.text));
154:             }
155:             picture = (ImageView) v.getTag(R.id.picture);
156:             name = (TextView) v.getTag(R.id.text);
157:             Item item = getItem(position);
158:             picture.setImageResource(item.drawableId);
159:             name.setText(item.name);
160:
161:             return v;
162:         }
163:
164:     }
165:
166: /**
167: * Klasse, die ein einzelnes Item definiert
168: *
169: * @author Tobias Straub, Susi Roos, Thomas Lesinski, Matthias Feichtmeier,
170: *         Veronika Kinzel, Elena Fedorow
171: * @version 1.0
172: *
173: */
174: private class Item {
175:     final String name;
176:     final int drawableId;
177:
178:     Item(String name, int drawableId) {
179:         this.name = name;
180:         this.drawableId = drawableId;
181:     }
182: }
183:
184: }
```

```
1: package com.HFUwerdMobil.activity;
2:
3: import java.util.ArrayList;
4: import java.util.List;
5: import java.util.concurrent.ExecutionException;
6:
7: import android.app.ListActivity;
8: import android.content.Intent;
9: import android.content.SharedPreferences;
10: import android.location.LocationManager;
11: import android.os.Bundle;
12: import android.provider.Settings;
13: import android.view.Menu;
14: import android.view.MenuItem;
15: import android.view.View;
16: import android.view.View.OnClickListener;
17: import android.widget.AdapterView;
18: import android.widget.AdapterView.OnItemClickListener;
19: import android.widget.ArrayAdapter;
20: import android.widget.Button;
21: import android.widget.ListView;
22: import android.widget.Spinner;
23: import android.widget.Toast;
24:
25: import com.HFUwerdMobil.main.R;
26: import com.HFUwerdMobil.memory.DataAccessHandler;
27: import com.HFUwerdMobil.memory.EntityRoute;
28: import com.HFUwerdMobil.util.Geocoding;
29:
30: /**
31:  * Diese Klasse handelt die Auswahl eines Routenziels und listet die bereits
32:  * abgeschlossenen Routen auf.
33:  *
34:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
35:  *         Veronika Kinzel
36:  * @version 1.0
37:  */
38: public class ChooseRouteActivity extends ListActivity {
39:     /**
40:      * Deklariert die Liste, mit den bereits abgeschlossenen Routen
41:      */
42:     ArrayList<String> list = new ArrayList<String>();
43:
44:     /**
45:      * Adapter fÃ¼r die Liste
46:      */
47:     ArrayAdapter<String> adapter;
48:
49:     /** {@link com.HFUwerdMobil.memory.DataAccessHandler} Objekt **/
```

```
50:         DataAccessHandler sHandler = new DataAccessHandler(this);
51:
52:        /**
53:         * Erzeugt die Activity und definiert einen Listener fÃ¼r den Button um die
54:         * ausgewÃ¤hlte Route zu starten. Definiert zu den auswÃ¤hlbaren Zielen die
55:         * Destination Codes und gibt beim Klick auf den Button den korrekten
56:         * Destination Code an den {@link com.HFUwerdMobil.service.GPSListener}
57:         * Service weiter. PrÃ¤ft zusÃ¤tzlich den GPS Status.
58:         */
59:        @Override
60:        public void onCreate(Bundle savedInstanceState) {
61:            super.onCreate(savedInstanceState);
62:            setContentView(R.layout.activity_choose_route);
63:            sHandler.openDB();
64:
65:            listRoutes(5);
66:
67:            Button startRoute = (Button) findViewById(R.id.start_route);
68:            startRoute.setOnClickListener(new OnClickListener() {
69:                @Override
70:                public void onClick(View v) {
71:                    Intent i = new Intent(getApplicationContext(), com.HFUwerdMobil.service.GPSListener.class);
ss);
72:                    Spinner spinner = (Spinner) findViewById(R.id.locations);
73:                    if (spinner.getSelectedItem().toString().equals(getString(R.string.furtwangen))) {
74:                        i.putExtra("selected_destination", 1);
75:                    }
76:                    if (spinner.getSelectedItem().toString().equals(getString(R.string.schwenningen))) {
77:                        i.putExtra("selected_destination", 2);
78:                    }
79:                    if (spinner.getSelectedItem().toString().equals(getString(R.string.tuttlingen))) {
80:                        i.putExtra("selected_destination", 3);
81:                    }
82:                    if (spinner.getSelectedItem().toString().equals(getString(R.string.home))) {
83:                        i.putExtra("selected_destination", 0);
84:                    }
85:                    startService(i);
86:                    startActivityForResult(new Intent(getApplicationContext(), RecordRouteActivity.class));
87:                }
88:            });
89:
90:            checkGps();
91:
92:            sHandler.closeDB();
93:        }
94:
95:        /**
96:         * Listet die bereits abgeschlossenen Routen in einer Liste auf. Kodiert
97:         * dazu die Koordinaten und den Destination Code zu Ortsnamen. Setzt den
```

```
98:             * Elementen auÃerdem Links zu der
99:             * {@link com.HFUwerdMobil.activity.RouteDetailsActivity}, um detaillierte
100:            * Informationen zu der bereits abgeschlossenen Route einzusehen.
101:            *
102:            * @param max
103:            *          Anzahl der maximal auszugebenden Listenelemente
104:            */
105:        private void listRoutes(int max) {
106:            List<EntityRoute> routes = sHandler.getAllRoutes();
107:            adapter = new ArrayAdapter<String>(this, R.layout.recent_routes, list);
108:            if (routes != null) {
109:                if (routes.size() < max) {
110:                    max = routes.size() - 1;
111:                }
112:                for (int i = 0; i <= max; i++) {
113:                    try {
114:                        String start = new Geocoding(routes.get(i).getStartLat(), routes.get(i).getStar
tLon()).execute().get();
115:                        if (start == null) {
116:                            start = "";
117:                        }
118:                        String dest = "";
119:                        switch (routes.get(i).getEndPoint()) {
120:                            case 0:
121:                                dest = getResources().getString(R.string.home);
122:                                break;
123:                            case 1:
124:                                dest = getResources().getString(R.string.furtwangen);
125:                                break;
126:                            case 2:
127:                                dest = getResources().getString(R.string.schwenningen);
128:                                break;
129:                            case 3:
130:                                dest = getResources().getString(R.string.tuttlingen);
131:                                break;
132:                            }
133:                            adapter.add(start + "\n" + dest);
134:                        } catch (InterruptedException e) {
135:                            e.printStackTrace();
136:                        } catch (ExecutionException e) {
137:                            e.printStackTrace();
138:                        }
139:                    }
140:                } else {
141:                    adapter.add(getResources().getString(R.string.no_routes_to_show));
142:                }
143:                setListAdapter(adapter);
144:
145:                ListView lv = getListView();
```

```
146:             lv.setTextFilterEnabled(true);
147:
148:             lv.setOnItemClickListener(new OnItemClickListener() {
149:                 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
150:                     sHandler.openDB();
151:                     Sharedpreferences userData = getApplicationContext().getSharedPreferences("userData", MODE_PRIVATE);
152:                     Sharedpreferences.Editor editor = userData.edit();
153:                     editor.putString("route_id", sHandler.getRoute((int) id + 1).getRouteId());
154:                     editor.commit();
155:                     startActivity(new Intent(getApplicationContext(), RouteDetailsActivity.class));
156:                     sHandler.closeDB();
157:                 }
158:             });
159:         }
160:
161:         /**
162:          * Prüft, ob GPS aktiviert ist. Sollte GPS deaktiviert sein, wird der Nutzer
163:          * zur entsprechenden Settings Activity weitergeleitet, auf der er GPS
164:          * aktivieren kann.
165:          */
166:         private void checkGps() {
167:             if (!((LocationManager) getSystemService(LOCATION_SERVICE)).isProviderEnabled(LocationManager.GPS_PROVIDER)) {
168:                 Toast.makeText(getApplicationContext(), R.string.activate_gps, Toast.LENGTH_LONG).show();
169:                 startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
170:             }
171:         }
172:
173:         /**
174:          * Erzeugt das Optionenmenü
175:          */
176:         @Override
177:         public boolean onCreateOptionsMenu(Menu menu) {
178:             getMenuInflater().inflate(R.menu.popupmenu, menu);
179:             return true;
180:         }
181:
182:         /**
183:          * Handelt das Optionenmenü
184:          */
185:         @Override
186:         public boolean onOptionsItemSelected(MenuItem item) {
187:             switch (item.getItemId()) {
188:                 case R.id.menu_home:
189:                     startActivity(new Intent(this, ChooseRouteActivity.class));
190:                     return true;
191:                 case R.id.menu_settings:
192:                     startActivity(new Intent(this, SettingsActivity.class));
```

```
193:             return true;
194:         case R.id.menu_about:
195:             startActivity(new Intent(this, AboutActivity.class));
196:             return true;
197:         case R.id.menu_imprint:
198:             startActivity(new Intent(this, ImprintActivity.class));
199:             return true;
200:         case R.id.menu_help:
201:             startActivity(new Intent(this, HelpActivity.class));
202:             return true;
203:         default:
204:             super.onOptionsItemSelected(item);
205:         }
206:     }
207: }
208: }
```



```
1: package com.HFUwerdMobil.activity;
2:
3: import android.app.Activity;
4: import android.content.Intent;
5: import android.os.Bundle;
6: import android.view.Menu;
7: import android.view.MenuItem;
8: import android.webkit.WebView;
9:
10: import com.HFUwerdMobil.main.R;
11:
12: /**
13:  * Diese Klasse handelt die Help Activity. Die Help Activity enthÃ¤lt
14:  * Hilfsinformationen zur Nutzung der App.
15: *
16: * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
17: *         Veronika Kinzel
18: * @version 1.0
19: */
20: public class HelpActivity extends Activity {
21:     /**
22:      * Erzeugt die Activity als WebView
23:      */
24:     @Override
25:     public void onCreate(Bundle savedInstanceState) {
26:         super.onCreate(savedInstanceState);
27:         setContentView(R.layout.activity_help);
28:         WebView helpContent = (WebView) findViewById(R.id.help_content);
29:         String text = "<html><body>" + "<p align=\"justify\">" + getString(R.string.help_content) + "</p>" + "<
/html>";
30:         helpContent.loadData(text, "text/html", "utf-8");
31:     }
32:
33:     /**
34:      * Erzeugt das OptionmenÃ¼
35:      */
36:     @Override
37:     public boolean onCreateOptionsMenu(Menu menu) {
38:         getMenuInflater().inflate(R.menu.popupmenu, menu);
39:         return true;
40:     }
41:
42:     /**
43:      * Handelt das OptionmenÃ¼
44:      */
45:     @Override
46:     public boolean onOptionsItemSelected(MenuItem item) {
47:         switch (item.getItemId()) {
48:             case R.id.menu_home:
```

```
./com/HFUwerdMobil/activity/HelpActivity.java      Tue Jan 20 22:33:17 2015      2
49:                     startActivity(new Intent(this, ChooseRouteActivity.class));
50:                     return true;
51:             case R.id.menu_settings:
52:                 startActivity(new Intent(this, SettingsActivity.class));
53:                 return true;
54:             case R.id.menu_about:
55:                 startActivity(new Intent(this, AboutActivity.class));
56:                 return true;
57:             case R.id.menu_imprint:
58:                 startActivity(new Intent(this, ImprintActivity.class));
59:                 return true;
60:             case R.id.menu_help:
61:                 startActivity(new Intent(this, HelpActivity.class));
62:                 return true;
63:             default:
64:                 super.onOptionsItemSelected(item);
65:             }
66:         return false;
67:     }
68: }
```

```
1: package com.HFUwerdMobil.activity;
2:
3: import android.app.Activity;
4: import android.content.Intent;
5: import android.os.Bundle;
6: import android.view.Menu;
7: import android.view.MenuItem;
8: import android.webkit.WebView;
9:
10: import com.HFUwerdMobil.main.R;
11:
12: /**
13:  * Diese Klasse handelt die Imprint Activity. Die Imprint Activity enthält das
14:  * Impressum der App.
15: *
16: * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
17: *         Veronika Kinzel
18: * @version 1.0
19: */
20: public class ImprintActivity extends Activity {
21:     /**
22:      * Erzeugt die Activity als WebView
23:      */
24:     @Override
25:     public void onCreate(Bundle savedInstanceState) {
26:         super.onCreate(savedInstanceState);
27:         setContentView(R.layout.activity_imprint);
28:         WebView imprintContent = (WebView) findViewById(R.id.imprint_content);
29:         String text = "<html><body>" + "<p align=\"justify\">" + getString(R.string.imprint_content) + "</p>" +
"</html>";
30:         imprintContent.loadData(text, "text/html", "utf-8");
31:     }
32:
33:     /**
34:      * Erzeugt das Optionmenü
35:      */
36:     @Override
37:     public boolean onCreateOptionsMenu(Menu menu) {
38:         getMenuInflater().inflate(R.menu.popupmenu, menu);
39:         return true;
40:     }
41:
42:     /**
43:      * Handelt das Optionmenü
44:      */
45:     @Override
46:     public boolean onOptionsItemSelected(MenuItem item) {
47:         switch (item.getItemId()) {
48:             case R.id.menu_home:
```

```
./com/HFUwerdMobil/activity/ImprintActivity.java      Tue Jan 20 22:33:22 2015      2
49:                     startActivity(new Intent(this, ChooseRouteActivity.class));
50:                     return true;
51:             case R.id.menu_settings:
52:                 startActivity(new Intent(this, SettingsActivity.class));
53:                 return true;
54:             case R.id.menu_about:
55:                 startActivity(new Intent(this, AboutActivity.class));
56:                 return true;
57:             case R.id.menu_imprint:
58:                 startActivity(new Intent(this, ImprintActivity.class));
59:                 return true;
60:             case R.id.menu_help:
61:                 startActivity(new Intent(this, HelpActivity.class));
62:                 return true;
63:             default:
64:                 super.onOptionsItemSelected(item);
65:             }
66:         return false;
67:     }
68: }
```

```
1: package com.HFUwerdMobil.activity;
2:
3: import java.util.regex.Pattern;
4:
5: import android.app.Activity;
6: import android.content.Context;
7: import android.content.Intent;
8: import android.content.SharedPreferences;
9: import android.os.Bundle;
10: import android.util.Patterns;
11: import android.view.View;
12: import android.view.View.OnClickListener;
13: import android.widget.EditText;
14: import android.widget.TextView;
15:
16: import com.HFUwerdMobil.main.R;
17: import com.HFUwerdMobil.memory.UserData;
18: import com.HFUwerdMobil.sync.DataSyncRoute;
19: import com.HFUwerdMobil.sync.DataSyncUser;
20: import com.HFUwerdMobil.util.IssueNotification;
21:
22: /**
23:  * Diese Klasse handelt die Login Activity, die bei jedem Start der App
24:  * aufgerufen wird.
25: *
26: * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
27: *         Veronika Kinzel
28: * @version 1.0
29: */
30: public class LoginActivity extends Activity {
31:     /**
32:      * {@link com.HFUwerdMobil.memory.UserData} Objekt
33:      */
34:     private UserData userData;
35:
36:     /**
37:      * Context Objekt, welches vom Syncronisationsthread benÃ¶tigt wird, um die
38:      * Systemservices nutzen zu kÃ¶nnen
39:      */
40:     private static Context context;
41:
42:     /** Speicher Objekt fÃ¼r Setting Flags fÃ¼r die Datensynchronisation **/
43:     private SharedPreferences syncData;
44:
45:     /**
46:      * Thread fÃ¼r die Synchronisation der Userdaten zu den externen Services
47:      */
48:     public Thread userDataSync;
49:
```

```
50:         /**
51:          * Thread fÃ¼r die Synchronisation der Routendaten
52:          */
53:         Thread routeThread;
54:
55:         /**
56:          * Erzeugt die Activity. FÃ¶hrt ggf. noch ausstehende Synchronisationen aus.
57:          * PrÃ¼ft ob es sich um den ersten oder nicht ersten Login handelt und fÃ¶hrt
58:          * jeweils weitere Aktionen durch.
59:          */
60:         @Override
61:         protected void onCreate(Bundle savedInstanceState) {
62:             super.onCreate(savedInstanceState);
63:             setContentView(R.layout.activity_login);
64:             userData = new UserData(this);
65:
66:             context = this;
67:             syncData = this.getSharedPreferences("syncData", MODE_PRIVATE);
68:
69:             if (userData.isHfuMember()) {
70:                 checkSync();
71:             }
72:
73:             if (userData.getUserMail() != null) {
74:                 notFirstLogin();
75:             } else {
76:                 findViewById(R.id.btn_hfu_student).setOnClickListener(new OnClickListener() {
77:                     @Override
78:                     public void onClick(View v) {
79:                         isHfuStudent(v);
80:                     }
81:                 });
82:
83:                 findViewById(R.id.btn_no_hfu_student).setOnClickListener(new OnClickListener() {
84:                     @Override
85:                     public void onClick(View v) {
86:                         isNotHfuStudent(v);
87:                     }
88:                 });
89:             }
90:         }
91:
92:         /**
93:          * PrÃ¼ft alle Synchronisationsflags, um ggf. noch ausstehende
94:          * Synchronisationen durchzufÃ¼hren, die beim letzten Durchlauf der App nicht
95:          * abgeschlossen werden konnten, z.B. durch Beenden der App.
96:          */
97:         private void checkSync() {
98:             if (syncData.getInt("user", -1) != 1) {
```

```
./com/HFUwerdMobil/activity/LoginActivity.java      Wed Jan 21 12:39:30 2015      3

99:                     userDataSync = new Thread(new DataSyncUser(context, userData.getJson()));
100:                    userDataSync.start();
101:                }
102:                if (syncData.getInt("route", -1) != 1) {
103:                    routeThread = new Thread(new DataSyncRoute(getApplicationContext()));
104:                    // routeThread.start();
105:                    // Anmerkung: Bitte erst entkommentieren, wenn die Backend-Gruppe,
106:                    // die RESTful API vollstndig funktionstchtig hat und der GPS-JSON
107:                    // eingebaut wurde.
108:                }
109:            }
110:
111:        /**
112:         * Blendet alle UI-Elemente aus, die fr Nutzer die bereits "registriert"
113:         * sind, irrelevant sind (Registrierungsbuttons). Prft ob sich die E-Mail
114:         * Adresse des Google Kontos des Nutzers gendert hat, und fr ggf. weitere
115:         * Aktionen durch (
116:         * {@link com.HFUwerdMobil.memory.UserData#checkMailChange()}, sofern der
117:         * Nutzer nicht HFU Student ist (Prinzip: HFU Nutzer haben mehr Rechte, ein
118:         * Downgrade auf eine E-Mail Adresse, die nicht der HFU angehrt, soll
119:         * verhindert werden). Gibt einen Text aus, der den User Willkommen zurck
120:         * heit und leitet ihn direkt zur
121:         * {@link com.HFUwerdMobil.activity.ChooseRouteActivity} weiter.
122:         */
123:        private void notFirstLogin() {
124:            findViewById(R.id.btn_no_hfu_student).setVisibility(View.GONE);
125:            findViewById(R.id.btn_hfu_student).setVisibility(View.GONE);
126:            TextView welcomeBack = (TextView) findViewById(R.id.text_welcome_back);
127:            welcomeBack.append(userData.getUserMail() + ".\nSchn dich wieder hier zu sehen.");
128:            findViewById(R.id.text_welcome_back).setVisibility(View.VISIBLE);
129:            if (!userData.isHfuMember()) {
130:                userData.checkMailChange();
131:            }
132:            startActivity(new Intent(getApplicationContext(), ChooseRouteActivity.class));
133:        }
134:
135:        /**
136:         * Blendet den Registrierungsbutton aus und blendet ein Eingabefeld ein, bei
137:         * dem der Nutzer aufgefordert wird seine HFU E-Mail Adresse einzugeben, um
138:         * die Registrierung abzuschlieen. Dabei wird "@hs-furtwangen.de" fest
139:         * codiert. Die E-Mail Adresse wird auf Syntax-Richtigkeit geprft. Sofern
140:         * die E-Mail valide ist, wird er zur
141:         * {@link com.HFUwerdMobil.activity.UserVerifyActivity} weitergeleitet.
142:         *
143:         * @param v
144:         *          UI-Element, welches dieses Ereignis ausgelsst hat
145:         */
146:        private void isHfuStudent(View v) {
147:            v.setVisibility(View.GONE);
```

```
148:         findViewById(R.id.btn_no_hfu_student).setVisibility(View.GONE);
149:         findViewById(R.id.text_email).setVisibility(View.VISIBLE);
150:         findViewById(R.id.text_email_hfu_string).setVisibility(View.VISIBLE);
151:         findViewById(R.id.input_email).setVisibility(View.VISIBLE);
152:         findViewById(R.id.btn_parse_mail).setVisibility(View.VISIBLE);
153:         findViewById(R.id.btn_parse_mail).setOnClickListener(new OnClickListener() {
154:             @Override
155:             public void onClick(View v) {
156:                 EditText unverifiedEmail = (EditText) findViewById(R.id.input_email);
157:                 String uet = unverifiedEmail.getText().toString();
158:                 if (uet.matches("[A-Za-z]+([A-Za-z]+)?([A-Za-z]+)?([A-Za-z]+)?")) {
159:                     Sharedpreferences userData = LoginActivity.this.getSharedPreferences("userData"
, MODE_PRIVATE);
160:                     Sharedpreferences.Editor editor = userData.edit();
161:                     editor.putBoolean("user_hfu_student", true);
162:                     editor.commit();
163:                     startActivity(new Intent(getApplicationContext(), UserVerifyActivity.class).put
Extra("unverified_email", uet
164:                           + "@hs-furtwangen.de"));
165:                 } else {
166:                     IssueNotification.emailNotValid(LoginActivity.this);
167:                 }
168:             }
169:         });
170:     }
171:
172:     /**
173:      * PrÃ¤ft ob die Google E-Mail Adresse ausgelesen werden kann und setzt diese
174:      * als Registrierungsmail (es wird kein Verifizierungsverfahren
175:      * durchgefÃ¤hrt, da die Google E-Mail grundsÃ¤tzlich schon von Google
176:      * validiert wurde). Ist die Google E-Mail Adresse nicht auslesbar (weil das
177:      * Konto bspw. nicht existiert), werden passende UI-Elemente eingeblendet,
178:      * die den Nutzer auffordern, eine E-Mail Adresse zu hinterlegen. Sollte
179:      * diese valide sein, wird der Nutzer zur
180:      * {@link com.HFUwerdMobil.activity.UserVerifyActivity} weitergeleitet.
181:      *
182:      * @param v
183:      *           UI-Element, welches dieses Ereignis ausgelÃ¶st hat
184:      */
185:     private void isNotHfuStudent(View v) {
186:         String deviceMail = userData.getDeviceMail();
187:         if (deviceMail != null) {
188:             userData.setUserMail(deviceMail);
189:             startActivity(new Intent(getApplicationContext(), SettingsActivity.class));
190:         } else {
191:             v.setVisibility(View.GONE);
192:             findViewById(R.id.btn_hfu_student).setVisibility(View.GONE);
193:             findViewById(R.id.text_email).setVisibility(View.VISIBLE);
194:             findViewById(R.id.input_email).setVisibility(View.VISIBLE);
```

```
195:                     findViewById(R.id.btn_parse_email).setVisibility(View.VISIBLE);
196:                     findViewById(R.id.btn_parse_email).setOnClickListener(new OnClickListener() {
197:                         @Override
198:                         public void onClick(View v) {
199:                             Pattern emailRegex = Patterns.EMAIL_ADDRESS;
200:                             EditText unverifiedEmail = (EditText) findViewById(R.id.input_email);
201:                             String uet = unverifiedEmail.getText().toString();
202:                             if (emailRegex.matcher(uet).matches()) {
203:                                 startActivity(new Intent(getApplicationContext(), UserVerifyActivity.class));
204:                             } else {
205:                                 IssueNotification.emailNotValid(LoginActivity.this);
206:                             }
207:                         }
208:                     });
209:                 }
210:             }
211:
212:             /**
213:              * Gibt das aktuelle Context Objekt zurÃ¼ck
214:              *
215:              * @return Context Objekt
216:              */
217:             public static Context getLoginActivityContext() {
218:                 return context;
219:             }
220:
221:             /**
222:              * Sobald die Activity in den Hintergrund rutscht, wird der mÃ¶glicherweise
223:              * derzeit ausgefÃ¼hrte Nutzer-Daten-Synchronisationsthread und der
224:              * Routen-Daten-Synchronisationsthread, interrupted
225:              */
226:             public void onPause() {
227:                 super.onPause();
228:                 if (userDataSync != null)
229:                     userDataSync.interrupt();
230:                 if (routeThread != null)
231:                     routeThread.interrupt();
232:             }
233: }
```



```
1: package com.HFUwerdMobil.activity;
2:
3: import android.app.Activity;
4: import android.content.Context;
5: import android.content.Intent;
6: import android.location.LocationManager;
7: import android.os.Bundle;
8: import android.provider.Settings;
9: import android.view.View;
10: import android.view.View.OnClickListener;
11: import android.widget.Button;
12: import android.widget.Toast;
13:
14: import com.HFUwerdMobil.main.R;
15: import com.HFUwerdMobil.memory.UserData;
16: import com.HFUwerdMobil.obd2.BluetoothActivity;
17: import com.HFUwerdMobil.obd2.OBDservice;
18:
19: /**
20:  * Diese Klasse handelt die Route aufzeichnen Activity.
21: *
22: * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
23: *         Veronika Kinzel
24: * @version 1.0
25: */
26: public class RecordRouteActivity extends Activity {
27:     /**
28:      * Status des OBD Services
29:      */
30:     private static boolean obdState = false;
31:
32:     /**
33:      * Context Objekt, welches vom OBD2 Service benÃ¤tigt wird, um Daten in die
34:      * Datenbank zu schreiben
35:      */
36:     private static Context context;
37:
38:     /**
39:      * Erzeugt die Activity. PrÃ¼ft den GPS Status und startet, sofern der Nutzer
40:      * den OBD in den Settings aktiviert hat, die
41:      * {@link com.HFUwerdMobil.obd2.BluetoothActivity}. Stoppt beim Klick auf
42:      * den Beenden Button die Services, nimmt die Activity vom Stack und startet
43:      * die {@link com.HFUwerdMobil.activity.RouteDetailsActivity}
44:      */
45:     @Override
46:     public void onCreate(Bundle savedInstanceState) {
47:         super.onCreate(savedInstanceState);
48:         setContentView(R.layout.activity_record_route);
49:
```

```
50:             context = this;
51:
52:             checkGps();
53:
54:             if (new UserData(this).getUsesObd()) {
55:                 if (!obdState) {
56:                     obdOn();
57:                     this.startActivityForResult(new Intent(this, BluetoothActivity.class), 0);
58:                 } else {
59:                     obdOff();
60:                     stopService(new Intent(this, OBDservice.class));
61:                 }
62:             }
63:
64:             Button endRoute = (Button) findViewById(R.id.endRoute);
65:             endRoute.setOnClickListener(new OnClickListener() {
66:                 @Override
67:                 public void onClick(View v) {
68:                     startActivity(new Intent(getApplicationContext(), RouteDetailsActivity.class));
69:                     finish();
70:                     stopService(new Intent(RecordRouteActivity.this, com.HFUwerdMobil.service.GPSListener.class));
71:                     stopService(new Intent(RecordRouteActivity.this, com.HFUwerdMobil.obd2.OBDservice.class));
72:                 }
73:             });
74:         }
75:
76:         /**
77:          * Handelt den Result der {@link com.HFUwerdMobil.obd2.BluetoothActivity}
78:          */
79:         @Override
80:         protected void onActivityResult(int requestCode, int resultCode, Intent data) {
81:             if (resultCode == RESULT_OK && !data.equals(null)) {
82:                 if (!data.getExtras().getBoolean("ison")) {
83:                     obdOff();
84:                 }
85:             } else {
86:                 obdOff();
87:             }
88:         }
89:
90:         /**
91:          * Stellt den OBD Status auf on
92:          */
93:         private void obdOn() {
94:             obdState = true;
95:         }
96:
```

```
97:         /**
98:          * Stellt den OBD Status auf off
99:          */
100:         private void obdOff() {
101:             obdState = false;
102:         }
103:
104:         /**
105:          * PrÃ¼ft, ob GPS aktiviert ist. Sollte GPS deaktiviert sein, wird der Nutzer
106:          * zur entsprechenden Settings Activity weitergeleitet, auf der er GPS
107:          * aktivieren kann.
108:          */
109:         private void checkGps() {
110:             if (!((LocationManager) getSystemService(LOCATION_SERVICE)).isProviderEnabled(LocationManager.GPS_PROVIDER)) {
111:                 Toast.makeText(getApplicationContext(), R.string.activate_gps, Toast.LENGTH_LONG).show();
112:                 startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
113:             }
114:         }
115:
116:         /**
117:          * Gibt das aktuelle Context Objekt zurÃ¼ck
118:          *
119:          * @return Context Objekt
120:          */
121:         public static Context getRouteRecordContext() {
122:             return context;
123:         }
124:     }
```



```
1: package com.HFUwerdMobil.activity;
2:
3: import java.text.SimpleDateFormat;
4: import java.util.ArrayList;
5: import java.util.List;
6: import java.util.concurrent.ExecutionException;
7:
8: import android.app.Activity;
9: import android.content.Intent;
10: import android.content.SharedPreferences;
11: import android.graphics.Color;
12: import android.os.Bundle;
13: import android.widget.TextView;
14: import android.widget.Toast;
15:
16: import com.HFUwerdMobil.main.R;
17: import com.HFUwerdMobil.memory.DataAccessHandler;
18: import com.HFUwerdMobil.memory.EntityGPS;
19: import com.HFUwerdMobil.util.Geocoding;
20: import com.HFUwerdMobil.util.IssueNotification;
21: import com.google.android.gms.maps.CameraUpdateFactory;
22: import com.google.android.gms.maps.GoogleMap;
23: import com.google.android.gms.maps.MapFragment;
24: import com.google.android.gms.maps.model.CameraPosition;
25: import com.google.android.gms.maps.model.LatLng;
26: import com.google.android.gms.maps.model.MarkerOptions;
27: import com.google.android.gms.maps.model.PolylineOptions;
28:
29: /**
30:  * Diese Klasse handelt die Routen Details Activity, welche alle Details zu
31:  * einer bereits gefahrenen Route visualisiert.
32:  *
33:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
34:  *         Veronika Kinzel
35:  * @version 1.0
36:  */
37: public class RouteDetailsActivity extends Activity {
38:     /**
39:      * DataAccessHandler Objekt
40:      */
41:     private DataAccessHandler sHandler;
42:
43:     /**
44:      * Externe Route ID
45:      */
46:     private String routeId;
47:
48:     /**
49:      * Liste mit den GPS Daten zu einer Route
```

```
50:         */
51:         List<EntityGPS> gpsData;
52:
53:         /**
54:          * Erzeugt die Activity. PrÃ¤ft die Intents und gibt entsprechende Meldungen
55:          * aus, wenn der User das Ziel gerade erreicht hat oder bereits seit Beginn
56:          * am Ziel war. PrÃ¤ft ob eine Route gestartet wurde und Initialwerte vom GPS
57:          * Provider erhalten hat. Und ob erste Wegpunkte erfasst wurden. Wenn das
58:          * der Fall ist, werden die Routendaten ausgegeben, und die Google Maps
59:          * Karte ausgegeben. Wenn das nicht der Fall ist, wird sofern die Route mit
60:          * Initialwerten besteht, aus der Datenbank gelÃ¶scht (da sich das Auto nicht
61:          * vom Ort bewegt hat und deshalb eine Route auch nicht existent ist), und
62:          * der User wird wieder zur
63:          * {@link com.HFUwerdMobil.activity.ChooseRouteActivity} weitergeleitet.
64:         */
65:         @Override
66:         protected void onCreate(Bundle savedInstanceState) {
67:             super.onCreate(savedInstanceState);
68:             setContentView(R.layout.activity_route_details);
69:             SharedPreferences userData = getApplicationContext().getSharedPreferences("userData", MODE_PRIVATE);
70:
71:             if (getIntent().getExtras() != null) {
72:                 if (getIntent().getExtras().getBoolean("is_destination_yet") != true) {
73:                     Toast.makeText(this, getResources().getString(R.string.is_destination_yet), Toast.LENGTH_LONG).show();
74:                 }
75:
76:                 if (getIntent().getExtras().getBoolean("is_destination") == true) {
77:                     Toast.makeText(this, getResources().getString(R.string.is_destination), Toast.LENGTH_LONG).show();
78:                 }
79:             }
80:
81:             if (userData.getString("route_id", null) != null) {
82:                 sHandler = new DataAccessHandler(this);
83:                 sHandler.openDB();
84:                 routeId = userData.getString("route_id", null);
85:                 SharedPreferences.Editor editor = userData.edit();
86:                 editor.putString("route_id", null);
87:                 editor.commit();
88:
89:                 sHandler.getRouteGps(routeId);
90:                 gpsData = sHandler.getRouteGps(routeId);
91:
92:                 if (gpsData != null) {
93:                     ((TextView) findViewById(R.id.start_destination)).setText(printStartDestination());
94:
95:                     String timeDate = new SimpleDateFormat("dd.MM.yyyy HH:mm").format(sHandler.getRoute(routeId).getTimestamp() * 1000L);
```

```
96:                         ((TextView) findViewById(R.id.route_details_date)).setText(timeDate);
97:
98:                         drawMap();
99:                     } else {
100:                         Toast.makeText(getApplicationContext(), getResources().getString(R.string.no_data_avail-
able), Toast.LENGTH_SHORT).show();
101:                         startActivity(new Intent(getApplicationContext(), ChooseRouteActivity.class));
102:                         finish();
103:                         // Es macht Sinn, nachdem keine GPS-Daten erfasst wurden, die
104:                         // Route hier wieder zu lÃ¶schen. AuÃerdem es soll fÃ¼r statistische
105:                         // Zwecke verwendet werden.
106:                     }
107:                     sHandler.closeDB();
108:                 } else {
109:                     Toast.makeText(getApplicationContext(), getResources().getString(R.string.no_data_available), T-
oast.LENGTH_SHORT).show();
110:                     startActivity(new Intent(getApplicationContext(), ChooseRouteActivity.class));
111:                     finish();
112:                 }
113:             }
114:
115:             /**
116:              * Kodiert aus den Koordinaten und dem Destination Flag einen Ort
117:              *
118:              * @return Ortskodierter Start- und Zielort als String im Format
119:              *         "{startort} - {zielort}"
120:             */
121:             private String printStartDestination() {
122:                 String startPointString = "";
123:                 try {
124:                     startPointString = new Geocoding(sHandler.getRoute(routeId).getStartLat(), sHandler.getRoute(ro-
uteId).getStartLon()).execute().get();
125:                 } catch (InterruptedException e) {
126:                     IssueNotification.fatalError(this);
127:                 } catch (ExecutionException e) {
128:                     IssueNotification.fatalError(this);
129:                 }
130:
131:                 String endPointString = "";
132:                 switch (sHandler.getRoute(routeId).getEndPoint()) {
133:                     case 0:
134:                         endPointString = getResources().getString(R.string.home);
135:                         break;
136:                     case 1:
137:                         endPointString = getResources().getString(R.string.furtwangen);
138:                         break;
139:                     case 2:
140:                         endPointString = getResources().getString(R.string.schwenningen);
141:                         break;
```

```
142:             case 3:
143:                 endPointString = getResources().getString(R.string.tuttlingen);
144:                 break;
145:             }
146:             return startPointString + " - " + endPointString;
147:         }
148:
149:         /**
150:          * Zeichnet die erfassten GPS-Daten zur Route in die Google Maps Karte und
151:          * gibt die OBD-Daten aus (sofern erfasst)
152:          */
153:     private void drawMap() {
154:         GoogleMap map = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMap();
155:         ArrayList<EntityGPS> markerList = new ArrayList<EntityGPS>();
156:         markerList.addAll(gpsData);
157:         PolylineOptions line = new PolylineOptions();
158:         line.width(5);
159:         line.color(Color.RED);
160:
161:         for (EntityGPS gps : markerList) {
162:             line.add(new LatLng(Double.parseDouble(gps.getLatitude()), Double.parseDouble(gps.getLongitude())));
163:         }
164:         map.addPolyline(line);
165:         EntityGPS gps = markerList.get(0);
166:         map.addMarker(new MarkerOptions().position(new LatLng(Double.parseDouble(gps.getLatitude()), Double.parseDouble(gps.getLongitude()))));
167:         gps = markerList.get(markerList.size() - 1);
168:         map.addMarker(new MarkerOptions().position(new LatLng(Double.parseDouble(gps.getLatitude()), Double.parseDouble(gps.getLongitude()))));
169:
170:         CameraPosition cameraPosition = new CameraPosition.Builder()
171:             .target(new LatLng(Double.parseDouble(gps.getLatitude()), Double.parseDouble(gps.getLongitude())))
172:             .zoom(12).tilt(40).build();
173:         map.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition));
174:
175:         String averageSpeed = sHandler.calculateAverageSpeed(routeId);
176:         if (averageSpeed != null) {
177:             ((TextView) findViewById(R.id.average_speed)).setText(" " + averageSpeed + " km/h");
178:         }
179:
180:         String averageConsumption = sHandler.calculateAverageConsumption(routeId);
181:         if (averageConsumption != null) {
182:             ((TextView) findViewById(R.id.average_consumption)).setText(" " + averageConsumption + " l");
183:         }
184:     }
```

```
1: package com.HFUwerdMobil.activity;
2:
3: import android.app.Activity;
4: import android.content.Intent;
5: import android.graphics.Point;
6: import android.os.Bundle;
7: import android.view.Display;
8: import android.view.Menu;
9: import android.view.MenuItem;
10: import android.view.View;
11: import android.widget.EditText;
12: import android.widget.ImageView;
13: import android.widget.Switch;
14: import android.widget.Toast;
15:
16: import com.HFUwerdMobil.main.R;
17: import com.HFUwerdMobil.memory.UserData;
18:
19: /**
20:  * Diese Klasse handelt die Einstellungs Activity. Die Activity enhÃ¤lt alle
21:  * Einstellungen die den Nutzer betreffen.
22:  *
23:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
24:  *         Veronika Kinzel
25:  * @version 1.0
26: */
27: public class SettingsActivity extends Activity {
28:     /**
29:      * Nutzer-Daten Objekt
30:      */
31:     UserData userData;
32:
33:     /**
34:      * Erzeugt und fÃ¶llt die View mit den Werten des Nutzers
35:      */
36:     @Override
37:     public void onCreate(Bundle savedInstanceState) {
38:         super.onCreate(savedInstanceState);
39:         setContentView(R.layout.activity_settings);
40:
41:         Display display = getWindowManager().getDefaultDisplay();
42:         Point size = new Point();
43:         display.getSize(size);
44:
45:         ImageView user_picture = (ImageView) findViewById(R.id.img_user_picture);
46:         user_picture.setLayoutParams().height = size.y / 3;
47:
48:         userData = new UserData(this);
49:
```

```
50:         EditText firstname = (EditText) findViewById(R.id.firstname);
51:         EditText lastname = (EditText) findViewById(R.id.lastname);
52:         EditText hometown = (EditText) findViewById(R.id.hometown);
53:         EditText phone = (EditText) findViewById(R.id.phone);
54:         EditText email = (EditText) findViewById(R.id.email);
55:         Switch obd = (Switch) findViewById(R.id.tBtn_obd);
56:
57:         email.setKeyListener(null);
58:         email.setEnabled(false);
59:
60:         firstname.setText(userData.getFirstname());
61:         lastname.setText(userData.getLastname());
62:         hometown.setText(userData.getHometownZip());
63:         phone.setText(userData.getMobileNumber());
64:         email.setText(userData.getUserMail());
65:
66:         if (userData.getUsesObd()) {
67:             obd.setChecked(true);
68:         } else {
69:             obd.setChecked(false);
70:         }
71:     }
72:
73:     /**
74:      * Erzeugt das Optionmenü
75:      */
76:     @Override
77:     public boolean onCreateOptionsMenu(Menu menu) {
78:         getMenuInflater().inflate(R.menu.popupmenu, menu);
79:         return true;
80:     }
81:
82:     /**
83:      * Handelt das Optionmenü
84:      */
85:     @Override
86:     public boolean onOptionsItemSelected(MenuItem item) {
87:         switch (item.getItemId()) {
88:             case R.id.menu_home:
89:                 startActivity(new Intent(this, ChooseRouteActivity.class));
90:                 return true;
91:             case R.id.menu_settings:
92:                 startActivity(new Intent(this, SettingsActivity.class));
93:                 return true;
94:             case R.id.menu_about:
95:                 startActivity(new Intent(this, AboutActivity.class));
96:                 return true;
97:             case R.id.menu_imprint:
98:                 startActivity(new Intent(this, ImprintActivity.class));
```

```
99:                     return true;
100:                case R.id.menu_help:
101:                    startActivity(new Intent(this, HelpActivity.class));
102:                    return true;
103:                default:
104:                    super.onOptionsItemSelected(item);
105:                }
106:            return false;
107:        }
108:
109:        /**
110:         * Speichert die Daten des Nutzers ab
111:         *
112:         * @param v
113:         *          UI-Element, welches die Speicherung ausgelöst hat
114:         */
115:        public void saveSettings(View v) {
116:            EditText firstname = (EditText) findViewById(R.id.firstname);
117:            EditText lastname = (EditText) findViewById(R.id.lastname);
118:            EditText hometown = (EditText) findViewById(R.id.hometown);
119:            EditText phone = (EditText) findViewById(R.id.phone);
120:            Switch obd = (Switch) findViewById(R.id.tBtn_obd);
121:
122:            boolean obdOn = obd.isChecked();
123:            if (obdOn) {
124:                userData.setUsesObd(true);
125:            } else {
126:                userData.setUsesObd(false);
127:            }
128:
129:            userData.setFirstname(firstname.getText().toString());
130:            userData.setLastname(lastname.getText().toString());
131:            userData.setHometownZip(hometown.getText().toString());
132:            userData.setMobileNumber(phone.getText().toString());
133:
134:            Toast.makeText(getApplicationContext(), getApplicationContext().getString(R.string.savedSettings), Toast.LENGTH_LONG).show();
135:        }
136:
137:        /**
138:         * Interruptet den Synchronisationsthread der Benutzerdaten, sobald die
139:         * Activity in den Hintergrund rückt
140:         */
141:        public void onPause() {
142:            super.onPause();
143:            if (userData.userDataSync != null)
144:                userData.userDataSync.interrupt();
145:        }
146:    }
```



```
1: package com.HFUwerdMobil.activity;
2:
3: import org.json.JSONException;
4: import org.json.JSONObject;
5:
6: import android.app.Activity;
7: import android.content.Context;
8: import android.content.Intent;
9: import android.os.Bundle;
10: import android.view.View;
11: import android.view.View.OnClickListener;
12: import android.widget.Button;
13: import android.widget.EditText;
14: import android.widget.TextView;
15:
16: import com.HFUwerdMobil.main.R;
17: import com.HFUwerdMobil.memory.UserData;
18: import com.HFUwerdMobil.sync.DataSyncVerifycode;
19: import com.HFUwerdMobil.util.IssueNotification;
20:
21: /**
22:  * Diese Klasse handelt die Benutzer Verifizierungs Activity
23:  *
24:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
25:  *         Veronika Kinzel
26:  * @version 1.0
27:  */
28: public class UserVerifyActivity extends Activity {
29:     /**
30:      * Benutzerdaten Objekt
31:      */
32:     UserData user;
33:
34:     /**
35:      * Unverifizierte E-Mail des Benutzers
36:      */
37:     String userEmail;
38:
39:     /**
40:      * Thread der die Kommunikation zum Server ansttzt, um die Verifizierung
41:      * durchzufren
42:      */
43:     Thread verifyThread;
44:
45:     /**
46:      * Erzeugt die Activity. Wartet auf den Verifizierungscode vom Server, der
47:      * zustzlich per E-Mail an die unverifizierte E-Mail Adresse des Benutzers
48:      * gesendet wurde. Wartet drauf, dass der Benutzer den korrekten
49:      * Verifizierungscode eingibt. Ist das der Fall, wird versucht die
```

```
50:         * Handynummer des Android GerÃ¤tes auszulesen. Ist das mÃ¶glich wird diese
51:         * als "Kontaktnummer" eingetragen, andernfalls ist der Nutzer aufgefordert
52:         * eine seine "Kontaktnummer" manuell einzugeben. Dabei wird die
53:         * "Kontaktnummer" validiert.
54:         */
55:     @Override
56:     protected void onCreate(Bundle savedInstanceState) {
57:         super.onCreate(savedInstanceState);
58:         setContentView(R.layout.activity_user_verify);
59:
60:         user = new UserData(this);
61:         userEmail = getIntent().getExtras().getString("unverified_email");
62:
63:         final String verifycode = getVerifycode(this);
64:         if (verifycode != null) {
65:             findViewById(R.id.btn_send_verify_code).setOnClickListener(new OnClickListener() {
66:                 @Override
67:                 public void onClick(View v) {
68:                     EditText userVerifycode = (EditText) findViewById(R.id.input_verify_code);
69:                     if (verifycode.equals(userVerifycode.getText().toString())) {
70:                         String deviceNumber = user.getDeviceNumber();
71:                         if (deviceNumber != null) {
72:                             user.initUser(userEmail, deviceNumber);
73:                             startActivityForResult(new Intent(getApplicationContext(), SettingsActiv
ity.class));
74:                         } else {
75:                             Button send_verify_code = (Button) findViewById(R.id.btn_send_v
erify_code);
76:                             TextView verifyUser = (TextView) findViewById(R.id.textView_ver
ifyUser);
77:
78:                             userVerifycode.setVisibility(View.GONE);
79:                             send_verify_code.setVisibility(View.GONE);
80:                             verifyUser.setVisibility(View.GONE);
81:
82:                             TextView text_info_number = (TextView) findViewById(R.id.textVi
ew_info_number);
83:                             TextView text_desc_number = (TextView) findViewById(R.id.textVi
ew_desc_number);
84:                             final EditText number = (EditText) findViewById(R.id.input_numb
er);
85:                             Button send_number = (Button) findViewById(R.id.btn_send_number
);
86:
87:                             text_info_number.setVisibility(View.VISIBLE);
88:                             text_desc_number.setVisibility(View.VISIBLE);
89:                             number.setVisibility(View.VISIBLE);
90:                             send_number.setVisibility(View.VISIBLE);
91:                         }
92:                     }
93:                 }
94:             });
95:         }
96:     }
```

```
92:         send_number.setOnClickListener(new OnClickListener() {
93:             @Override
94:             public void onClick(View v) {
95:                 String deviceNumber = number.getText().toString()
96: ;
97: ;
98: ;
99: ;
100:             Context(), SettingsActivity.class));
101:             UserVerifyActivity.this);
102: ;
103: ;
104: ;
105: ;
106: ;
107:             } else {
108:                 IssueNotification.verifycodeInputFailed(UserVerifyActivity.this);
109:             }
110:             });
111:         } else {
112:             IssueNotification.networkFailed(this);
113:         }
114:     }
115: }
116: /**
117: * Startet einen {@link com.HFUwerdMobil.sync.DataSyncVerifycode} Thread an
118: * und wartet auf das Ergebnis (in der Regel Verifizierungscode)
119: *
120: * @param context
121: *         Context Objekt
122: * @return Verifizierungscode oder null, wenn der Server kein oder einen
123: *         falschen Wert liefert
124: */
125: private String getVerifycode(Context context) {
126:     verifyThread = new Thread(new DataSyncVerifycode(userEmail, context));
127:     verifyThread.start();
128:     try {
129:         verifyThread.join();
130:     } catch (InterruptedException e1) {
131:         return null;
132:     }
133:     }
134:     String[] response = DataSyncVerifycode.VERIFYCODE_RESPONSE;
135:     if (response != null) {
136:         if (response[0].equals("200")) {
```

```
138:                     try {
139:                         return new JSONObject(response[1]).getString("verify_code");
140:                     } catch (JSONException e) {
141:                         return null;
142:                     }
143:                 }
144:             }
145:             return null;
146:         }
147:
148:         /**
149:          * Sobald die Activity in den Hintergrund rutscht, wird der
150:          * Verifizierungsthread und der Synchronisationsthread, der durch
151:          * {@link com.HFUwerdMobil.memory.UserData#initUser(String, String)}
152:          * angestoÃÂren wurde, unterbrochen
153:          */
154:         public void onPause() {
155:             super.onPause();
156:             if (verifyThread != null)
157:                 verifyThread.interrupt();
158:             if (user.userDataSync != null)
159:                 user.userDataSync.interrupt();
160:         }
161:     }
```

```
1: package com.HFUwerdMobil.memory;
2:
3: import java.util.ArrayList;
4: import java.util.List;
5:
6: import android.app.Activity;
7: import android.content.ContentValues;
8: import android.content.Context;
9: import android.database.Cursor;
10: import android.database.sqlite.SQLiteDatabase;
11: import android.database.sqlite.SQLiteException;
12:
13: import com.HFUwerdMobil.util.IssueNotification;
14:
15: /**
16:  * Diese Klasse handelt den Datenzugriff zur und von der internen App-Datenbank.
17:  * Sie stellt dabei entsprechende Methoden zur VerfÃ¼gung, die es erleichtern,
18:  * auf bestimmte Daten zuzugreifen, sowie bestimmte Daten in die Datenbank
19:  * einzutragen.
20:  *
21:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
22:  *         Veronika Kinzel
23:  * @version 1.0
24:  */
25: public class DataAccessHandler extends Activity {
26:     /** {@link android.database.sqlite.SQLiteDatabase} Objekt **/
27:     private SQLiteDatabase db;
28:
29:     /** {@link com.HFUwerdMobil.memory.DataAccessHelper} Objekt **/
30:     private DataAccessHelper sHelper;
31:
32:     /** {@link android.content.Context} Objekt **/
33:     private Context context;
34:
35:     /**
36:      * Erzeugt ein Hilfsobjekt fÃ¼r Datenbank Operationen
37:      *
38:      * @param context
39:      *          {@link android.content.Context} Objekt zur Erstellung des
40:      *          Hilfsobjekt
41:      * @see com.HFUwerdMobil.memory.DataAccessHelper
42:      */
43:     public DataAccessHandler(Context context) {
44:         sHelper = new DataAccessHelper(context);
45:     }
46:
47:     /**
48:      * Ã226ffnet die App-Datenbank, oder erzeugt diese mit Hilfe
49:      * {@link com.HFUwerdMobil.memory.DataAccessHelper#onCreate(SQLiteDatabase)}
```

```
50:         * , sollte die App-Datenbank nicht bestehen oder fÃ¼hrt bei Abweichung der
51:         * {@link com.HFUwerdMobil.memory.DataAccessHelper#DATABASE_VERSION} ein
52:         * Upgrade mit Hilfe
53:         * {@link com.HFUwerdMobil.memory.DataAccessHelper#onUpgrade(SQLiteDatabase, int, int)}
54:         * durch bzw. ein Downgrade mit Hilfe
55:         * {@link com.HFUwerdMobil.memory.DataAccessHelper#onDowngrade(SQLiteDatabase, int, int)}
56:         * . Kann die Datenbank nicht zum Schreiben geÃ¶ffnet werden, wird dem
57:         * Anwender das mitgeteilt und die App beendet.
58:         *
59:         */
60:     public void openDB() {
61:         try {
62:             db = sHelper.getWritableDatabase();
63:         } catch (SQLException ex) {
64:             IssueNotification.fatalError(context);
65:         }
66:     }
67:
68: /**
69: * SchlieÃt die Datenbank
70: */
71: public void closeDB() {
72:     sHelper.close();
73: }
74:
75: /**
76: * Speichert die Daten des Ã¼bergebenen
77: * {@link com.HFUwerdMobil.memory.EntityRoute} Objekt in der internen
78: * App-Datenbank
79: *
80: * @param route
81: *     {@link com.HFUwerdMobil.memory.EntityRoute} Objekt
82: * @return ID der neu zugefÃgten Route in der App-Datenbank, oder null falls
83: *         die Route nicht eingetragen werden konnte
84: */
85: public String addRoute(EntityRoute route) {
86:     ContentValues values = new ContentValues();
87:     values.put(DataAccessHelper.ROUTES_CID, route.getRouteId());
88:     values.put(DataAccessHelper.ROUTES_C1, route.getTimestamp());
89:     values.put(DataAccessHelper.ROUTES_C2, route.getStartLat());
90:     values.put(DataAccessHelper.ROUTES_C3, route.getStartLon());
91:     values.put(DataAccessHelper.ROUTES_C4, route.getEndPoint());
92:     if (db.insert(DataAccessHelper.TABLE_ROUTES, null, values) != -1) {
93:         return route.getRouteId();
94:     }
95:     return null;
96: }
97:
98: /**
```

```
99:             * Liefert das {@link com.HFUwerdMobil.memory.EntityRoute} Objekt zur
100:            * gegeben internal_route_id aus der App-Datenbank
101:            *
102:            * @param internal_route_id
103:            *          ID der Route, die aus der Datenbank gelesen werden soll
104:            * @return {@link com.HFUwerdMobil.memory.EntityRoute} Objekt der gegebenen
105:            *          internal_route_id, oder null falls internal_route_id nicht
106:            *          gefunden wurde
107:            */
108:        public EntityRoute getRoute(int internal_route_id) {
109:            Cursor cursor = db.query(DataAccessHelper.TABLE_ROUTES, new String[] { DataAccessHelper.ROUTES_C5, DataAccessHelper.ROUTES_C1,
110:                DataAccessHelper.ROUTES_C2, DataAccessHelper.ROUTES_C3, DataAccessHelper.ROUTES_C4, DataAccessHelper.ROUTES_CID },
111:                DataAccessHelper.ROUTES_C5 + "=?", new String[] { String.valueOf(internal_route_id) },
112:                null, null, null, null);
113:            if (cursor.moveToFirst()) {
114:                return new EntityRoute(internal_route_id, Integer.parseInt(cursor.getString(1)), cursor.getString(2),
115:                    Integer.parseInt(cursor.getString(3)),
116:                    Integer.parseInt(cursor.getString(4)), cursor.getString(5));
117:            }
118:            return null;
119:        }
120:        /**
121:         * Liefert das {@link com.HFUwerdMobil.memory.EntityRoute} Objekt zur
122:         * gegeben route_id aus der App-Datenbank
123:         *
124:         * @param route_id
125:         *          ID der Route, die aus der Datenbank gelesen werden soll
126:         * @return {@link com.HFUwerdMobil.memory.EntityRoute} Objekt der gegebenen
127:         *          route_id, oder null falls route_id nicht gefunden wurde
128:         */
129:        public EntityRoute getRoute(String route_id) {
130:            Cursor cursor = db.query(DataAccessHelper.TABLE_ROUTES, new String[] { DataAccessHelper.ROUTES_C5, DataAccessHelper.ROUTES_C1,
131:                DataAccessHelper.ROUTES_C2, DataAccessHelper.ROUTES_C3, DataAccessHelper.ROUTES_C4, DataAccessHelper.ROUTES_CID },
132:                DataAccessHelper.ROUTES_CID + "=?", new String[] { route_id }, null, null, null, null);
133:            if (cursor.moveToFirst()) {
134:                return new EntityRoute(Integer.parseInt(cursor.getString(0)), Integer.parseInt(cursor.getString(1)),
135:                    cursor.getString(2),
136:                    cursor.getString(3), Integer.parseInt(cursor.getString(4)), route_id);
137:            }
138:            return null;
139:        }
140:        /**
141:         * Liefert alle {@link com.HFUwerdMobil.memory.EntityRoute} Objekte aus der
```

```
141:         * App-Datenbank
142:         *
143:         * @return Liste mit allen Routen Objekten vom Typ
144:         * {@link com.HFUwerdMobil.memory.EntityRoute}, oder null falls
145:         * keine Route gefunden wurde
146:         */
147:     public List<EntityRoute> getAllRoutes() {
148:         List<EntityRoute> routeList = new ArrayList<EntityRoute>();
149:         Cursor cursor = db.rawQuery("SELECT * FROM " + DataAccessHelper.TABLE_ROUTES, null);
150:         if (cursor.moveToFirst()) {
151:             do {
152:                 EntityRoute route = new EntityRoute(Integer.parseInt(cursor.getString(0)), Integer.parseInt(cursor.getString(1)),
153:   cursor.getString(2), cursor.getString(3), Integer.parseInt(cursor.getString(4)),
154:   cursor.getString(5));
155:                 routeList.add(route);
156:             } while (cursor.moveToNext());
157:             return routeList;
158:         }
159:         return null;
160:     }
161:     /**
162:      * Liefert die interne ID der letzten in der App-Datenbank eingetragenen
163:      * Route zurÃ¼ck
164:      *
165:      * @return Letzte in der Datenbank eingetragene Route ID, oder null falls
166:      * keine Route verfÃ¼gbar ist
167:      */
168:     public int getLastAddedRouteId() {
169:         Cursor cursor = db.rawQuery("SELECT " + DataAccessHelper.ROUTES_C5 + " FROM " + DataAccessHelper.TABLE_
ROUTES + " ORDER BY "
170:                                     + DataAccessHelper.ROUTES_C5 + " DESC LIMIT 1", null);
171:         if (cursor.moveToFirst()) {
172:             return Integer.parseInt(cursor.getString(0));
173:         }
174:         return -1;
175:     }
176:     /**
177:      * Speichert die Daten des Ã¼bergebenen
178:      * {@link com.HFUwerdMobil.memory.EntityGPS} Objekt in der internen
179:      * App-Datenbank
180:      *
181:      * @param gps
182:      * {@link com.HFUwerdMobil.memory.EntityGPS} Objekt
183:      * @return ID des neu zugefÃ¼gten GPS-Satzes in der App-Datenbank, oder -1
184:      * falls der GPS-Satz nicht eingetragen werden konnte
185:      */
186: 
```

```
187:     public long addGPS(EntityGPS gps) {
188:         ContentValues values = new ContentValues();
189:         values.put(DataAccessHelper.GPS_C1, gps.getRouteId());
190:         values.put(DataAccessHelper.GPS_C2, gps.getTimestamp());
191:         values.put(DataAccessHelper.GPS_C3, gps.getLatitude());
192:         values.put(DataAccessHelper.GPS_C4, gps.getLongitude());
193:         return db.insert(DataAccessHelper.TABLE_GPS, null, values);
194:     }
195:
196:    /**
197:     * Speichert die Daten des übergebenen
198:     * {@link com.HFUwerdMobil.memory.EntityOBD} Objekt in der internen
199:     * App-Datenbank
200:     *
201:     * @param obd
202:     *      {@link com.HFUwerdMobil.memory.EntityOBD} Objekt
203:     *      @return ID des neu zugefügten OBD-Datensatzes in der App-Datenbank, oder
204:     *              -1 falls der OBD-Datensatz nicht eingetragen werden konnte
205:     */
206:    public long addOBD(EntityOBD obd) {
207:        ContentValues values = new ContentValues();
208:        values.put(DataAccessHelper.OBD_C1, obd.getRouteId());
209:        values.put(DataAccessHelper.OBD_C2, obd.getCurrentSpeed());
210:        values.put(DataAccessHelper.OBD_C3, obd.getCurrentConsumption());
211:        return db.insert(DataAccessHelper.TABLE_OBD, null, values);
212:    }
213:
214:    /**
215:     * Liefert alle {@link com.HFUwerdMobil.memory.EntityGPS} Objekte zur
216:     * gegebenen route_id aus der App-Datenbank aus der App-Datenbank
217:     *
218:     * @return Liste mit GPS Objekten vom Typ
219:     *          {@link com.HFUwerdMobil.memory.EntityGPS}, oder null falls keine
220:     *          GPS-Daten zur Route gefunden wurden
221:     */
222:    public List<EntityGPS> getRouteGps(String route_id) {
223:        List<EntityGPS> routeGpsList = new ArrayList<EntityGPS>();
224:        Cursor cursor = db.rawQuery("SELECT * FROM " + DataAccessHelper.TABLE_GPS + " WHERE " + DataAccessHelper.GPS_C1 + " = '" + route_id + "'", null);
225:        if (cursor.moveToFirst()) {
226:            do {
227:                EntityGPS gps = new EntityGPS(Integer.parseInt(cursor.getString(0)), cursor.getString(1), Integer.parseInt(cursor.getString(2)), cursor.getString(3), cursor.getString(4));
228:                routeGpsList.add(gps);
229:            } while (cursor.moveToNext());
230:        }
231:        return routeGpsList;
232:    }
233:}
```

```
234:             return null;
235:         }
236:
237:         /**
238:          * Liefert die ID des letzten in der App-Datenbank eingetragenen
239:          * GPS-Datensatzes zurÃ¼ck
240:          *
241:          * @return Letzte in der Datenbank eingetragene GPS ID, oder -1 falls kein
242:          *         GPS-Datensatz verfÃ¼gbar ist
243:          */
244:         public int getLastAddedGpsId() {
245:             Cursor cursor = db.rawQuery("SELECT " + DataAccessHelper.GPS_CID + " FROM " + DataAccessHelper.TABLE_GP
S + " ORDER BY "
246:   + DataAccessHelper.GPS_CID + " DESC LIMIT 1", null);
247:             if (cursor.moveToFirst()) {
248:                 return Integer.parseInt(cursor.getString(0));
249:             }
250:             return -1;
251:         }
252:
253:         /**
254:          * Liefert alle {@link com.HFUwerdMobil.memory.EntityOBD} Objekte aus zur
255:          * gegebenen route_id aus der App-Datenbank aus der App-Datenbank
256:          *
257:          * @return ArrayList mit Objekten vom Typ
258:          *          {@link com.HFUwerdMobil.memory.EntityOBD}, oder null falls keine
259:          *          OBD-Daten gefunden wurden
260:          */
261:         public List<EntityOBD> getRouteOBD(String route_id) {
262:             List<EntityOBD> routeOBDlist = new ArrayList<EntityOBD>();
263:             Cursor cursor = db.rawQuery("SELECT * FROM " + DataAccessHelper.TABLE_OBD + " WHERE " + DataAccessHelp
r.OBD_C1 + " = '" + route_id + "'",
264:   null);
265:             if (cursor.moveToFirst()) {
266:                 do {
267:                     EntityOBD obd = new EntityOBD(Integer.parseInt(cursor.getString(1)), Integer.parseInt(c
ursor.getString(2)), Integer.parseInt(cursor
268:   .getString(3)));
269:                     routeOBDlist.add(obd);
270:                 } while (cursor.moveToNext());
271:                 return routeOBDlist;
272:             }
273:             return null;
274:         }
275:
276:         /**
277:          * Liefert die ID des letzten in der App-Datenbank eingetragenen
278:          * OBD-Datensatzes zurÃ¼ck
279:          *
```

```
280:         * @return Letzte in der Datenbank eingetragene OBD ID, oder -1 falls kein
281:         * OBD-Datensatz verfÃ¼gbar ist
282:         */
283:     public int getLastAddedObdId() {
284:         Cursor cursor = db.rawQuery("SELECT " + DataAccessHelper.OBD_CID + " FROM " + DataAccessHelper.TABLE_OB
D + " ORDER BY "
285:                                     + DataAccessHelper.GPS_CID + " DESC LIMIT 1", null);
286:         if (cursor.moveToFirst()) {
287:             return Integer.parseInt(cursor.getString(0));
288:         }
289:         return -1;
290:     }
291:
292:     /**
293:      * Berechnet die durchschnittliche Geschwindigkeit aus den aktuellen
294:      * Geschwindigkeitsdaten die vom OBD2 erfasst wurden
295:      *
296:      * @return Aktuelle Geschwindigkeit oder null, falls keine OBD Daten erfasst
297:      * wurden
298:      */
299:     public String calculateAverageSpeed(String routeId) {
300:         List<EntityOBD> obdData = getRouteOBD(routeId);
301:         if (obdData != null) {
302:             int averageSpeed = 0;
303:             for (int i = 0; i < obdData.size(); i++) {
304:                 averageSpeed = averageSpeed + obdData.get(i).getCurrentSpeed();
305:             }
306:             return Integer.toString(averageSpeed / obdData.size());
307:         }
308:         return null;
309:     }
310:
311:     /**
312:      * Berechnet den durchschnittlichen Verbrauch aus den aktuellen
313:      * Verbrauchsdaten die vom OBD2 erfasst wurden
314:      *
315:      * @return Aktueller Verbrauch oder null, falls keine OBD Daten erfasst
316:      * wurden
317:      */
318:     public String calculateAverageConsumption(String routeId) {
319:         List<EntityOBD> obdData = getRouteOBD(routeId);
320:         if (obdData != null) {
321:             int averageConsumption = 0;
322:             for (int i = 0; i < obdData.size(); i++) {
323:                 averageConsumption = averageConsumption + obdData.get(i).getCurrentConsumption();
324:             }
325:             return Integer.toString(averageConsumption / obdData.size());
326:         }
327:         return null;
```

```
328:  
329:  
330: }
```

```
1: package com.HFUwerdMobil.memory;
2:
3: import android.content.Context;
4: import android.database.sqlite.SQLiteDatabase;
5: import android.database.sqlite.SQLiteOpenHelper;
6:
7: /**
8:  * Diese Klasse ist Hilfsklasse der Klasse
9:  * {@link com.HFUwerdMobil.memory.DataAccessHandler} und definiert die Struktur
10: * fÃ¼r die interne SQLite Datenbank zur Speicherung App interner Daten. Sie
11: * stellt weiterhin Methoden durch die UnterstÃützung der Klasse
12: * {@link android.database.sqlite.SQLiteOpenHelper} zur Erzeugung sowie zum
13: * Update der Datenbank bereit.
14: *
15: * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
16: *         Veronika Kinzel
17: * @version 1.0
18: */
19: public class DataAccessHelper extends SQLiteOpenHelper {
20:
21:     /**
22:      * Name der internen Datenbank
23:      */
24:     protected static final String DATABASE_NAME = "hfuwerdmobil.db";
25:
26:     /**
27:      * Version der internen SQLite Datenbank. Eine Änderung der Version hat zur
28:      * Folge, dass die Methode {@link #onUpgrade(SQLiteDatabase, int, int)}
29:      * aufgerufen wird, sofern die SQLite Datenbank auf dem auszufÃ¼hrenden
30:      * Device besteht und eine hier nicht identische Datenbank Version enthÃ¤lt.
31:      */
32:     // ACHTUNG: Bevor die Versionsnummer geÄndert wird, bitte Hinweise bei
33:     // onUpgrade bzw. bei onDowngrade beachten!
34:     protected static final int DATABASE_VERSION = 1;
35:
36:     /**
37:      * Name der Tabelle "Routes" fÃ¼r die interne SQLite Datenbank.
38:      */
39:     protected static final String TABLE_ROUTES = "routes";
40:
41:     /**
42:      * Spaltenname des PrimÃ¤rschlÃ¶ssels der Tabelle {@link #TABLE_ROUTES}
43:      */
44:     protected static final String ROUTES_C5 = "internal_route_id";
45:
46:     /**
47:      * Spaltenname 1 der Tabelle {@link #TABLE_ROUTES}
48:      */
49:     protected static final String ROUTES_C1 = "timestamp";
```

```
50:  
51:        /**  
52:         * Spaltenname 2 der Tabelle {@link #TABLE_ROUTES}  
53:         */  
54:        protected static final String ROUTES_C2 = "start_lat";  
55:  
56:        /**  
57:         * Spaltenname 3 der Tabelle {@link #TABLE_ROUTES}  
58:         */  
59:        protected static final String ROUTES_C3 = "start_lon";  
60:  
61:        /**  
62:         * Spaltenname 4 der Tabelle {@link #TABLE_ROUTES}  
63:         */  
64:        protected static final String ROUTES_C4 = "end_point";  
65:  
66:        /**  
67:         * Spaltenname 5 der Tabelle {@link #TABLE_ROUTES}  
68:         */  
69:        protected static final String ROUTES_CID = "route_id";  
70:  
71:        /**  
72:         * CREATE-Statement fÃ¼r die Tabelle {@link #TABLE_ROUTES}.  
73:         */  
74:        protected static final String ROUTES_CREATE_STATEMENT = "CREATE TABLE " + TABLE_ROUTES + "(" + ROUTES_C5 + " IN  
TEGER PRIMARY KEY AUTOINCREMENT, "  
75:   + ROUTES_C1 + " INTEGER NOT NULL, " + ROUTES_C2 + " TEXT NOT NULL, " + ROUTES_C3 + " TEXT NOT N  
ULL, " + ROUTES_C4 + " INTEGER NOT NULL, "  
76:   + ROUTES_CID + " TEXT)";  
77:  
78:        /**  
79:         * Name der Tabelle "GPS" fÃ¼r die interne SQLite Datenbank.  
80:         */  
81:        protected static final String TABLE_GPS = "gps";  
82:  
83:        /**  
84:         * Spaltenname des PrimÃ¤rschlÃ¼ssels der Tabelle {@link #TABLE_GPS}  
85:         */  
86:        protected static final String GPS_CID = "gps_id";  
87:  
88:        /**  
89:         * Spaltenname 1 der Tabelle {@link #TABLE_GPS}  
90:         */  
91:        protected static final String GPS_C1 = "route_id";  
92:  
93:        /**  
94:         * Spaltenname 2 der Tabelle {@link #TABLE_GPS}  
95:         */  
96:        protected static final String GPS_C2 = "timestamp";
```

```
97:
98:        /**
99:         * Spaltenname 3 der Tabelle {@link #TABLE_GPS}
100:        */
101:       protected static final String GPS_C3 = "lat";
102:
103:       /**
104:         * Spaltenname 4 der Tabelle {@link #TABLE_GPS}
105:        */
106:       protected static final String GPS_C4 = "lon";
107:
108:       /**
109:         * CREATE-Statement fÃ¼r die Tabelle {@link #TABLE_GPS}.
110:        */
111:       protected static final String GPS_CREATE_STATEMENT = "CREATE TABLE " + TABLE_GPS + "(" + GPS_CID + " INTEGER PR
IMARY KEY AUTOINCREMENT, "
112:   + GPS_C1 + " TEXT NOT NULL, " + GPS_C2 + " INTEGER NOT NULL, " + GPS_C3 + " TEXT NOT NULL, " +
GPS_C4 + " TEXT NOT NULL);";
113:
114:       /**
115:         * Name der Tabelle "OBD" fÃ¼r die interne SQLite Datenbank.
116:        */
117:       protected static final String TABLE_OBD = "obd";
118:
119:       /**
120:         * Spaltenname des PrimÃ¤rschlÃ¼ssels der Tabelle {@link #TABLE_OBD}
121:        */
122:       protected static final String OBD_CID = "obd_id";
123:
124:       /**
125:         * Spaltenname 1 der Tabelle {@link #TABLE_OBD}
126:        */
127:       protected static final String OBD_C1 = "route_id";
128:
129:       /**
130:         * Spaltenname 1 der Tabelle {@link #TABLE_OBD}
131:        */
132:       protected static final String OBD_C2 = "current_speed";
133:
134:       /**
135:         * Spaltenname 2 der Tabelle {@link #TABLE_OBD}
136:        */
137:       protected static final String OBD_C3 = "current_consumption";
138:
139:       /**
140:         * CREATE-Statement fÃ¼r die Tabelle {@link #TABLE_OBD}.
141:        */
142:       protected static final String OBD_CREATE_STATEMENT = "CREATE TABLE " + TABLE_OBD + "(" + OBD_CID + " INTEGER PR
IMARY KEY AUTOINCREMENT, "
```

```
./com/HFUwerdMobil/memory/DataAccessHelper.java      Tue Jan 20 22:34:17 2015      4
143:                               + OBD_C1 + " TEXT NOT NULL, " + OBD_C2 + " INTEGER NOT NULL, " + OBD_C3 + " INTEGER NOT NULL);"
;
144:
145:    /**
146:     * Erstellt ein Hilfsobjekt um Datenbanken erstellen, öffnen und managen zu
147:     * können
148:     *
149:     * @param context
150:     *          {@link android.content.Context} Objekt zur Erstellung des
151:     *          Hilfsobjekt
152:     * @see android.database.sqlite.SQLiteOpenHelper#SQLiteOpenHelper(Context,
153:     *          String, android.database.sqlite.SQLiteDatabase.CursorFactory, int)
154:     */
155:    protected DataAccessHelper(Context context) {
156:        super(context, DATABASE_NAME, null, DATABASE_VERSION);
157:    }
158:
159:    /**
160:     * Erstellt die Datenbank und führt die in {@link #ROUTES_CREATE_STATEMENT}
161:     * und {@link #GPS_CREATE_STATEMENT} definierten CREATE-Statements aus,
162:     * sofern die Datenbank auf dem Device nicht schon besteht
163:     *
164:     * @param db
165:     *          Datenbank Objekt
166:     * @see android.database.sqlite.SQLiteOpenHelper#onCreate(SQLiteDatabase)
167:     */
168:    @Override
169:    public void onCreate(SQLiteDatabase db) {
170:        db.execSQL(ROUTES_CREATE_STATEMENT);
171:        db.execSQL(GPS_CREATE_STATEMENT);
172:        db.execSQL(OBD_CREATE_STATEMENT);
173:    }
174:
175:    /**
176:     * Sofern die Datenbank Version auf dem Device älter ist als die in
177:     * {@link #DATABASE_VERSION} definierte Version, werden die Tabellen
178:     * entfernt und neu angelegt. Daten werden dabei NICHT migriert.
179:     *
180:     * @param db
181:     *          Datenbank Objekt
182:     * @param oldVersion
183:     *          Die alte Datenbank Version
184:     * @param newVersion
185:     *          Die neue Datenbank Version
186:     * @see android.database.sqlite.SQLiteOpenHelper#onUpgrade(SQLiteDatabase,
187:     *          int, int)
188:     */
189:    @Override
190:    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
```

```
191:         // ACHTUNG: Wenn die Versionsnummer geupdatet wird, sollten die Daten
192:         // des aktuellen Versionsschemas sorgfÃ¤ltig migriert werden und das am
193:         // besten in einem eigenem Thread (da das viel Zeit kostet)
194:         db.execSQL("DROP TABLE IF EXISTS " + TABLE_ROUTES);
195:         db.execSQL("DROP TABLE IF EXISTS " + TABLE_GPS);
196:         db.execSQL("DROP TABLE IF EXISTS " + TABLE_OBD);
197:         onCreate(db);
198:     }
199:
200:    /**
201:     * Sofern die Datenbank Version auf dem Device neuer ist als die in
202:     * {@link #DATABASE_VERSION} definierte Version, werden die Tabellen
203:     * entfernt und neu angelegt. Daten werden dabei NICHT migriert.
204:     *
205:     * @param db
206:     *         Datenbank Objekt
207:     * @param oldVersion
208:     *         Die alte Datenbank Version
209:     * @param newVersion
210:     *         Die neue Datenbank Version
211:     * @see android.database.sqlite.SQLiteOpenHelper#onUpgrade(SQLiteDatabase,
212:     *         int, int)
213:     */
214:    @Override
215:    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
216:        // ACHTUNG: Wenn die Versionsnummer gedowngradet wird, sollten die Daten
217:        // des aktuellen Versionsschemas sorgfÃ¤ltig migriert werden und das am
218:        // besten in einem eigenem Thread (da das viel Zeit kostet)
219:        db.execSQL("DROP TABLE IF EXISTS " + TABLE_ROUTES);
220:        db.execSQL("DROP TABLE IF EXISTS " + TABLE_GPS);
221:        db.execSQL("DROP TABLE IF EXISTS " + TABLE_OBD);
222:        onCreate(db);
223:    }
224:
225: }
```



```
1: package com.HFUwerdMobil.memory;
2:
3: /**
4:  * Diese Klasse repräsentiert ein Satz von GPS-Daten die einer bestimmten Route
5:  * {@link com.HFUwerdMobil.memory.EntityRoute}) zugeordnet werden.
6:  *
7:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
8:  *         Veronika Kinzel
9:  * @version 1.0
10: */
11: public class EntityGPS {
12:     /**
13:      * GPS ID des Datensatzes
14:      */
15:     private int gps_id;
16:
17:     /**
18:      * Route ID, welche der GPS-Datensatz zugeordnet ist
19:      */
20:     private String route_id;
21:
22:     /**
23:      * Zeitstempel im UNIX Format der Objekt-Konstruktion
24:      */
25:     private long timestamp;
26:
27:     /**
28:      * Latitude des Datensatzes
29:      */
30:     private String lat;
31:
32:     /**
33:      * Longitude des Datensatzes
34:      */
35:     private String lon;
36:
37:     /**
38:      * Setzt mit den übergebenen Daten das
39:      * {@link com.HFUwerdMobil.memory.EntityGPS} Objekt zusammen. Diese Methode
40:      * sollte nur verwendet werden, um Objekte mit den Daten aus der Datenbank
41:      * zu konstruieren. Sollen Daten Objekte für die Datenbank konstruiert
42:      * werden, sollte der
43:      * {@link com.HFUwerdMobil.memory.EntityGPS#EntityGPS(int, String, String)}
44:      * Konstruktor verwendet werden.
45:      *
46:      * @param gps_id
47:      *        Primärschlüssel des GPS-Datensatzes
48:      * @param route_id
49:      *        In Datenbank vorhandene Route ID, welche der GPS-Datensatz
```

```
50:             * zugeordnet wird
51:             * @param timestamp
52:             * Zeitstempel der Objekt-Konstruktion im UNIX Format
53:             * @param lat
54:             * Latitude
55:             * @param lon
56:             * Longitude
57:             */
58:     protected EntityGPS(int gps_id, String route_id, long timestamp, String lat, String lon) {
59:         this.gps_id = gps_id;
60:         this.route_id = route_id;
61:         this.timestamp = timestamp;
62:         this.lat = lat;
63:         this.lon = lon;
64:     }
65:
66:     /**
67:      * Setzt mit den Ã¼bergebenen Daten das
68:      * {@link com.HFUwerdMobil.memory.EntityGPS} Objekt zusammen.
69:      *
70:      * @param route_id
71:      *          In Datenbank vorhandene Route ID, welche der GPS-Datensatz
72:      *          zugeordnet wird
73:      * @param lat
74:      *          Latitude
75:      * @param lon
76:      *          Longitude
77:      */
78:     public EntityGPS(String route_id, String lat, String lon) {
79:         this.route_id = route_id;
80:         this.timestamp = System.currentTimeMillis() / 1000L;
81:         this.lat = lat;
82:         this.lon = lon;
83:     }
84:
85:     /**
86:      * Gibt die GPS ID des Datensatzes zurÃ¼ck
87:      *
88:      * @return GPS ID des Datensatzes
89:      */
90:     public int getGpsId() {
91:         return gps_id;
92:     }
93:
94:     /**
95:      * Gibt die Route ID zurÃ¼ck, welche der GPS-Datensatz zugeordnet ist
96:      *
97:      * @return Route ID zu der die GPS-Daten zugehÃ¶rig sind
98:      */
```

```
99:         public String getRouteId() {
100:             return route_id;
101:         }
102:
103:         /**
104:          * Gibt den Zeitstempel der Objekt-Konstruktion zurück
105:          *
106:          * @return UNIX Zeistempel des Datensatzes
107:          */
108:         public long getTimestamp() {
109:             return timestamp;
110:         }
111:
112:         /**
113:          * Gibt die Latitude des GPS-Datensatzes zurück
114:          *
115:          * @return Latitude des Datensatzes
116:          */
117:         public String getLat() {
118:             return lat;
119:         }
120:
121:         /**
122:          * Gibt die Longitude des Datensatzes zurück
123:          *
124:          * @return Longitude des Datensatzes
125:          */
126:         public String getLon() {
127:             return lon;
128:         }
129:     }
```



```
1: package com.HFUwerdMobil.memory;
2:
3: /**
4:  * Diese Klasse repräsentiert ein Satz von OBD-Daten die einer bestimmten Route
5:  * {@link com.HFUwerdMobil.memory.EntityRoute}) zugeordnet werden.
6:  *
7:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
8:  *         Veronika Kinzel
9:  * @version 1.0
10: */
11: public class EntityOBD {
12:     /**
13:      * OBD ID des Datensatzes
14:      */
15:     private int obd_id;
16:
17:     /**
18:      * Route ID, welche der OBD-Datensatz zugeordnet ist
19:      */
20:     private int route_id;
21:
22:     /**
23:      * Aktuelle Geschwindigkeit des OBD-Datensatzes
24:      */
25:     private int current_speed;
26:
27:     /**
28:      * Aktueller Verbrauch des OBD-Datensatzes
29:      */
30:     private int current_consumption;
31:
32:     /**
33:      * Setzt mit den übergebenen Daten das
34:      * {@link com.HFUwerdMobil.memory.EntityOBD} Objekt zusammen. Diese Methode
35:      * sollte nur verwendet werden, um Objekte mit den Daten aus der Datenbank
36:      * zu konstruieren. Sollen Daten Objekte für die Datenbank konstruiert
37:      * werden, sollte der
38:      * {@link com.HFUwerdMobil.memory.EntityOBD#EntityOBD(int, int, int)}
39:      * Konstruktor verwendet werden.
40:      *
41:      * @param obd_id
42:      *        Primärschlüssel des OBD-Datensatzes
43:      * @param route_id
44:      *        In Datenbank vorhandene Route ID, welche der OBD-Datensatz
45:      *        zugeordnet wird
46:      * @param current_speed
47:      *        Aktuelle Geschwindigkeit
48:      * @param current_consumption
49:      *        Aktueller Verbrauch
```

```
50:         */
51:     protected EntityOBD(int obd_id, int route_id, int current_speed, int current_consumption) {
52:         this.obd_id = obd_id;
53:         this.route_id = route_id;
54:         this.current_speed = current_speed;
55:         this.current_consumption = current_consumption;
56:     }
57:
58:    /**
59:     * Setzt mit den übergebenen Daten das
60:     * {@link com.HFUwerdMobil.memory.EntityOBD} Objekt zusammen.
61:     *
62:     * @param route_id
63:     *          In Datenbank vorhandene Route ID, welche der OBD-Datensatz
64:     *          zugeordnet wird
65:     * @param current_speed
66:     *          Aktuelle Geschwindigkeit
67:     * @param current_consumption
68:     *          Aktueller Verbrauch
69:     */
70:    public EntityOBD(int route_id, int current_speed, int current_consumption) {
71:        this.route_id = route_id;
72:        this.current_speed = current_speed;
73:        this.current_consumption = current_consumption;
74:    }
75:
76:    /**
77:     * Gibt die OBD ID des Datensatzes zurück
78:     *
79:     * @return OBD ID des Datensatzes
80:     */
81:    public int getObdId() {
82:        return obd_id;
83:    }
84:
85:    /**
86:     * Gibt die Route ID zurück, welche der OBD-Datensatz zugeordnet ist
87:     *
88:     * @return Route ID zu der die OBS-Daten zugehörig sind
89:     */
90:    public int getRouteId() {
91:        return route_id;
92:    }
93:
94:    /**
95:     * Gibt die aktuelle Geschwindigkeit des OBD-Datensatzes zurück
96:     *
97:     * @return Aktuelle Geschwindigkeit des Datensatzes
98:     */
```

```
99:         public int getCurrentSpeed() {
100:             return current_speed;
101:         }
102:
103:         /**
104:          * Gibt den aktuellen Verbrauch des OBD-Datensatzes zurÃ¼ck
105:          *
106:          * @return Aktueller Verbrauch des Datensatzes
107:          */
108:         public int getCurrentConsumption() {
109:             return current_consumption;
110:         }
111:     }
```



```
1: package com.HFUwerdMobil.memory;
2:
3: import android.app.Activity;
4: import android.content.Context;
5: import android.provider.Settings.Secure;
6:
7: import com.HFUwerdMobil.activity.LoginActivity;
8:
9: /**
10:  * Diese Klasse repräsentiert eine Route und beinhaltet grundlegende
11:  * Informationen zu dieser.
12:  *
13:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
14:  *         Veronika Kinzel
15:  * @version 1.0
16:  */
17: public class EntityRoute extends Activity {
18:
19:     /**
20:      * Die interne fortlaufende Route ID
21:      */
22:     private int internal_route_id;
23:
24:     /**
25:      * Die für externe Services bestimmte Route ID
26:      */
27:     private String route_id;
28:
29:     /**
30:      * Zeitstempel im UNIX Format der Objekt-Konstruktion
31:      */
32:     private long timestamp;
33:
34:     /**
35:      * Latitude des Startorts des Datensatzes
36:      */
37:     private String startLat;
38:
39:     /**
40:      * Longitude des Startorts des Datensatzes
41:      */
42:     private String startLon;
43:
44:     /**
45:      * Destination-Flag des Zielort (0: in den App-Settings eingetragener
46:      * Heimatort des App-Nutzers; 1: Furtwangen; 2: Schwenningen; 3: Tuttlingen)
47:      */
48:     private int endPoint;
49:
```

```
50:         /**
51:          * Setzt mit den Ã¼bergebenen Daten das
52:          * {@link com.HFUwerdMobil.memory.EntityRoute} Objekt zusammen. Diese
53:          * Methode sollte nur verwendet werden, um Objekte mit den Daten aus der
54:          * Datenbank zu konstruieren. Sollen Daten Objekte fÃ¼r die Datenbank
55:          * konstruiert werden, sollte der
56:          * {@link com.HFUwerdMobil.memory.EntityRoute#EntityRoute(String, String, int, Context)}
57:          * Konstruktor verwendet werden.
58:          *
59:          * @param internal_route_id
60:          *           Interne Route ID
61:          * @param timestamp
62:          *           Zeitstempel der Objekt-Konstruktion im UNIX Format
63:          * @param startLat
64:          *           Latitude des Startorts der Route
65:          * @param startLon
66:          *           Longitude des Startorts der Route
67:          * @param endPoint
68:          *           Destination-Flag des Zielort (0: in den App-Settings
69:          *           eingetragener Heimatort des App-Nutzers; 1: Furtwangen; 2:
70:          *           Schwenningen; 3: Tuttlingen)
71:          * @param route_id
72:          *           Die fÃ¼r externe Services verwendete Route ID, um bspw. auf dem
73:          *           Server eine wirklich eindeutige ID zu haben. Da bei jedem
74:          *           Nutzer der die App verwendet, intern ab 0 hochgezÃ¤hlt wird,
75:          *           wÃ¶rden die externen Services von jedem Nutzer eine Route mit
76:          *           der ID 0 erhalten. Um dies zu verhindern, wird fÃ¼r die
77:          *           externen Services eine wirkliche eindeutige ID generiert, die
78:          *           sich zusammensetzt auf Android Device ID + timestamp. Die
79:          *           Zusammensetzung mit zusÃ¤tzlichem timestamp soll es auÃ\237erdem
80:          *           ermÃ¶glichen, dass der gleiche Benutzer die App auf allen
81:          *           seinen Android GerÃ¤ten verwenden kann, und dass ggf. sogar
82:          *           (fast) gleichzeitig (timestamp).
83:          */
84:     protected EntityRoute(int internal_route_id, int timestamp, String startLat, String startLon, int endPoint, Str
ing route_id) {
85:         this.internal_route_id = internal_route_id;
86:         this.timestamp = timestamp;
87:         this.startLat = startLat;
88:         this.startLon = startLon;
89:         this.endPoint = endPoint;
90:         this.route_id = route_id;
91:     }
92:
93:     /**
94:      * Setzt mit den Ã¼bergebenen Daten das
95:      * {@link com.HFUwerdMobil.memory.EntityRoute} Objekt zusammen.
96:      *
97:      * @param startLat
```

```
98:             * Latitude des Startorts der Route
99:             * @param startLat
100:            * Longitude des Startorts der Route
101:           * @param endPoint
102:           *          Destination-Flag des Zielorts der Route (0: in den
103:           *          App-Settings eingetragener Heimatort des App-Nutzers; 1:
104:           *          Furtwangen; 2: Schwenningen; 3: Tuttlingen)
105:           * @param context
106:           *          Context Objekt
107:           */
108:      public EntityRoute(String startLat, String startLon, int endPoint, Context context) {
109:          this.route_id = Secure.getString(context.getContentResolver(), Secure.ANDROID_ID) + (System.currentTimeMillis() / 1000L);
110:          this.timestamp = (System.currentTimeMillis() / 1000L);
111:          this.startLat = startLat;
112:          this.startLon = startLon;
113:          this.endPoint = endPoint;
114:      }
115:
116:      /**
117:       * Gibt die Route ID des Datensatzes zurÃ¼ck
118:       *
119:       * @return Route ID des Datensatzes
120:       */
121:      public String getRouteId() {
122:          return route_id;
123:      }
124:
125:      /**
126:       * Gibt die interne RRoute ID des Datensatzes zurÃ¼ck
127:       *
128:       * @return Interne Route ID des Datensatzes
129:       */
130:      public int getInternalRouteId() {
131:          return internal_route_id;
132:      }
133:
134:      /**
135:       * Gibt den Zeitstempel der Objekt-Konstruktion zurÃ¼ck
136:       *
137:       * @return UNIX Zeitstempel des Datensatzes
138:       */
139:      public long getTimestamp() {
140:          return timestamp;
141:      }
142:
143:      /**
144:       * Gibt die Latitude des Startorts der Route zurÃ¼ck
145:       *
```

```
146:             * @return Latitude des Startorts der Route
147:             */
148:         public String getStartLat() {
149:             return startLat;
150:         }
151:
152:         /**
153:             * Gibt die Longitude des Startorts der Route zurück
154:             *
155:             * @return Longitude des Startorts der Route
156:             */
157:         public String getStartLon() {
158:             return startLon;
159:         }
160:
161:         /**
162:             * Gibt das Destination-Flag des Zielorts der Route zurück
163:             *
164:             * @return Destination-Flag des Zielorts der Route (0: in den App-Settings
165:             *         eingetragener Heimatort des App-Nutzers; 1: Furtwangen; 2:
166:             *         Schwenningen; 3: Tuttlingen)
167:             */
168:         public int getEndPoint() {
169:             return endPoint;
170:         }
171:
172:         /**
173:             * JSON-Objekt, welches beim Start einer Route an die externen Services
174:             * übertragen werden
175:             *
176:             * @return JSON-Objekt vom Routen Start
177:             */
178:         public String getJsonForRouteStart() {
179:             return "\""route_id\":"\" + getRouteId() + "\",\"timestamp\":" + (System.currentTimeMillis() / 1000L)
180:                 + "\",\"start_point\":{\"latitude\":" + getStartLat() + "\",\"longitude\":" + getSt
artLon() + "\"},\"end_point\":" +
181:                 + getEndPoint() + "\",\"token\":" + ";
182:         }
183:
184:         /**
185:             * JSON-Objekt, welches beim Beenden der Route an die externen Services
186:             * übertragen wird
187:             *
188:             * @return JSON-Objekt vom Routen Ende
189:             */
190:         public String getJsonForRouteEnd() {
191:             DataAccessHandler sHandler = new DataAccessHandler(LoginActivity.getLoginActivityContext());
192:             return "\""route_id\":"\" + getRouteId() + "\",\"timestamp\":" + (System.currentTimeMillis() / 1000L)
+ "\",\"total_duration_seconds\":" + "
```

```
./com/HFUwerdMobil/memory/EntityRoute.java      Wed Jan 21 18:10:35 2015      5
  193:                               + xyz() + "\",\"total_kilometers\":\"", + xyz() + "\",\"obd_info\":{\"average_speed\":"
"
  194:                               + sHandler.calculateAverageSpeed(getRouteId()) + "\",\"average_consumption\":\"", + sHan
dler.calculateAverageConsumption(getRouteId())
  195:                               + "\"},\"token\":\"";
  196: }
  197:
  198: // total_duration_seconds & total_kilometers aus gps daten berechnen
  199: private String xyz() {
 200:     return "0";
 201: }
 202: }
```



```
1: package com.HFUwerdMobil.memory;
2:
3: import java.util.regex.Pattern;
4:
5: import android.accounts.Account;
6: import android.accounts.AccountManager;
7: import android.app.Activity;
8: import android.content.Context;
9: import android.content.SharedPreferences;
10: import android.telephony.TelephonyManager;
11: import android.util.Patterns;
12:
13: import com.HFUwerdMobil.sync.DataSyncUser;
14: import com.HFUwerdMobil.util.IssueNotification;
15:
16: /**
17:  * Diese Klasse repräsentiert den App Nutzer. Sie enthält alle relevanten
18:  * Informationen zum Nutzer.
19:  *
20:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
21:  *         Veronika Kinzel
22:  * @version 1.0
23: */
24: public class UserData extends Activity {
25:     /** {@link android.content.Context} Objekt **/
26:     private Context context;
27:
28:     /** Speicher Objekt der User Daten **/
29:     private SharedPreferences userData;
30:
31:     /** Speicher Objekt für Setting Flags für die Datensynchronisation **/
32:     private SharedPreferences syncData;
33:
34:     /** Editor Objekt um die Daten der Speicher Objekte zu bearbeiten **/
35:     SharedPreferences.Editor editor;
36:
37:     /**
38:      * Thread für die Synchronisation der Userdaten zu den externen Services
39:      */
40:     public Thread userDataSync;
41:
42:     /**
43:      * \226ffnet Speicher Objekte
44:      *
45:      * @param context
46:      *         {@link android.content.Context} Objekt
47:      */
48:     public UserData(Context context) {
49:         this.context = context;
```

```
50:             userData = context.getSharedPreferences("userData", MODE_PRIVATE);
51:             syncData = context.getSharedPreferences("syncData", MODE_PRIVATE);
52:         }
53:
54:         /**
55:          * Gibt die im Android Gerät hinterlegte E-Mail Adresse zurück
56:          *
57:          * @return Android Google Account E-Mail Adresse, oder null falls keine
58:          *         hinterlegt
59:         */
60:        public String getDeviceMail() {
61:            Pattern emailRegex = Patterns.EMAIL_ADDRESS;
62:            Account[] accounts = AccountManager.get(context).getAccounts();
63:            for (Account account : accounts) {
64:                if (account.type.equals("com.google") && emailRegex.matcher(account.name).matches()) {
65:                    return account.name;
66:                }
67:            }
68:            return null;
69:        }
70:
71:        /**
72:          * Gibt die im Android Device hinterlegte mobile Nummer zurück
73:          *
74:          * @return Android Device mobile number, oder null falls keine hinterlegt
75:         */
76:        public String getDeviceNumber() {
77:            TelephonyManager tMgr = (TelephonyManager) context.getSystemService(Context.TELEPHONY_SERVICE);
78:            if (tMgr.getLine1Number() == "") {
79:                return null;
80:            }
81:            return tMgr.getLine1Number();
82:        }
83:
84:        /**
85:          * Prüft ob der Nutzer ein HFU-Mitglied ist
86:          *
87:          * @return true, wenn der Nutzer ein HFU-Mitglied ist
88:         */
89:        public boolean isHfuMember() {
90:            return userData.getBoolean("user_hfu_student", false);
91:        }
92:
93:        /**
94:          * Belegt die Anwender Einstellungen mit Default Werten. Bis auf email und
95:          * mobileNumber sind das null-Werte. Die E-Mail und die mobile Nummer sind
96:          * Pflichtwerte. Des Weiteren wird ein Synchronisation Thread angestoßen,
97:          * welcher die Daten an die externen Services überträgt.
98:         */
```

```
99:             * @param email
100:            *          GÄltige, verifizierte E-Mail Adresse des Nutzers
101:            * @param mobileNumber
102:            *          GÄltige mobile Nummer des Nutzers
103:            */
104:        public void initUser(String email, String mobileNumber) {
105:            SharedPreferences.Editor editor = userData.edit();
106:            editor.putString("user_email", email);
107:            editor.putString("firstname", null);
108:            editor.putString("lastname", null);
109:            editor.putString("mobile_number", mobileNumber);
110:            editor.putString("hometown_zip", null);
111:            editor.commit();
112:            if (isHfuMember()) {
113:                syncUpdatedData(-1);
114:            }
115:        }
116:
117:        /**
118:         * Ã234berprÃft, ob der App-Nutzer seine E-Mail Adresse im Google Account
119:         * seines Devices verÃndert hat. Bei einer Ã204nderung der Mail Adresse im
120:         * Google Account wird die neue E-Mail Adresse im Speicher Objekt userData
121:         * persistiert, sofern der Nutzer der Ã204nderung zustimmt. Wurde der Google
122:         * Account nicht gefunden, wird die Mail Adresse nicht geÃndert. Die E-Mail
123:         * wird ebenfalls nicht geÃndert, wenn der Nutzer die Ã204nderung bereits schon
124:         * einmal abgelehnt hat. Desweiteren wird ein Synchronisation Thread
125:         * angestoÃ237en, welcher die geÃnderten Daten an die externen Services
126:         * Ã4bertrÃ¤gt.
127:         */
128:        public void checkMailChange() {
129:            String currentMail = userData.getString("user_email", null);
130:            String newMail = null;
131:            Pattern emailRegex = Patterns.EMAIL_ADDRESS;
132:            Account[] accounts = AccountManager.get(context).getAccounts();
133:            for (Account account : accounts) {
134:                if (account.type.equals("com.google") && emailRegex.matcher(account.name).matches()) {
135:                    newMail = account.name;
136:                }
137:            }
138:
139:            if (currentMail != newMail && newMail != null && newMail != userData.getString("old_user_email", null)
&& !isHfuMember()) {
140:                SharedPreferences.Editor editor = userData.edit();
141:                IssueNotification.primaryMailChanged(context, newMail);
142:
143:                editor.putString("old_user_email", currentMail);
144:
145:                editor.commit();
146:                if (isHfuMember()) {
```

```
147:                     syncUpdatedData(0);
148:                 }
149:             }
150:         }
151:
152:         /**
153:          * Gibt das JSON-Objekt der User Daten zurÃ¼ck
154:          *
155:          * @return JSON-Objekt der UserDaten
156:          */
157:         public String getJson() {
158:             return "user_email\" :\"" + getUserMail() + "\",\"firstname\" :\"" + getFirstname() + "\",\"lastname\" :\""
" + getLastname()
159:                         + "\",\"mobile_number\" :\"" + getMobileNumber() + "\",\"hometown_zip\" :\"" + getHometownZip() + "\",\"token\" :\"";
160:         }
161:
162:         /**
163:          * Gibt den Vornamen des Nutzers zurÃ¼ck
164:          *
165:          * @return Vorname des Nutzers, oder null falls nicht gesetzt
166:          */
167:         public String getFirstname() {
168:             return userData.getString("firstname", null);
169:         }
170:
171:         /**
172:          * Setzt den Vornamen des Nutzers und stÃ¶tzt einen Synchronisations Thread
173:          * an, um die Ãnderungen den externen Services mitzuteilen.
174:          *
175:          * @param firstname
176:          *          Vorname des Nutzers
177:          */
178:         public void setFirstname(String firstname) {
179:             editor = userData.edit();
180:             editor.putString("firstname", firstname);
181:             editor.commit();
182:             if (isHfuMember()) {
183:                 syncUpdatedData(0);
184:             }
185:         }
186:
187:         /**
188:          * Gibt den Nachnamen des Nutzers zurÃ¼ck
189:          *
190:          * @return Nachname des Nutzers, oder null falls nicht gesetzt
191:          */
192:         public String getLastname() {
193:             return userData.getString("lastname", null);
```

```
194:         }
195:
196:        /**
197:         * Setzt den Nachnamen des Nutzers und startet einen Synchronisations Thread
198:         * an, um die Änderungen den externen Services mitzuteilen.
199:         *
200:         * @param lastname
201:             Nachname des Nutzers
202:         */
203:        public void setLastname(String lastname) {
204:            editor = userData.edit();
205:            editor.putString("lastname", lastname);
206:            editor.commit();
207:            if (isHfuMember()) {
208:                syncUpdatedData(0);
209:            }
210:        }
211:
212:        /**
213:         * Gibt die Mobile Nummer des Nutzers zurück
214:         *
215:         * @return Mobile Nummer des Nutzers, oder null falls nicht gesetzt
216:         */
217:        public String getMobileNumber() {
218:            return userData.getString("mobile_number", null);
219:        }
220:
221:        /**
222:         * Setzt die Mobile Nummer des Nutzers und startet einen Synchronisations
223:         * Thread an, um die Änderungen den externen Services mitzuteilen.
224:         *
225:         * @param mobile_number
226:             Mobile Nummer des Nutzers
227:         */
228:        public void setMobileNumber(String mobile_number) {
229:            editor = userData.edit();
230:            editor.putString("mobile_number", mobile_number);
231:            editor.commit();
232:            if (isHfuMember()) {
233:                syncUpdatedData(0);
234:            }
235:        }
236:
237:        /**
238:         * Gibt die Postleitzahl des Heimatorts des Nutzers zurück
239:         *
240:         * @return Postleitzahl des Heimatorts des Nutzers, oder null falls nicht
241:             gesetzt
242:         */
```

```
243:     public String getHometownZip() {
244:         return userData.getString("hometown_zip", null);
245:     }
246:
247:     /**
248:      * Setzt die Postleitzahl des Heimatorts des Nutzers und startet einen
249:      * Synchronisations Thread an, um die Änderungen den externen Services
250:      * mitzuteilen.
251:      *
252:      * @param hometownZip
253:          Postleitzahl des Heimatorts des Nutzers
254:      */
255:     public void setHometownZip(String hometownZip) {
256:         editor = userData.edit();
257:         editor.putString("hometown_zip", hometownZip);
258:         editor.commit();
259:         if (isHfuMember()) {
260:             syncUpdatedData(0);
261:         }
262:     }
263:
264:     /**
265:      * Gibt den Namen des Heimatorts des Nutzers zurück
266:      *
267:      * @return Namen des Heimatorts des Nutzers, oder null falls nicht gesetzt
268:      */
269:     public String getHometownName() {
270:         return userData.getString("hometown_name", null);
271:     }
272:
273:     /**
274:      * Gibt die Longitude des Heimatorts des Nutzers zurück
275:      *
276:      * @return Longitude des Heimatorts des Nutzers, oder null falls nicht
277:      * gesetzt
278:      */
279:     public String getHometownLon() {
280:         return userData.getString("hometown_lon", null);
281:     }
282:
283:     /**
284:      * Gibt die Langitude des Heimatorts des Nutzers zurück
285:      *
286:      * @return Langitude des Heimatorts des Nutzers, oder null falls nicht
287:      * gesetzt
288:      */
289:     public String getHometownLan() {
290:         return userData.getString("hometown_lan", null);
291:     }
```

```
292:  
293:     /**  
294:      * Gibt die E-Mail Adresse des Nutzers zurÃ¼ck  
295:      *  
296:      * @return E-Mail Adresse des Nutzers, oder null falls nicht gesetzt  
297:      */  
298:     public String getUserMail() {  
299:         return userData.getString("user_email", null);  
300:     }  
301:  
302:     /**  
303:      * Setzt die E-Mail Adresse des Nutzers und startet einen Synchronisations  
304:      * Thread an, um die Ãnderungen den externen Services mitzuteilen. Sollte  
305:      * nur aufgerufen werden, wenn die E-Mail Adresse vom System nicht ermittelt  
306:      * werden konnte.  
307:      *  
308:      * @param user_email  
309:          E-Mail Adresse des Nutzers  
310:      */  
311:     public void setUserMail(String user_email) {  
312:         editor = userData.edit();  
313:         editor.putString("user_email", user_email);  
314:         editor.commit();  
315:         if (isHfuMember()) {  
316:             syncUpdatedData(0);  
317:         }  
318:     }  
319:  
320:     /**  
321:      * Setzt den OBD-Status des Nutzers. Wird dieser Wert auf true gesetzt, so  
322:      * bedeutet dass, das beim Start einer Routenaufzeichnung der OBD gesucht  
323:      * wird.  
324:      *  
325:      * @param on  
326:          OBD-Status  
327:      */  
328:     public void setUsesObd(boolean on) {  
329:         editor = userData.edit();  
330:         editor.putBoolean("uses_oobd", on);  
331:         editor.commit();  
332:         if (isHfuMember()) {  
333:             syncUpdatedData(0);  
334:         }  
335:     }  
336:  
337:     /**  
338:      * Gibt den OBD-Status des Nutzers zurÃ¼ck  
339:      *  
340:      * @return true, wenn der Nutzer einen OBD-Adapter verwendet
```

```
341:         */
342:     public boolean getUsesObd() {
343:         return userData.getBoolean("uses_obd", false);
344:     }
345:
346:     /**
347:      * Setzt das User-Datensynchronisations Flag auf 0 (=Update) und startet die
348:      * Synchronisation
349:      *
350:      * @param userFlag
351:      *          Sync-Bit (0: Update; -1: Initialisierung; 1: kein Sync)
352:      */
353:     private void syncUpdatedData(int userFlag) {
354:         editor = syncData.edit();
355:         editor.putInt("user", userFlag);
356:         editor.commit();
357:         userDataSync = new Thread(new DataSyncUser(context, getJson()));
358:         userDataSync.start();
359:     }
360:
361: }
```

```
1: package com.HFUwerdMobil.obd2;
2:
3: import android.os.AsyncTask;
4: import android.util.Log;
5:
6: /**
7:  * Diese Klasse sorgt fÃ¼r die Initialisierung des ODB Adapters mit dem ELM
8:  * Protokoll. Dazu wird ELM Ã¼ber die AT Befehle wird das ELM Protokoll
9:  * zurÃ¼ckgesetzt, dannach werden die PIDS die benÃ¶tigt werden initialisiert. Die
10: * klasse hat einen BluetoothStream der schon mit dem OBD Adapter verbunden ist.
11: * Die Klasse lauft als Thread.
12: *
13: * @author OBD-Gruppe
14: * @version 1.0
15: * */
16: public class AutoInitObd2 extends AsyncTask<BluetoothStream, Integer, Integer> {
17:
18:     private BluetoothStream stream;
19:
20:     @Override
21:     protected Integer doInBackground(BluetoothStream... stream) {
22:         /**
23:          * Initialisierung des ELM Protokolls. Dabei wird das Protokoll durch
24:          * ATZ zurÃ¼ckgesetzt ATL0 aktiviert das senden von Headern ATSP0
25:          * initialisiert das ELM Protokoll das das Auto unterstÃützt automatisch
26:          * ATDP fragt ab ob die Abfrage automatisch passiert.
27:          *
28:          * FIRSTCOMMAND initialisiert die ersten 20 PIDs SECOUNDCOMMAND
29:          * initialisiert die PIDs 20-40 THIRDCOMMAND initialisiert die PIDs
30:          * 40-60
31:          *
32:          * Initialisierung dauert 22 Sekunden
33:          * */
34:         this.stream = stream[0];
35:
36:         int sup = 0;
37:
38:         initMsg(0, OBD2E.INITATZ);
39:         initMsg(OBD2E.LONGTIME, OBD2E.INITATE0);
40:         initMsg(OBD2E.MIDDLETIME, OBD2E.INITATL0);
41:
42:         initMsg(OBD2E.LONGTIME, OBD2E.AUTO_ATSP0);
43:         initMsg(OBD2E.LONGTIME, OBD2E.AUTO_ATDP);
44:         if (initMsg(OBD2E.LONGTIME, 3000, OBD2E.FIRSTCOMMAND, 2).contains("41"))
45:             sup += 1;
46:         if (initMsg(OBD2E.LONGTIME, 3000, OBD2E.SECONDCOMMAND, 1).contains("41"))
47:             sup += 2;
48:         if (initMsg(OBD2E.LONGTIME, 3000, OBD2E.THIRDCOMMAND, 1).contains("41"))
49:             sup += 3;
```

```
50:             return sup;
51:         }
52:
53:         private void initMsg(int time, String wrote) {
54:             /** normale readtime und einmalige wiederholung */
55:             initMsg(time, OBD2E.READTIME, wrote, 1);
56:         }
57:
58:         private String initMsg(int time, int readtime, String wrote, int readtimes) {
59:             /** schreibt eine Nachricht an den OBD Bluetooth Adapter */
60:             stream.write(OBD2E.sleepCommand(time, wrote));
61:             Log.e("wrote", wrote);
62:             String msg = "";
63:             for (int i = 0; i < readtimes; i++) {
64:                 OBD2E.sleep(readtime);
65:                 msg = Casts.getString(stream.read());
66:                 Log.e("read", msg);
67:             }
68:             return msg;
69:         }
70:
71:     }
```

```
1: package com.HFUwerdMobil.obd2;
2:
3: import java.io.IOException;
4: import java.lang.reflect.Method;
5: import java.util.ArrayList;
6: import java.util.UUID;
7:
8: import android.app.Activity;
9: import android.bluetooth.BluetoothAdapter;
10: import android.bluetooth.BluetoothDevice;
11: import android.bluetooth.BluetoothSocket;
12: import android.content.BroadcastReceiver;
13: import android.content.Context;
14: import android.content.Intent;
15: import android.content.IntentFilter;
16: import android.os.AsyncTask;
17: import android.os.Bundle;
18: import android.util.Log;
19: import android.view.View;
20: import android.view.Window;
21: import android.widget.AdapterView;
22: import android.widget.AdapterView.OnItemClickListener;
23: import android.widget.ArrayAdapter;
24: import android.widget.ListView;
25: import android.widget.Toast;
26:
27: import com.HFUwerdMobil.main.R;
28:
29: /**
30:  * Die Mainactivity ist der Einstiegspunkt des Bluetoothfeatures. Durch diese
31:  * Activity wird das Paaren (pairing) zweier GerÃ¤te intern Ã¶berwacht. Nach dem
32:  * pairing, werden Die GerÃ¤te miteinander Verbunden, diese Verbindung wird im
33:  * Attribute bluesocket gespeichert. Nach dem der Bluetoothsocket etabliert
34:  * wurde startet der OBD Service automatisch
35:
36:  * @author OBD-Gruppe
37:  * @version 1.0
38:  */
39: public class BluetoothActivity extends Activity {
40:
41:     private ArrayAdapter<String> listAdapter;
42:     private ListView listView;
43:     private Activity view;
44:     private BluetoothAdapter bluetooth;
45:     private ArrayList<BluetoothDevice> devices;
46:     private IntentFilter filter;
47:     private BroadcastReceiver receiver, receiver1, receiver2, receiver3, receiverpair;
48:     private OnItemClickListener listener;
49:     public static BluetoothSocket bluesocket;
```

```
50:         private static boolean isturnonrequest = false;
51:
52:         @Override
53:         public void onCreate(Bundle savedInstanceState) {
54:             /**
55:              * Hier werden Ähnlich wie in einem Konstruktor Startdaten festgelegt.
56:              * Es wird eine Layout verknüpft, das wir als XMLvordefiniert haben
57:              * (select_bluetooth). Nach dem Layout werden die einzelnen
58:              * LayoutElemente Verknüpft. Das wichtigste Element des Layouts ist die
59:              * Liste, Sie zeigt später die verfügbaren Bluetoothgeräte an. eine
60:              * Liste besteht aus einem GUI Element ListView und einem ListAdapter,
61:              * der den Inhalt der Liste verwaltet.
62:              *
63:              * Desweiteren wird hier der BluetoothAdapter bluetooth inizialisiert,
64:              * indem der Bluetoothsensor per getDefaultAdapter zur Verfügung
65:              * gestellt wird. Falls Bluetooth aus ist, oder kein Sensor vorhanden
66:              * ist, hat bluetooth den Wert null. Ist das der Fall, wird Bluetooth
67:              * angeschalten oder eine Meldung erscheint das kein Sensor vorhanden
68:              * ist.
69:              *
70:              * Zuletzt werden Intentfilter angemeldet. Diese werden durch Aktionen
71:              * der Bluetoothfilters ausgelöst, um später die Fortschritte zu
72:              * überwachen (tracing)
73:              */
74:             super.onCreate(savedInstanceState);
75:             requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
76:             setContentView(R.layout.activity_select_bluetooth);
77:             view = this;
78:             setProgressBarIndeterminateVisibility(false);
79:             listView = (ListView) findViewById(R.id.b_listView);
80:             listAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, 0);
81:             listView.setAdapter(listAdapter);
82:             listener = new ItemClickBT();
83:             listView.setOnItemClickListener(listener);
84:
85:             bluetooth = BluetoothAdapter.getDefaultAdapter();
86:             devices = new ArrayList<BluetoothDevice>();
87:
88:             if (bluetooth == null) {
89:                 Toast.makeText(getApplicationContext(), "No bluetooth detected", Toast.LENGTH_SHORT).show();
90:                 close();
91:             } else {
92:                 if (!bluetooth.isEnabled()) {
93:                     turnOnBT();
94:                 } else
95:                     startDiscovery();
96:
97:             }
98:             for (BluetoothDevice d : bluetooth.getBondedDevices())
```

```
99:                     unpairDevice(d);
100:
101:             filter = new IntentFilter(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
102:             receiverpair = new ReceiverBTPair();
103:             registerReceiver(receiverpair, filter);
104:             receiver = new ReceiverBT();
105:             filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
106:             registerReceiver(receiver, filter);
107:             receiver1 = new ReceiverBT();
108:             filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_STARTED);
109:             registerReceiver(receiver1, filter);
110:             receiver2 = new ReceiverBT();
111:             filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
112:             registerReceiver(receiver2, filter);
113:             receiver3 = new ReceiverBT();
114:             filter = new IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
115:             registerReceiver(receiver3, filter);
116:             reset();
117:
118:
119:     private void startDiscovery() {
120:         /**
121:          * startet eine Bluetooth GerÃ¤tesuche, und beendet die vorherige, falls
122:          * vorhanden
123:          */
124:         bluetooth.cancelDiscovery();
125:         bluetooth.startDiscovery();
126:
127:
128:
129:     private void reset() {
130:         /**
131:          * LÃ¶scht die Liste der Bluetooth GerÃ¤te und startet einen neuen
132:          * GerÃ¤tesuchlauf
133:          */
134:         devices.clear();
135:         startDiscovery();
136:
137:
138:     public void refresh(View view) {
139:         /**
140:          * Wrapper der ResetMethode fÃ¼r einen Button (erwartet intern View
141:          * Parameter)
142:          */
143:         reset();
144:
145:
146:     private void turnOnBT() {
147:         /**
```

```
148:             * isturnonrequest wird auf true gesetzt, dannach wird eine
149:             * Bluetoothanfrage ausgelÃ¶st
150:             */
151:             isturnonrequest = true;
152:             Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
153:             startActivityForResult(intent, 1);
154:         }
155:
156:         @Override
157:         protected void onActivityResult(int requestCode, int resultCode, Intent data) {
158:             /**
159:                 * Diese Methode wird aufgerufen, wenn die Methode ein Ergebnis
160:                 * zurÃ¼ckbekommt (Hier von der Bluetoothanfrage) falls das anschalten
161:                 * des Bluetooth nicht erfolg wird die App beendet falls das Bluetooth
162:                 * angeschalten wurde, wird das suchen nach GerÃ¤ten gestartet.
163:                 */
164:             super.onActivityResult(requestCode, resultCode, data);
165:             if (resultCode == RESULT_CANCELED) {
166:                 Toast.makeText(getApplicationContext(), "Bluetooth must be enabled to continue", Toast.LENGTH_S
HORT).show();
167:                 close();
168:             }
169:             isturnonrequest = false;
170:             startDiscovery();
171:         }
172:
173:         private class ReceiverBT extends BroadcastReceiver {
174:             /**
175:                 * Private Klasse ReceiverBT, diese Klasse arbeitet mit den Einkommenden
176:                 * Bluetooth Ereignissen. Der Receiver wurde oben mit IntentFiltern auf
177:                 * die Ereignisse spezialisiert.
178:                 */
179:
180:             @Override
181:             public void onReceive(Context context, Intent intent) {
182:                 /**
183:                     * diese Methode entscheidet was passiert, wenn ein angemeldetes
184:                     * Ereignis ausgelÃ¶st wird. 1) Ein neues BluetoothgerÃ¤t wird
185:                     * gefunden, falls es noch nicht vorhanden ist, wird es der Liste
186:                     * HinzugefÃ¼gt, dannach wird die Liste gelÃ¶scht und alle gefunden
187:                     * GerÃ¤te werden wieder eingetragen, damit GerÃ¤te die nicht mehr
188:                     * vorhanden sind auch nicht auftauchen. 2) GerÃ¤tesuche wird
189:                     * gestartet, Wartesymbol wird gesetzt. 3) GerÃ¤tesuche wird beendet,
190:                     * Wartesymbol wird versteckt. 4) Bluetooth wird deaktiviert, eine
191:                     * Anfrage zur Bluetoothaktivierung wird gesendet.
192:                     */
193:                 switch (intent.getAction()) {
194:                     case BluetoothDevice.ACTION_FOUND:
195:                         BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
```

```
196:  
197:             if (!devices.contains(device))  
198:                 devices.add(device);  
199:             listAdapter.clear();  
200:             for (BluetoothDevice dev : devices) {  
201:                 if (bluetooth.getBondedDevices().contains(dev)) {  
202:                     listAdapter.add(dev.getName() + "      paired      " + "\n" + dev.getAddress());  
dress());  
203:                 } else {  
204:                     listAdapter.add(dev.getName() + "\n" + dev.getAddress());  
205:                 }  
206:             }  
207:  
208:             break;  
209:         case BluetoothAdapter.ACTION_DISCOVERY_STARTED:  
210:             setProgressBarIndeterminateVisibility(true);  
211:             break;  
212:         case BluetoothAdapter.ACTION_DISCOVERY_FINISHED:  
213:             setProgressBarIndeterminateVisibility(false);  
214:             break;  
215:  
216:         case BluetoothAdapter.ACTION_STATE_CHANGED:  
217:             if (bluetooth.getState() == BluetoothAdapter.STATE_OFF && !isturnonrequest) {  
218:                 turnOnBT();  
219:             }  
220:         }  
221:     }  
222:  
223: }
```

224:

```
225: private class ItemClickBT implements OnItemClickListener {  
226:     /** Private Klasse die die Clicks auf Listensymbole auswertet */  
227:     @Override  
228:     public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
229:         /**  
230:          * Diese Methode wird durch eine Klick auf eine Listenelement  
231:          * ausgelöst. das Gerätesuchen wird beendet. Dannach wird das Gerät  
232:          * gepaired, oder falls es schon gepaired ist unpaired.  
233:          * */  
234:         if (bluetooth.isDiscovering()) {  
235:             bluetooth.cancelDiscovery();  
236:         }  
237:         BluetoothDevice device = devices.get(position);  
238:         if (bluetooth.getBondedDevices().contains(device)) {  
239:             unpairDeciveConnected(device);  
240:         } else {  
241:             new AsyncPair().execute(device);  
242:         }  
243:     }
```

```
244:  
245:        }  
246:  
247:        /** RMI METHODES OF BLUETOOTH **/  
248:        private void unpairDevice(BluetoothDevice device) {  
249:            /**  
250:             * Diese Methode lÃ¶scht ein gepairet es GerÃ¤t. Um auch GerÃ¤te ab 3.0 Zu  
251:             * nutzen, wird die private Methode removeBond aufgerufen.  
252:             * */  
253:            try {  
254:                Method method = device.getClass().getMethod("removeBond", (Class[]) null);  
255:                method.invoke(device, (Object[]) null);  
256:            } catch (Exception e) {  
257:                e.printStackTrace();  
258:            }  
259:        }  
260:  
261:        private void unpairDeciveConnected(BluetoothDevice device) {  
262:            /**  
263:             * Diese Methode lÃ¶scht alle gepairet es GerÃ¤t und gibt einen Text aus.  
264:             * */  
265:            unpairDevice(device);  
266:            Toast.makeText(getApplicationContext(), "Pairing dissconnected", Toast.LENGTH_SHORT).show();  
267:        }  
268:  
269:        private void pairDevice(BluetoothDevice device) {  
270:            /**  
271:             * Diese Methode paired ein GerÃ¤t. Um auch GerÃ¤te ab 3.0 Zu nutzen,  
272:             * wird die private Methode removeBond aufgerufen.  
273:             * */  
274:            try {  
275:                Method method = device.getClass().getMethod("createBond", (Class[]) null);  
276:                method.invoke(device, (Object[]) null);  
277:            } catch (Exception e) {  
278:                e.printStackTrace();  
279:            }  
280:        }  
281:  
282:        private class AsyncPair extends AsyncTask<BluetoothDevice, BluetoothDevice, BluetoothDevice> {  
283:            /**  
284:             * Dieser Thread fÃ¼hrt das Paaren zweiter BluetoothgerÃ¤te asynchrone aus  
285:             */  
286:  
287:            @Override  
288:            protected BluetoothDevice doInBackground(BluetoothDevice... params) {  
289:                /**  
290:                 * asynchrone werden alle GerÃ¤te gelÃ¶scht, dannach wir das richtige  
291:                 * GerÃ¤te mit dem Device gepaired.  
292:                 * */
```

```
293:             if (bluetooth.getBondedDevices().size() > 0)
294:                 for (BluetoothDevice d : bluetooth.getBondedDevices()) {
295:                     unpairDevice(d);
296:                 }
297:             pairDevice(params[0]);
298:             return params[0];
299:         }
300:
301:     @Override
302:     protected void onPreExecute() {
303:         /**
304:          * Wird vor der ausfÃ¼hrung des asynchronen Tasks ausgefÃ¼hrt, sorgt
305:          * fÃ¼r das Bessere UserverstÃ¤ndiss (Wartesymbol und Text was
306:          * passiert)
307:          * */
308:         super.onPreExecute();
309:         bluetooth.cancelDiscovery();
310:         view.setProgressBarIndeterminateVisibility(true);
311:         Toast.makeText(getApplicationContext(), "Pairing started", Toast.LENGTH_SHORT).show();
312:     }
313:
314: }
315:
316: private class ReceiverBTPair extends BroadcastReceiver {
317:     /**
318:      * Receiver der wie davor die Ereignisse des BluetoothgerÃ¤tes abfÃ¤ngt,
319:      * dieser Receiver ist auf den Verbindungsstatus von Bluetooth
320:      * spezialisiert. 1) Wird ausgelÃ¶st falls etwas mit Bluetooth passiert
321:      * 1.1) Falls das GerÃ¤t versucht hat sich zu paaren und nun gepaart ist,
322:      * wird eine Meldung ausgegeben das Bluetooth gepaired ist, auÃ\x237erdem
323:      * wird jetzt die verbindung (connection) gestartet. Und die Liste der
324:      * GerÃ¤te gelÃ¶scht. 1.2) Falls das GerÃ¤t vom Zustand gepaart zu nicht
325:      * gepaart wechselt wird hier die Liste der GerÃ¤te gelÃ¶scht
326:      *
327:      * das lÃ¶schen der Liste Ã¤hnelt einem update, da alle Werte erneut
328:      * eingetragen werden.
329:      *
330:      * */
331:
332:     @Override
333:     public void onReceive(Context context, Intent intent) {
334:         String action = intent.getAction();
335:
336:         if (BluetoothDevice.ACTION_BOND_STATE_CHANGED == action) {
337:             final int state = intent.getIntExtra(BluetoothDevice.EXTRA_BOND_STATE, BluetoothDevice.
ERROR);
338:             final int prevState = intent.getIntExtra(BluetoothDevice.EXTRA_PREVIOUS_BOND_STATE, BluetoothDevice.ERROR);
339:         }
340:     }
341:
```

```
./com/HFUwerdMobil/obd2/BluetoothActivity.java      Wed Jan 21 16:34:53 2015      8
 340:                     if (state == BluetoothDevice.BOND_BONDED && prevState == BluetoothDevice.BOND_BONDING)
 341:                         Toast.makeText(getApplicationContext(), "Pairing established", Toast.LENGTH_SHORT)
RT).show();
 342:                     reset();
 343:                     new AsyncConnect().execute((BluetoothDevice) bluetooth.getBondedDevices().toArray(
ay())[0]);
 344:                 } else if (state == BluetoothDevice.BOND_NONE && prevState == BluetoothDevice.BOND_BOND-
ED) {
 345:                     // gets also called when existing paired Devices get deleted
 346:                     reset();
 347:                 }
 348:             }
 349:         }
 350:     }
 351: }
 352:
 353: private class AsyncConnect extends AsyncTask<BluetoothDevice, BluetoothSocket, BluetoothSocket> {
 354:     /** Diese Klasse ist eine Thread der asynchrone die Verbindung etabliert */
 355:
 356:     @Override
 357:     protected BluetoothSocket doInBackground(BluetoothDevice... params) {
 358:         /**
 359:          * Diese Methode wird asynchron ausgefÃ¼hr, hier wird ein Gerät das
 360:          * gepaart wurde verbunden, dabei wird die GerätId des anderen
 361:          * Gerätes in die Methode createRfcommSocketToServiceRecord
 362:          * Ã¶bergeben, diese Methode öffnet den Bluetoothsocket Ã¶ber den die
 363:          * Communication läuft. Dannach wird der Socket verbunden. Fehler
 364:          * werden abgefangen und gelogged. Der Socket wird aus zurückgegeben.
 365:          */
 366:         BluetoothSocket socket = null;
 367:         try {
 368:             // MY_UUID is the app's UUID string, also used by the server
 369:             // code
 370:             Log.e("error s1", socket + "");
 371:             UUID deviceUuid = params[0].getUuids()[0].getUuid();
 372:             Log.e("Uuid", deviceUuid + "");
 373:             socket = params[0].createRfcommSocketToServiceRecord(deviceUuid);
 374:             Log.e("error s2", socket + "");
 375:             socket.connect();
 376:             Log.e("error s3", socket + "");
 377:         } catch (IOException e) {
 378:             Log.e("Bluetooth run", "connect failed:/n" + e.getStackTrace());
 379:             // Unable to connect; close the socket and get out
 380:             try {
 381:                 socket.close();
 382:                 socket = null;
 383:                 close();
 384:             } catch (IOException closeException) {
```

```
385:                     }
386:                 }
387:
388:             return socket;
389:         }
390:
391:         @Override
392:         protected void onPreExecute() {
393:             /**
394:              * Diese Methode wird vor dem asynchronen Thread ausgefÃ¼hrt und
395:              * informiert den Nutzer was gerade Passiert.
396:              */
397:             super.onPreExecute();
398:             view.setProgressBarIndeterminateVisibility(true);
399:             Toast.makeText(getApplicationContext(), "Connection started", Toast.LENGTH_SHORT).show();
400:         }
401:
402:         @Override
403:         protected void onPostExecute(BluetoothSocket result) {
404:             /**
405:              * Diese Methode wird nach dem Beenden des asynchronen Threads
406:              * ausgefÃ¼hrt, der Nutzer wird wieder informiert, desweiteren wird
407:              * der OBD2 Service gestartet. Falls das Verbinden Fehlgeschlagen
408:              * ist, erscheint eine Meldung.
409:              */
410:             super.onPostExecute(result);
411:             if (result != null) {
412:                 bluesocket = result;
413:                 Toast.makeText(getApplicationContext(), "Connection success", Toast.LENGTH_SHORT).show();
414:             }
415:             view.setProgressBarIndeterminateVisibility(false);
416:             startService(new Intent(getApplicationContext(), com.HFUwerdMobil.obd2.OBDservice.class);
417:             close();
418:         } else
419:             Toast.makeText(getApplicationContext(), "Connection failed", Toast.LENGTH_SHORT).show();
420:         }
421:     }
422:
423:     @Override
424:     public void onBackPressed() {
425:         /**
426:          * die Action bei dem RÃ¼ckwÃ¤rtsbutton wird hier Ã¼berschrieben */
427:         close();
428:     }
429:
430:     public void close() {
```

```
431:         /**
432:          * diese Funktion schlieÃ\237t die Activity und gibt zurÃ¼ck ob die
433:          * Verbindung geklappt hat oder nicht
434:          */
435:         Intent data = new Intent();
436:         if (bluesocket != null)
437:             data.putExtra("ison", true);
438:         else
439:             data.putExtra("ison", false);
440:         if (getParent() == null) {
441:             setResult(Activity.RESULT_OK, data);
442:         } else {
443:             getParent().setResult(Activity.RESULT_OK, data);
444:         }
445:         this.unregisterReceiver(receiver);
446:         this.unregisterReceiver(receiver1);
447:         this.unregisterReceiver(receiver2);
448:         this.unregisterReceiver(receiver3);
449:         this.unregisterReceiver(receiverpair);
450:         finish();
451:     }
452: }
453: }
```

```
1: package com.HFUwerdMobil.obd2;
2:
3: import java.io.IOException;
4: import java.io.InputStream;
5: import java.io.OutputStream;
6:
7: import android.bluetooth.BluetoothDevice;
8: import android.bluetooth.BluetoothSocket;
9: import android.util.Log;
10:
11: /**
12:  * Die Stream Klasse bietet die MÃ¶glichkeit einen Bluetoothsocket einfach zu
13:  * benutztten dabei sorgt die Klasse immer dafÃ¼r das der Socket verbunden ist,
14:  * und gleichzeitig die in und out put sockets benutzbar bleibe.
15: *
16: * @author OBD-Gruppe
17: * @version 1.0
18: */
19: public class BluetoothStream {
20:
21:     private BluetoothSocket socket;
22:     private BluetoothDevice device;
23:     private InputStream in;
24:     private OutputStream out;
25:
26:     public BluetoothStream(BluetoothSocket socket) throws SocketError {
27:         /**
28:          * Konstruktor der Klasse BluetootStram, welche einen BluetoothSocket
29:          * entgegennimmt, der Socket enthÃ¤lt das Verbunde GerÃ¤t, falls kein GrÃ¤t
30:          * verbunden ist, wird ein SocketError ausgelÃ¶st, da offensichtlich kein
31:          * Stream ohne zweites GerÃ¤t etabliert werden kann. Das verbunde GerÃ¤t
32:          * wir zusÃ¤tzlich im Attribute device gespeichert, falls die Verbindung
33:          * abbricht, kann so der BluetoothSocket neu erstellt werden
34:          */
35:         if (socket == null)
36:             throw new SocketError("Socket is null, BluetoothStream is useless");
37:         this.socket = socket;
38:         device = socket.getRemoteDevice();
39:
40:         try {
41:             in = socket.getInputStream();
42:             out = socket.getOutputStream();
43:         } catch (IOException e) {
44:             // Should never been thrown, because socket is never null
45:             e.printStackTrace();
46:         }
47:
48:     }
49:
```

```
50:     public BluetoothDevice getDevice() {
51:         /** device Getter */
52:         return device;
53:     }
54:
55:     public byte[] read() {
56:         /**
57:          * Diese Methode prÃ¤ft die Verbindung der GerÃ¤te, wenn diese Verbunden
58:          * sind, werden die eingehenden Daten ausgelesen und als Bytes
59:          * zurÃ¼ckgibt.
60:          */
61:         if (!isConnected()) {
62:             Log.e("read", "socket is not connected, but is should be");
63:             return null;
64:         }
65:         byte[] msg = null;
66:
67:         try {
68:             msg = new byte[OBD2E.PIDRECEIVELENGTH];
69:             in.read(msg);
70:         } catch (IOException e) {
71:             e.printStackTrace();
72:         }
73:         return msg;
74:     }
75:
76:     public boolean write(String sqz) {
77:         /**
78:          * Diese Methode prÃ¤ft die Verbindung der GerÃ¤te, wenn diese Verbunden
79:          * sind, werden die Ã¼bergebenen Daten als Bytes zum anderen GerÃ¤te
80:          * Ã¶bertragen.
81:          */
82:         if (!isConnected()) {
83:             Log.e("write", "socket is not connected, but is should be");
84:             return false;
85:         }
86:         try {
87:             out.write(sqz.getBytes());
88:             out.flush();
89:         } catch (IOException e) {
90:             e.printStackTrace();
91:         }
92:         return true;
93:     }
94:
95:     private boolean isConnected() {
96:         /**
97:          * Diese Methode prÃ¤ft die Verbindung, falls keine Verbindung vorhanden
98:          * ist, gibt wird einmal versucht sich mit dem gespeicherten GerÃ¤t zu
```

./com/HFUwerdMobil/obd2/BluetoothStream.java        Wed Jan 21 16:37:06 2015        3

```
99:                  * verbinden. Das Ergebnis der Verbindung wird durch einen Boolean
100:                * zurÃ¼ckgegeben (true fÃ¶r verbunden).
101:                * */
102:            if (!socket.isConnected())
103:                try {
104:                    socket.connect();
105:                catch (IOException e) {
106:                    e.printStackTrace();
107:                }
108:            return socket.isConnected();
109:        }
110:
111: }
```



```
1: package com.HFUwerdMobil.obd2;
2:
3: import android.util.Log;
4:
5: /**
6:  * Diese Klasse Castet die Ergebnisse des Bluetoothstreams NOVALUE ist der
7:  * RÃ¼ckgabewert bei falschen Eingaben
8:  *
9:  * @author OBD-Gruppe
10: * @version 1.0
11: */
12:
13: public class Casts {
14:     public static String NOVALUE = "NoData";
15:     private static String speedresponse = "410D";
16:     private static String rpmresponse = "410C";
17:     private static String iatresponse = "410F";
18:     private static String mapresponse = "410B";
19:
20:     private static final float GASCONST = 8.314f;
21:     private static final float AIRMASS = 28.97f;
22:     private static final float VOLUMERATE = 0.8f;
23:     private static final float ENGINEDIPLACEMENT = 1.6f;
24:
25:     public static String getString(byte... bs) {
26:         /** castet ein Byte Array zu einem String */
27:         String summe = "";
28:         for (byte b : bs) {
29:             summe += ((char) ((Byte) b).intValue());
30:         }
31:         return summe;
32:     }
33:
34:     private static String clearString(String msg) {
35:         /** lÃ¶scht alle Sonderzeichen aus einem String */
36:         msg = msg.trim();
37:         msg = msg.replaceAll("[ <>\\r\\n]", " ");
38:         return msg;
39:     }
40:
41:     public static String getSpeed(String msg) throws NumberFormatException {
42:         /**
43:          * gibt einen String als Speed zurÃ¼ck falls das nicht klappt gibt die
44:          * Funktion NOVALUE zurÃ¼ck
45:          */
46:         Log.e("msgspeed", msg);
47:         msg = clearString(msg);
48:         if (msg.contains(speedresponse)) {
49:             msg = msg.replace(speedresponse, "");
```

```
50:                     return Integer.parseInt(msg, 16) + "";
51:                 }
52:             return NOVALUE;
53:         }
54:
55:         public static double getRPM(String msg) throws NumberFormatException {
56:             /**
57:              * gibt einen Double als RMP zurÃ¼ck falls das nicht klappt gibt die
58:              * Funktion 0 zurÃ¼ck
59:              */
60:             Log.e("rpm", msg);
61:             msg = clearString(msg);
62:             if (msg.contains(rpmresponse)) {
63:                 msg = msg.replace(rpmresponse, "");
64:                 int A = Integer.parseInt(msg.substring(0, 2), 16);
65:                 int B = Integer.parseInt(msg.substring(2, 4), 16);
66:                 double rpm = ((A * 256) + B) / 100;
67:                 // Log.e("msgrpm", rpm+"");
68:                 return rpm;
69:             }
70:             return 1;
71:         }
72:
73:         public static double getIAT(String msg) throws NumberFormatException {
74:             /**
75:              * gibt einen Double als RMP zurÃ¼ck falls das nicht klappt gibt die
76:              * Funktion 0 zurÃ¼ck
77:              */
78:             Log.e("iat", msg);
79:             msg = clearString(msg);
80:             if (msg.contains(iatresponse)) {
81:                 msg = msg.replace(iatresponse, "");
82:                 double iat = Integer.parseInt(msg, 16) - 40;
83:                 // Log.e("msgiat", iat+"");
84:                 return iat;
85:             }
86:             return 1;
87:         }
88:
89:         public static double getMAP(String msg) throws NumberFormatException {
90:             /**
91:              * gibt einen Double als RMP zurÃ¼ck falls das nicht klappt gibt die
92:              * Funktion 0 zurÃ¼ck
93:              */
94:             Log.e("map", msg);
95:             msg = clearString(msg);
96:             if (msg.contains(mapresponse)) {
97:                 msg = msg.replace(mapresponse, "");
98:                 double map = Integer.parseInt(msg, 16);
```

```
99:                     // Log.e("msgmap", map+"");
100:                    return map;
101:                }
102:            return 1;
103:        }
104:
105:        private static double getMAF(double rpm, double map, double iat) {
106:            /**
107:             * returns MAF (Air Flow Rate) because it is not always supported
108:             */
109:            double imap = rpm * map / iat / 2;
110:            double maf = (imap / 60) * VOLUMERATE * ENGINEDIPLACEMENT * (AIRMASS * GASCONST);
111:            return maf;
112:        }
113:
114:        public static double getFuel(double rpm, double map, double iat) {
115:            /**
116:             * returns fuel
117:             */
118:            double maf = getMAF(rpm, map, iat);
119:            double gof = (maf / 14.7 / 6.17 / 454);
120:            double lph = gof * 3.78541178 * 100;
121:            return lph;
122:        }
123:    }
```



```
1: package com.HFUwerdMobil.obd2;
2:
3: import java.util.concurrent.ExecutionException;
4:
5: import android.util.Log;
6:
7: import com.HFUwerdMobil.memory.DataAccessHandler;
8: import com.HFUwerdMobil.memory.EntityOBD;
9:
10: /**
11:  * Diese Klasse kÃ¶mmert sich um den Datenaustausch zwischen Handy und Gerät. Sie
12:  * verbirgt die Verbindung mit dem OBD2 Gerät. Dazu initialisiert Sie zuerst die
13:  * OBD Protokolle Ã¼ber einer Instanz der Klasse AutoInitObd2, dannach frÃ¤gt Sie
14:  * verschiedene PIDS ab und schreibt Sie in eine Datenbank.
15:  *
16:  * stream ist die Bluetoothverbindung zum OBD Adapter send ist eine Instanz des
17:  * privaten Threads BgSend pidsupport hÃ¤lt die Anzahl der unterstÃ¤tzten PIDs
18:  *
19:  * @author OBD-Gruppe
20:  * @version 1.0
21:  */
22: public class Connect {
23:
24:     private BluetoothStream stream;
25:     private BgSend send;
26:     private int pidsupport;
27:
28:     private DataAccessHandler sHandler = new DataAccessHandler(com.HFUwerdMobil.activity.RecordRouteActivity.getRou
teRecordContext());
29:
30:     public Connect() {
31:         /** erstellt die Instanzen */
32:         send = new BgSend();
33:         try {
34:             stream = new BluetoothStream(BluetoothActivity.bluesocket);
35:         } catch (SocketError e) {
36:             e.printStackTrace();
37:         }
38:     }
39:
40:     public void start() {
41:         /** Startet die OBD2 Verbindung */
42:         send.start();
43:     }
44:
45:     public void close() {
46:         /** Beendet den Thread send */
47:         send.close();
48:     }
```

```
49:  
50:    private void saveToDB(String... msg) {  
51:        /** Hier werden die übergeben Werte in die Datenbank gespeichert */  
52:        sHandler.addOBD(new EntityOBD(sHandler.getLastAddedRouteId(), Integer.parseInt(msg[0]), Integer.parseInt(msg[1])));  
53:    }  
54:  
55:  
56:    private class BgSend extends Thread {  
57:        /**  
58:            * @author Alex privater Thread, der neben der MAIN läuft um deren  
59:            * Thread nicht zu blocken und so eine Exception zu werfen, die  
60:            * die App abstürzen lassen kann. Hier werden Daten an den OBD  
61:            * Adapter gesendet und abgefragt  
62:            *  
63:            * running dient zum ausschalten des Threads  
64:            */  
65:        public boolean running;  
66:  
67:        public BgSend() {  
68:            /** Thread wird angeschalten */  
69:            running = true;  
70:        }  
71:  
72:        @Override  
73:        public void run() {  
74:            /**  
75:            * diese Methode läuft asynchrone zuerst wird OBD initialisiert,  
76:            * über eine Instanz von AutoInitObd2 dieser Thread wartet bis die  
77:            * Initialisierung abgeschlossen wird und prüft an den Rückgabewerten  
78:            * welche PIDS unterstützt werden in der folgenden while Schleife  
79:            * werden dann immer wieder speed und fuel vom OBD Adapter  
80:            * abgefragt. Die Ergebnisse werden über die Casts in den richtigen  
81:            * Datentyp konvertiert und dann an die saveToDB übergeben  
82:            *  
83:            * Dauer für einen Durchlauf 8 Sekunden.  
84:            */  
85:        super.run();  
86:        AutoInitObd2 init = new AutoInitObd2();  
87:        init.execute(stream);  
88:        try {  
89:            pidsupport = init.get();  
90:            OBD2E.sleep(OBD2E.MIDDLETIME);  
91:            switch (pidsupport) {  
92:                case 0:  
93:                    Log.i("PID SUPPORT", "NONE");  
94:                    break;  
95:                case 1:  
96:                    Log.i("PID SUPPORT", "0-20");
```

```
97:                                break;
98:
99:
100:
101:
102:
103:
104:
105:        case 2:
106:            Log.i("PID SUPPORT", "20-40");
107:            break;
108:        case 3:
109:            Log.i("PID SUPPORT", "0-40");
110:            break;
111:        case 4:
112:            Log.i("PID SUPPORT", "0-20,40-60");
113:            break;
114:        case 5:
115:            Log.i("PID SUPPORT", "20-40,40-60");
116:            break;
117:        case 6:
118:            Log.i("PID SUPPORT", "0-60");
119:            break;
120:        }
121:    } catch (InterruptedException | ExecutionException e1) {
122:        e1.printStackTrace();
123:    }
124:    String fuel = "";
125:    String speed = "";
126:    double rpm, iat, map;
127:
128:    sHandler.openDB();
129:    while (running) {
130:        String num = OBD2E.Msg(OBD2E.sleepCommand(OBD2E.MIDDLETIME, OBD2E.M1_2_PID_SPEED));
131:        stream.write(num);
132:        Log.i("wrote: ", num);
133:        speed = readSpeed(OBD2E.READTIME);
134:        Log.i("read ", speed);
135:
136:        num = OBD2E.Msg(OBD2E.sleepCommand(OBD2E.LONGTIME, OBD2E.M1_2_PID_RPM));
137:        stream.write(num);
138:        Log.i("wrote: ", num);
139:        rpm = readRPM(OBD2E.READTIME);
140:        Log.i("read ", rpm + "");
141:
142:        num = OBD2E.Msg(OBD2E.sleepCommand(OBD2E.LONGTIME, OBD2E.M1_2_PID_IAT));
143:        stream.write(num);
144:        Log.i("wrote: ", num);
145:        iat = readIAT(OBD2E.READTIME);
146:        Log.i("read ", iat + "");
```

```
146:  
147:             fuel = Casts.getFuel(rpm, map, iat) + "";  
148:             Log.i("fuel", fuel);  
149:             saveToDB(speed, fuel);  
150:         }  
151:         sHandler.closeDB();  
152:     }  
153:  
154:     protected void close() {  
155:         /** Thread wird beended */  
156:         running = false;  
157:     }  
158:  
159:     private double readRPM(int timeout) {  
160:         /** Methode die RPM liest */  
161:         OBD2E.sleep(timeout);  
162:         return Casts.getRPM(Casts.getString(stream.read()));  
163:     }  
164:  
165:     private double readIAT(int timeout) {  
166:         /** Methode die IAT liest */  
167:         OBD2E.sleep(timeout);  
168:         return Casts.getIAT(Casts.getString(stream.read()));  
169:     }  
170:  
171:     private double readMAP(int timeout) {  
172:         /** Methode die MAP liest */  
173:         OBD2E.sleep(timeout);  
174:         return Casts.getMap(Casts.getString(stream.read()));  
175:     }  
176:  
177:     private String readSpeed(int timeout) {  
178:         /** Die Methode gibt die Geschwindigkeit zurÃ¼ck */  
179:         OBD2E.sleep(timeout);  
180:         return Casts.getSpeed(Casts.getString(stream.read()));  
181:     }  
182:  
183: }  
184: }
```

```
1: package com.HFUwerdMobil.obd2;
2:
3: /**
4:  * For more support look at http://en.wikipedia.org/wiki/OBD-II_PIDs statische
5:  * Klasse die die PIDs und einige Support Methoden enthÃ¤lt
6:  *
7:  * @author OBD-Gruppe
8:  * @version 1.0
9:  *
10: */
11: public class OBD2E {
12:
13:     public static int M_CURRENTDATA = 1;
14:     public static int M_FREEZFRAME = 2;
15:     // public static int M_TROUBLECODES = 3;
16:     // public static int M_CLEARTROUBLECODES = 4;
17:     public static int M_VEHICLEINFO = 9;
18:
19:     public static final char fin = '\r';
20:     public static final int PIDRECEIVELENGTH = 24;
21:
22:     public static final String INITATZ = "ATZ" + fin;
23:     public static final String INITATS0 = "AT$0" + fin;
24:     public static final String INITATA0 = "ATATO" + fin;
25:     public static final String INITATST10 = "ATST10" + fin;
26:     public static final String INITATSS = "ATSS" + fin;
27:     public static final String INITATSI = "ATSI" + fin;
28:     public static final String INITATH1 = "ATH1" + fin;
29:     public static final String INITATE0 = "ATE0" + fin;
30:     public static final String INITATL0 = "ATL0" + fin;
31:     public static final String INITATST = "ATST" + fin;
32:     public static final String FIRSTCOMMAND = "0100" + fin;
33:     public static final String SECOUNDCOMMAND = "0120" + fin;
34:     public static final String THIRDCOMMAND = "0140" + fin;
35:
36:     public static final int SHORTTIME = 500;
37:     public static final int MIDDLETIME = 1000;
38:     public static final int LONGTIME = 2000;
39:     public static final int READTIME = 400;
40:
41:     public static String INITATSP(String i) {
42:         /** Methode die ein bestimmtes Protokoll ausfÃ¼hrt */
43:         return "ATSP" + i + fin;
44:     }
45:
46:     public static String AUTO_ATSP0 = "ATSP$0" + fin;
47:     public static String AUTO_ATDP = "ATDP" + fin;
48:
49:     public static final String M1_2_PID_PIDSUPPORT_20 = "00";
```

```
./com/HFUwerdMobil/obd2/OBD2E.java      Wed Jan 21 16:41:16 2015      2

50:      public static final String M1_2_PID_FUELSYSSTATUS = "03";
51:      public static final String M1_2_PID_ENGINELOADVALUE = "04";
52:      public static final String M1_2_PID_ENGINETEMPERATURE = "05";
53:      public static final String M1_2_PID_FUELSHORTTERMB1 = "06";
54:      public static final String M1_2_PID_FUELSHORTTERMB2 = "08";
55:      public static final String M1_2_PID_FUELLONGTERMB1 = "07";
56:      public static final String M1_2_PID_FUELLONGTERMB2 = "09";
57:      public static final String M1_2_PID_FUELPRESSURE = "0A";
58:      public static final String M1_2_PID_RPM = "0C";
59:      public static final String M1_2_PID_ENGINEFUEL = "5E";
60:      public static final String M1_2_PID_SPEED = "0D";
61:      public static final String M1_2_PID_IAT = "0F";
62:      public static final String M1_2_PID_MAP = "0B";
63:      public static final String M1_2_PID_MAF = "10";
64:      public static final String M1_2_PID_ODBSTANDART = "1C";
65:      public static final String M1_2_PID_ENGINERUNTIME = "1F";
66:      public static final String M1_2_PID_FUELTYPE = "51";
67:      public static final String M1_2_PID_PIDSUPPORT_40 = "20";
68:      public static final String M1_2_PID_FUELREMAINING = "2F";
69:
70:      public static final String M9_PID_PIDSUPPORT_20 = "00";
71:      public static final String M9_PID_VIN = "02";
72:      public static final String M9_PID_ECUENAME = "0A";
73:
74:      /**
75:       * returns the max value of bytes returned -> normally 2-4 because of m9 its
76:       * 20
77:       */
78:      public static int MAX_RETURN_LENGTH = 20;
79:
80:      public static String Msg(String pid) {
81:          /** Methode die den PID mit Mode 1 zurückgibt */
82:          return "01" + pid + fin;
83:      }
84:
85:      public static String Msg(int mode, String pid) {
86:          /** Methode die den PID mit dem Mode zurückgibt */
87:          String m = Integer.toHexString(mode);
88:          return "0" + m + pid + fin;
89:      }
90:
91:      public static synchronized String sleepCommand(int time, String command) {
92:          /**
93:           * Methode die einen Thread eine bestimmte Zeit warten lässt und dann
94:           * den eingegeben Befehl zurück gibt.
95:           */
96:           sleep(time);
97:           return command;
98:      }
```

```
99:  
100:    public static void sleep(int time) {  
101:        /** lässt einen Thread einen bestimmte Zeit warten */  
102:        try {  
103:            Thread.sleep(time);  
104:        } catch (InterruptedException e) {  
105:            e.printStackTrace();  
106:        }  
107:    }  
108:  
109: }
```



```
1: package com.HFUwerdMobil.obd2;
2:
3: import java.io.Serializable;
4:
5: import android.app.Service;
6: import android.content.Intent;
7: import android.os.IBinder;
8: import android.util.Log;
9:
10: public class OBDservice extends Service implements Serializable {
11:
12:     /**
13:      * Service der gestartet und gestoppt werden kann, er verwaltet den Ablauf
14:      * der OBD2 verbindung, und läuft im Hintergrund der App. Dazu führt die
15:      * Klasse eine Instanz der Connection Klasse aus.
16:      *
17:      * serialVersionUID ist eine DefaultID Connect obd2 ist die Instanz der
18:      * Connect Klasse
19:      *
20:      * @author OBD-Gruppe
21:      * @version 1.0
22:      */
23:     private static final long serialVersionUID = 1L;
24:     private Connect obd2;
25:
26:     @Override
27:     public IBinder onBind(Intent arg0) {
28:         /** Unwichtige Methode da der Service ohne Activity läuft */
29:         return null;
30:     }
31:
32:     @Override
33:     public void onCreate() {
34:         /** Service wird erstellt, Instanz wird erstellt */
35:         obd2 = new Connect();
36:         Log.d("obd2 service", "created");
37:     }
38:
39:     @Override
40:     public void onStart(Intent intent, int startId) {
41:         /** Service wird gestartet, obd2 wird verbunden und ausgelesen */
42:         Log.d("obd2 service", "started");
43:         obd2.start();
44:     }
45:
46:     @Override
47:     public void onDestroy() {
48:         /** Service wird beended obd2 wird beended */
49:         Log.d("obd2 service", "destroyed");
```

```
50:             obd2.close();  
51:         }  
52:  
53:     }
```

```
1: package com.HFUwerdMobil.obd2;
2:
3: public class SocketError extends Exception {
4:
5:     /**
6:      * Diese Klasse ist der Fehler den die Klasse BluetoothStream auslÃ¶sen kann,
7:      * wenn kein gÃ¼ltiger Bluetoothsocket verbunden wird.
8:      *
9:      * @author OBD-Gruppe
10:     * @version 1.0
11:     */
12:    private static final long serialVersionUID = 1L;
13:
14:    public SocketError(String s) {
15:        super(s);
16:    }
17:
18: }
```



```
1: package com.HFUwerdMobil.service;
2:
3: import android.app.Service;
4: import android.content.Context;
5: import android.content.Intent;
6: import android.content.SharedPreferences;
7: import android.location.Criteria;
8: import android.location.Location;
9: import android.location.LocationListener;
10: import android.location.LocationManager;
11: import android.location.LocationProvider;
12: import android.os.Bundle;
13: import android.os.IBinder;
14:
15: import com.HFUwerdMobil.activity.ChooseRouteActivity;
16: import com.HFUwerdMobil.activity.RouteDetailsActivity;
17: import com.HFUwerdMobil.memory.DataAccessHandler;
18: import com.HFUwerdMobil.memory.EntityGPS;
19: import com.HFUwerdMobil.memory.EntityRoute;
20: import com.HFUwerdMobil.sync.DataSyncRoute;
21: import com.HFUwerdMobil.util.IssueNotification;
22:
23: /**
24:  * Diese Klasse trackt die GPS Daten als Service während der Routen
25:  * Aufzeichnung.
26: *
27: * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
28: *         Veronika Kinzel
29: * @version 1.0
30: */
31: public class GPSListener extends Service implements LocationListener {
32:     /**
33:      * Location Provider Objekt
34:      */
35:     private LocationManager lm;
36:
37:     /**
38:      * GPS Provider
39:      */
40:     private String provider;
41:
42:     /**
43:      * DataAccessHandler Objekt
44:      */
45:     private DataAccessHandler sHandler = new DataAccessHandler(this);
46:
47:     /**
48:      * Routen ID
49:      */
```

```
50:         private String routeId;
51:
52:         /**
53:          * routeInit=true bedeutet, dass der Listener beim ersten GPS-Datensatz die
54:          * Route in der Datenbank persistieren soll
55:          */
56:         private boolean routeInit = true;
57:
58:         /**
59:          * Zielort des Benutzers. Standard: nach Hause (Code 0)
60:          */
61:         private int userDestination = 0; // Standard
62:
63:         /**
64:          * Latitude des Zielorts Furtwangen (Code 1)
65:          */
66:         private final double DES_FURTWANGEN_LAT = 48.0511197;
67:
68:         /**
69:          * Longitude des Zielorts Furtwangen (Code 1)
70:          */
71:         private final double DES_FURTWANGEN_LON = 8.208085299999993;
72:
73:         /**
74:          * Latitude des Zielorts Schwenningen (Code 2)
75:          */
76:         private final double DES_SCHWENNINGEN_LAT = 48.06107739999999;
77:
78:         /**
79:          * Longitude des Zielorts Schwenningen (Code 1)
80:          */
81:         private final double DES_SCHWENNINGEN_LON = 8.53560570000002;
82:
83:         /**
84:          * Latitude des Zielorts Tuttlingen (Code 3)
85:          */
86:         private final double DES_TUTTLINGEN_LAT = 47.9826537;
87:
88:         /**
89:          * Longitude des Zielorts Tuttlingen (Code 3)
90:          */
91:         private final double DES_TUTTLINGEN_LON = 8.820721999999932;
92:
93:         /**
94:          * Thread fÃ¼r die Synchronisation der Routendaten
95:          */
96:         Thread routeThread;
97:
98:         /**
```

```
99:             * Editor, um SharedPreferences zu bearbeiten
100:            */
101:        private SharedPreferences.Editor editor;
102:
103:        /**
104:         * Wird beim expliziten Start des Services ausgefÃ¼hrt. Die Datenbank wird
105:         * zum Schreiben geÃ¶ffnet und das anzufahrende Ziel wird ausgelesen
106:         */
107:        public int onStartCommand(Intent intent, int flags, int startId) {
108:            sHandler.openDB();
109:            if (intent.getExtras() != null) {
110:                if (intent.getExtras().containsKey("selected_destination")) {
111:                    userDestination = intent.getInt("selected_destination");
112:                }
113:            }
114:            return START_STICKY;
115:        }
116:
117:        /**
118:         * Wird beim Erzeugen des Services ausgefÃ¼hrt. Es werden Kriterien zur
119:         * Auswahl des Location Providers definiert und nach diesen Kriterien einen
120:         * Provider gewÃ¤hlt, der nun in definiertem Intervallen
121:         * {@link com.HFUwerdMobil.service.GPSListener#onLocationChanged(Location)}
122:         * ausfÃ¼hrt
123:         */
124:        @Override
125:        public void onCreate() {
126:            super.onCreate();
127:            lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
128:
129:            Criteria criteria = new Criteria();
130:            criteria.setAccuracy(Criteria.ACCURACY_FINE);
131:            criteria.setAltitudeRequired(false);
132:            criteria.setBearingRequired(false);
133:            criteria.setCostAllowed(true);
134:            criteria.setPowerRequirement(Criteria.POWER_LOW);
135:            provider = lm.getBestProvider(criteria, true);
136:            if (provider == null) {
137:                IssueNotification.locationProviderNotAvailable(getApplicationContext());
138:                startActivity(new Intent(getApplicationContext(), ChooseRouteActivity.class));
139:                this.stopSelf();
140:            }
141:
142:            /*
143:             * Fragt alle 4 Sekunden die aktuelle GPS Position (Listener) ab, sofern
144:             * die GPS Position sich um mindestens 150 Meter verÃ¤ndert hat
145:             */
146:            lm.requestLocationUpdates(provider, 4000, 150, this);
147:        }
```

```
148:  
149:    /**  
150:     * Startet den Routen Synchronisations Thread und erfasst die aktuellen  
151:     * GPS-Daten die vom Provider geliefert werden. Prüft weiterhin, ob das Ziel  
152:     * erreicht wurde und quittiert dieses dem Nutzer mit entsprechender  
153:     * Meldung.  
154:    */  
155:   @Override  
156:   public void onLocationChanged(Location location) {  
157:       // Anmerkung: Bitte erst entkommentieren, wenn die Backend-Gruppe,  
158:       // die RESTful API vollständig funktionstüchtig hat und der GPS-JSON  
159:       // eingebaut wurde.  
160:       routeThread = new Thread(new DataSyncRoute(getApplicationContext()));  
161:       routeThread.start();  
162:       editor = getApplicationContext().getSharedPreferences("syncData", MODE_PRIVATE).edit();  
163:       editor.putInt("route", -1);  
164:       editor.commit();  
165:  
166:       double lan = location.getLatitude();  
167:       double lon = location.getLongitude();  
168:  
169:       if (routeInit) {  
170:           if (!isDestination(lan, lon)) {  
171:               routeId = sHandler.addRoute(new EntityRoute(Double.toString(lan), Double.toString(lon),  
userDestination, getApplicationContext()));  
172:               SharedPreferences userData = getApplicationContext().getSharedPreferences("userData", M  
ODE_PRIVATE);  
173:               editor = userData.edit();  
174:               editor.putString("route_id", routeId);  
175:               editor.commit();  
176:               if (routeId == null)  
177:                   IssueNotification.fatalError(getApplicationContext());  
178:               routeInit = false;  
179:           } else {  
180:               Intent i = new Intent(getApplicationContext(), RouteDetailsActivity.class);  
181:               i.putExtra("is_destination_yet", true);  
182:               startActivity(i);  
183:               this.stopSelf();  
184:           }  
185:       } else {  
186:           if (sHandler.addGPS(new EntityGPS(routeId, String.valueOf(lan), String.valueOf(lon))) == -1) {  
187:               IssueNotification.fatalError(getApplicationContext());  
188:           }  
189:           if (isDestination(lan, lon)) {  
190:               Intent i = new Intent(getApplicationContext(), RouteDetailsActivity.class);  
191:               i.putExtra("is_destination", true);  
192:               startActivity(i);  
193:               this.stopSelf();  
194:           }  
}
```

```
195:             }
196:         }
197:
198:         /**
199:          * PrÃ¤ft ob Nutzer den Zielort erreicht hat
200:          *
201:          * @param currentLat
202:          *           Latitude der aktuellen Position
203:          * @param currentLon
204:          *           Longitude der aktuellen Position
205:          * @return true, wenn die aktuelle Position der Zielort ist
206:         */
207:     private boolean isDestination(double currentLat, double currentLon) {
208:         if (userDestination == 0) {
209:
210:         }
211:         if (userDestination == 1) {
212:             if (currentLat == DES_FURTWANGEN_LAT && currentLon == DES_FURTWANGEN_LON)
213:                 return true;
214:             else
215:                 return false;
216:         }
217:         if (userDestination == 2) {
218:             if (currentLat == DES_SCHWENNINGEN_LAT && currentLon == DES_SCHWENNINGEN_LON)
219:                 return true;
220:             else
221:                 return false;
222:         }
223:         if (userDestination == 3) {
224:             if (currentLat == DES_TUTTLINGEN_LAT && currentLon == DES_TUTTLINGEN_LON)
225:                 return true;
226:             else
227:                 return false;
228:         }
229:         return false;
230:     }
231:
232:     /**
233:      * Wenn der Service zerstÃ¤rt wird, werden die Providerupdates entfernt und
234:      * der Routen Synchrisionsthread interrupted. AuÃ\237erdem wird die Datenbank
235:      * geschloÃ\237en.
236:      */
237:     @Override
238:     public void onDestroy() {
239:         super.onDestroy();
240:         lm.removeUpdates(this);
241:         if (routeThread != null) {
242:             routeThread.interrupt();
243:         }
244:     }
```

```
244:             sHandler.closeDB();
245:         }
246:
247:         /**
248:          * Wird aufgerufen, wenn sich der Status des Location Providers verÄndert
249:          * hat und gibt entsprechende Meldung an den Nutzer aus, wenn der Provider
250:          * nicht weiter verfÄigbar ist.
251:          */
252:         @Override
253:         public void onStatusChanged(String provider, int status, Bundle extras) {
254:             if (status == LocationProvider.OUT_OF_SERVICE) {
255:                 IssueNotification.locationProviderNotAvailable(getApplicationContext());
256:                 startActivity(new Intent(getApplicationContext(), RouteDetailsActivity.class));
257:                 this.stopSelf();
258:             }
259:         }
260:
261:         /**
262:          * Wird aufgerufen, wenn ein Location Provider aktiviert wird
263:          */
264:         @Override
265:         public void onProviderEnabled(String provider) {
266:         }
267:
268:         /**
269:          * Wird aufgerufen, wenn der Benutzer den Location Provider deaktiviert hat
270:          * und gibt somit entsprechende Meldung an den Nutzer aus und beendet
271:          * auÃ237erdem die Routenaufzeichnung
272:          */
273:         @Override
274:         public void onProviderDisabled(String provider) {
275:             IssueNotification.locationProviderNotAvailable(getApplicationContext());
276:             startActivity(new Intent(getApplicationContext(), RouteDetailsActivity.class));
277:             this.stopSelf();
278:         }
279:
280:         @Override
281:         public IBinder onBind(Intent intent) {
282:             return null;
283:         }
284:     }
```

```
1: package com.HFUwerdMobil.sync;
2:
3: import java.io.BufferedReader;
4: import java.io.IOException;
5: import java.io.InputStreamReader;
6: import java.io.OutputStream;
7: import java.net.HttpURLConnection;
8: import java.net.URL;
9: import java.security.MessageDigest;
10: import java.security.NoSuchAlgorithmException;
11:
12: import android.content.Context;
13: import android.net.ConnectivityManager;
14: import android.net.NetworkInfo;
15:
16: import com.HFUwerdMobil.activity.LoginActivity;
17:
18: /**
19:  * Diese Klasse handelt die Synchronisation zu externen Services (RESTful API).
20:  * Sie stellt die entsprechenden Methoden zur VerfÃ¼gung um einfach Daten an die
21:  * Services zu senden, bzw. Daten von externen Services zu empfangen.
22:  *
23:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
24:  *         Veronika Kinzel
25:  * @version 1.0
26: */
27: public class DataSyncHandler {
28:     /**
29:      * EnthÃ¤lt die "Home"-URL des REST-Services
30:      */
31:     private static final String BASE_REST_URL = "http://api.hfuwerdmobil.de:80";
32:
33:     /**
34:      * EnthÃ¤lt den "geheimen" API-Key zur Sicherung der DatenÃ¼bertragung. Wird
35:      * verwendet um ein Token fÃ¼r eine sichere DatenÃ¼bertragung zu generieren.
36:      */
37:     private static final String API_KEY = "/u8teH}@A}__&sQ%3DUusbW/m88Wk,K";
38:
39:     /**
40:      * E-Mail Adresse des Benutzers, die fÃ¼r die meisten API-Endpunkte benÃ¶tigt
41:      * wird zur eindeutigen Identifikation
42:      */
43:     private String userEmail;
44:
45:     /**
46:      * EmpfÃ¤ngt und speichert die Benutzeremail zwischen
47:      *
48:      * @param userEmail
49:      *          Benutzer E-Mail Adresse
```

```
50:         */
51:     public DataSyncHandler(String userEmail) {
52:         this.userEmail = userEmail;
53:     }
54:
55:    /**
56:     * Senden einen GET-Request mit vorheriger Sicherheitstoken Generierung an
57:     * den Ã¼bergebenen Endpunkt der RESTful API. Gibt anschlieÃ\x237end den
58:     * Response-Code und die JSON-RÃ\xackgabedaten zurÃ\x23ck.
59:     *
60:     * @param endpoint
61:     *          API-Endpunkt (ohne "Home"-URL und ohne Sicherheitstoken) von
62:     *          welchem Daten empfangen werden sollen
63:     * @return String Array mit dem Response Code an Index 0 und dem
64:     *          RÃ\xackgabe-JSON an Index 1, oder null bei nicht bestehender
65:     *          Internetverbindung oder Unerreichen des Servers
66:     */
67:    protected String[] receiveData(String endpoint) {
68:        if (hasInternetCon()) {
69:            HttpURLConnection con = null;
70:            try {
71:                con = (HttpURLConnection) new URL(BASE_REST_URL + endpoint + "?token=" + createToken(en
dpoint, "")).openConnection();
72:                con.setRequestMethod("GET");
73:                con.setRequestProperty("Accept", "application/json");
74:
75:                BufferedReader br = new BufferedReader(new InputStreamReader(con.getInputStream()));
76:                String data = "";
77:                String dataTmp;
78:                while ((dataTmp = br.readLine()) != null) {
79:                    data += dataTmp;
80:                }
81:                return new String[] { Integer.toString(con.getResponseCode()), data };
82:            } catch (IOException e) {
83:                return null;
84:            } catch (NullPointerException e) {
85:                return null;
86:            } finally {
87:                con.disconnect();
88:            }
89:        }
90:        return null;
91:    }
92:
93:    /**
94:     * Senden ein Ã¼bergebenes JSON-Objekt als POST- oder PUT-Request mit
95:     * Sicherheitstoken an den Ã¼bergebenen Endpunkt der RESTful API. Gibt
96:     * anschlieÃ\x237end den Response-Code zurÃ\x23ck.
97:     *
```

```
98:         * @param endpoint
99:         *          API-Endpunkt (ohne "Home"-URL und ohne Sicherheitstoken) an
100:        *          welchen die Daten gesendet werden sollen
101:       * @param httpMethod
102:       *          HTTP-Methode zum Senden der Daten (POST oder PUT)
103:       * @param dataOut
104:       *          JSON-Objekt, ohne fÃ¼hrende und endende geschweifte Klammern
105:       *          (da Sicherheitstoken noch angehÃ¤ngt wird)
106:       * @return Response-Code, oder -1 bei nicht bestehender Internetverbindung
107:       *          oder Unerreichen des Servers
108:      */
109:     protected int sendData(String endpoint, String httpMethod, String dataOut) {
110:       dataOut = "{\"" + dataOut + createToken(endpoint, dataOut) + "\"}";
111:       if (hasInternetCon()) {
112:         HttpURLConnection con = null;
113:         try {
114:           con = (HttpURLConnection) new URL(BASE_REST_URL + endpoint).openConnection();
115:           con.setDoOutput(true);
116:           con.setRequestMethod(httpMethod);
117:           con.setRequestProperty("Content-Type", "application/json");
118:
119:           OutputStream outStream = con.getOutputStream();
120:           outStream.write(dataOut.getBytes());
121:           outStream.flush();
122:
123:           return con.getResponseCode();
124:         } catch (NullPointerException e) {
125:           return -1;
126:         } catch (IOException e) {
127:           return -1;
128:         } finally {
129:           con.disconnect();
130:         }
131:       }
132:       return -1;
133:     }
134:
135:   /**
136:    * Generiert das Sicherheitstoken, welcher bei jedem Request von oder zur
137:    * RESTful API generiert und mit gesendet wird, und durch gleiches
138:    * Berechnungsverfahren auf Serverseite auf Korrektheit Ã¼berprÃ¼ft wird
139:    *
140:    * @param endpoint
141:    *          API-Endpunkt (ohne "Home"-URL)
142:    * @param json
143:    *          JSON-Objekt, ohne fÃ¼hrende und endende geschweifte Klammern
144:    * @return Sicherheitstoken, oder null falls die Generierung fehlgeschlug
145:    */
146:   protected String createToken(String endpoint, String json) {
```

```
147:             String tokenData = userEmail + API_KEY + endpoint + json;
148:
149:         try {
150:             MessageDigest md = MessageDigest.getInstance("SHA-256");
151:             md.update(tokenData.getBytes());
152:             byte byteData[] = md.digest();
153:
154:             StringBuffer sb = new StringBuffer();
155:             for (int i = 0; i < byteData.length; i++) {
156:                 sb.append(Integer.toString((byteData[i] & 0xff) + 0x100, 16).substring(1));
157:             }
158:
159:             return sb.toString();
160:         } catch (NoSuchAlgorithmException e) {
161:             return null;
162:         }
163:     }
164:
165:     /**
166:      * PrÃ¼ft, ob das Android Device derzeit Zugang zum Internet hat
167:      *
168:      * @return true, wenn eine Internetverbindung besteht
169:      */
170:     private boolean hasInternetCon() {
171:         ConnectivityManager cm = (ConnectivityManager) LoginActivity.getLoginActivityContext().getSystemService
(Context.CONNECTIVITY_SERVICE);
172:         NetworkInfo ni = cm.getActiveNetworkInfo();
173:         return ni != null && ni.isConnectedOrConnecting();
174:     }
175: }
```

```
1: package com.HFUwerdMobil.sync;
2:
3: import android.app.Activity;
4: import android.content.Context;
5: import android.content.SharedPreferences;
6:
7: import com.HFUwerdMobil.memory.DataAccessHandler;
8: import com.HFUwerdMobil.memory.EntityRoute;
9: import com.HFUwerdMobil.memory.UserData;
10:
11: /**
12:  * Synchronisiert die Routen Daten mit externen Services (RESTful API)
13:  *
14:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
15:  *         Veronika Kinzel
16:  * @version 1.0
17: */
18: public class DataSyncRoute extends Activity implements Runnable {
19:     /**
20:      * Beinhaltet den Endpunkt fÃ¼r Nutzer-Aktionen an oder von der RESTful-API
21:      */
22:     private static final String ENDPOINT_USER = "/user/";
23:
24:     /**
25:      * Beinhaltet den Endpunkt fÃ¼r Routen-Aktionen an oder von der RESTful-API
26:      */
27:     private static final String ENDPOINT_ROUTE = "/route/";
28:
29:     /**
30:      * Beinhaltet die Zeit in Millisekunden, die bei einer fehlerhaften
31:      * Ã234bertragung an oder von der RESTful-API gewartet werden soll
32:      */
33:     private static final int BREAK_TIME = 120000; // 2 Minuten
34:
35:     /**
36:      * Beinhaltet die Zeit in Millisekunden, die mit der Ã234bertragung an oder von
37:      * der RESTful-API gewartet werden soll, wenn der Provider den nÃ¤chsten
38:      * GPS-Datensatz noch nicht erfasst hat
39:      */
40:     private static final int BREAK_TIME_GPS_REQUEST = 4000; // 4 Sekunden
41:
42:     /**
43:      * Synchronisations-Flags
44:      */
45:     private SharedPreferences syncData;
46:
47:     /**
48:      * DataAccessHandler
49:      */
```

```
50:         private DataAccessHandler sHandler;
51:
52:         /**
53:          * DataSyncHandler
54:          */
55:         private DataSyncHandler syncHandler;
56:
57:         /**
58:          * Editor, um SharedPreferences zu bearbeiten
59:          */
60:         private SharedPreferences.Editor editor;
61:
62:         /**
63:          * Nutzer-Daten Objekt
64:          */
65:         private UserData userData;
66:
67:         /**
68:          * Sync-User Objekt
69:          */
70:         private DataSyncUser syncUser;
71:
72:         /**
73:          * Ãœbernet die syncData und syncHandler Objekte
74:          *
75:          * @param context
76:          *          Context Objekt
77:          */
78:         public DataSyncRoute(Context context) {
79:             syncData = context.getSharedPreferences("syncData", MODE_PRIVATE);
80:             syncHandler = new DataSyncHandler(new UserData(context).getUserMail());
81:         }
82:
83:         /**
84:          * Synchronisiert die Routen Daten mit den externen Services (RESTful API),
85:          * dabei werden auch noch-nicht synchronisierte Daten, synchronisiert.
86:          */
87:         @Override
88:         public void run() {
89:             int routePointer = syncData.getInt("route_pointer", 1);
90:             int gpsPointer = syncData.getInt("gps_pointer", 1);
91:             editor = syncData.edit();
92:
93:             do {
94:                 if (startRoute(sHandler.getRoute(routePointer)) == 424) {
95:                     syncUser.newUser();
96:                     startRoute(sHandler.getRoute(routePointer));
97:                 }
98:             while (!Thread.interrupted()) {
```

```

./com/HFUwerdMobil/sync/DataSyncRoute.java      Wed Jan 21 12:23:49 2015      3

99:                               if (sHandler.getLastAddedGpsId() >= gpsPointer) {
100:                                 // GPS-Daten senden (siehe Spezifikation)
101:                                 // Noch nicht implementiert, da Backend-Gruppe diesen
102:                                 // Endpunkt
103:                                 // noch nicht realisiert hat
104:                               } else {
105:                                 try {
106:                                   Thread.sleep(BREAK_TIME_GPS_REQUEST);
107:                                 } catch (InterruptedException e) {
108:                                   // Keine Behandlung nötig
109:                                 }
110:                               }
111:                               editor.putInt("gps_pointer", ++gpsPointer);
112:                               editor.commit();
113:                           }
114:                           if (endRoute(sHandler.getRoute(routePointer)) != 424) {
115:                             editor.putInt("route_pointer", ++routePointer);
116:                             editor.putInt("gps_pointer", 1);
117:                             editor.commit();
118:                           }
119:                           } while (routePointer <= sHandler.getLastAddedRouteId());
120:                           editor.putInt("route", 1);
121:                           editor.commit();
122:                         }
123:
124: /**
125: * Änderungsbericht das JSON-Objekt, welches Daten vom Routen Start enthält, an die
126: * externen Services (RESTful API)
127: *
128: * @param routeData
129: *   Objekt, der aktuellen Route
130: * @return HTTP-Rückgabecode, oder -1 falls der Thread unterbrochen wurde
131: */
132: private int startRoute(EntityRoute routeData) {
133:   int responseCode = syncHandler.sendData(ENDPOINT_USER + userData.getUserMail() + ENDPOINT_ROUTE + "new"
, "POST",
134:   routeData.toJsonForRouteStart());
135:   while (responseCode != 200 && responseCode != 409 && responseCode != 424) {
136:     try {
137:       Thread.sleep(BREAK_TIME);
138:     } catch (InterruptedException e) {
139:       return -1;
140:     }
141:     responseCode = syncHandler.sendData(ENDPOINT_USER + userData.getUserMail() + ENDPOINT_ROUTE + "
new", "POST",
142:   routeData.toJsonForRouteStart());
143:   }
144:   return responseCode;
145: }
```

```
146:  
147:     /**  
148:      * ÃœbertrÃ¤gt das JSON-Objekt, welches Daten vom Routen Ende enthÃ¤lt, an die  
149:      * externen Services (RESTful API)  
150:      *  
151:      * @param routeData  
152:      * Objekt, der aktuellen Route  
153:      * @return HTTP-RÃ¼ckgabecode, oder -1 falls der Thread unterbrochen wurde  
154:      */  
155:     private int endRoute(EntityRoute routeData) {  
156:         int responseCode = syncHandler.sendData(ENDPOINT_USER + userData.getUserMail() + ENDPOINT_ROUTE + "end"  
,"POST",  
157:                         routeData.getJsonForRouteEnd());  
158:         while (responseCode != 200 && responseCode != 409 && responseCode != 424) {  
159:             try {  
160:                 Thread.sleep(BREAK_TIME);  
161:             } catch (InterruptedException e) {  
162:                 return -1;  
163:             }  
164:             responseCode = syncHandler.sendData(ENDPOINT_USER + userData.getUserMail() + ENDPOINT_ROUTE + "  
end", "POST",  
165:                         routeData.getJsonForRouteEnd());  
166:         }  
167:         return responseCode;  
168:     }  
169: }
```

```
1: package com.HFUwerdMobil.sync;
2:
3: import android.app.Activity;
4: import android.content.Context;
5: import android.content.SharedPreferences;
6:
7: import com.HFUwerdMobil.memory.UserData;
8:
9: /**
10:  * Synchronisiert die User Daten mit externen Services (RESTful API)
11:  *
12:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
13:  *         Veronika Kinzel
14:  * @version 1.0
15: */
16: public class DataSyncUser extends Activity implements Runnable {
17:     /**
18:      * Beinhaltet den Endpunkt fÃ¼r Nutzer-Aktionen an oder von der RESTful-API
19:      */
20:     private static final String ENDPOINT_USER = "/user/";
21:
22:     /**
23:      * Beinhaltet die Zeit in Millisekunden, die bei einer fehlerhaften
24:      * Ã234bertragung an oder von der RESTful-API gewartet werden soll
25:      */
26:     private static final int BREAK_TIME = 600000; // 10 Minuten
27:
28:     /** Speicher Objekt fÃ¼r Setting Flags fÃ¼r die Datensynchronisation **/
29:     private SharedPreferences syncData;
30:
31:     /** Speicher Objekt der User Daten **/
32:     private SharedPreferences userData;
33:
34:     /**
35:      * {@link com.HFUwerdMobil.sync.DataSyncHandler} Objekt
36:      */
37:     private DataSyncHandler syncHandler;
38:
39:     /**
40:      * Editor Objekt um die Daten der Speicher Objekte zu bearbeiten
41:      */
42:     private SharedPreferences.Editor editor;
43:
44:     /**
45:      * Beinhaltet das JSON-Objekt, ohne fÃ¼hrende und endende geschweifte
46:      * Klammern
47:      */
48:     private String jsonOut;
49:
```

```
50:         /**
51:          * Ä\226ffnet die Speicher Objekte, legt den Sync Handler (
52:          * {@link com.HFUwerdMobil.sync.DataSyncHandler}) an und legt das
53:          * JSON-Objekt fest
54:          *
55:          * @param context
56:          *      {@link android.content.Context} Objekt
57:          * @param jsonOut
58:          *      JSON-Objekt ohne fÄ\4hrende und endende geschweifte Klammern
59:          */
60:     public DataSyncUser(Context context, String jsonOut) {
61:         syncData = context.getSharedPreferences("syncData", MODE_PRIVATE);
62:         userData = context.getSharedPreferences("userData", MODE_PRIVATE);
63:         syncHandler = new DataSyncHandler(new UserData(context).getUserMail());
64:         this.jsonOut = jsonOut;
65:     }
66:
67:     /**
68:      * Synchronisiert einen neuen Benutzer oder Ä\204nderung der Benutzerdaten.
69:      * Meldet die API bei der Ä\204nderung der Benutzerdaten, dass der Benutzer auf
70:      * dem Server nicht existiert, so wird der Nutzer mit den aktualisierten
71:      * Daten an die API Ä\4bertragen.
72:      */
73:     public void run() {
74:         int userFlag = syncData.getInt("user", -1);
75:         if (userFlag == -1) {
76:             newUser();
77:             updatePref();
78:         } else if (userFlag == 0) {
79:             if (updateUser() == 424) {
80:                 newUser();
81:             }
82:             updatePref();
83:         }
84:     }
85:
86:     /**
87:      * Schickt einen neuen Benutzer an die RESTful-API, und wartet auf einen
88:      * Response Code. Bei fehlgeschlagener Ä\234bertragung wird der Request nach
89:      * Ablauf einer Unterbrechungszeit erneut gesendet
90:      *
91:      * @return Response-Code, oder -1 falls die Synchronisation aufgrund eines
92:      *         Abbruch des Threads (bspw. Nutzer beendet App) nicht beendet
93:      *         werden konnte.
94:      */
95:     synchronized protected int newUser() {
96:         int responseCode = syncHandler.sendData(ENDPOINT_USER + "new", "POST", jsonOut);
97:         while (responseCode != 200 && responseCode != 409) {
98:             try {
```

```
99:                     Thread.sleep(BREAK_TIME);
100:                } catch (InterruptedException e) {
101:                    return -1;
102:                }
103:                responseCode = syncHandler.sendData(ENDPOINT_USER + "new", "POST", jsonOut);
104:            }
105:            return responseCode;
106:        }
107:
108:        /**
109:         * Updated einen Benutzer gegenÃ¼ber der RESTful-API, und wartet auf einen
110:         * Response Code. Bei fehlgeschlagener Ã\234bertragung wird der Request nach
111:         * Ablauf einer Unterbrechungszeit erneut gesendet
112:         *
113:         * @return Response-Code, oder -1 falls die Synchronisation aufgrund eines
114:         *         Abbruch des Threads (bspw. Nutzer beendet App) nicht beendet
115:         *         werden konnte.
116:         */
117:        private int updateUser() {
118:            int responseCode = syncHandler.sendData(ENDPOINT_USER + getUserIdentification(), "POST", jsonOut);
119:            while (responseCode != 200 && responseCode != 424) {
120:                try {
121:                    Thread.sleep(BREAK_TIME);
122:                } catch (InterruptedException e) {
123:                    return -1;
124:                }
125:                responseCode = syncHandler.sendData(ENDPOINT_USER + getUserIdentification(), "POST", jsonOut);
126:            }
127:            return responseCode;
128:        }
129:
130:        /**
131:         * Updated die Synchronisations-Flags auf "Erfolgreich synchronisiert"
132:         */
133:        private void updatePref() {
134:            editor = syncData.edit();
135:            editor.putInt("user", 1);
136:            editor.commit();
137:        }
138:
139:        /**
140:         * Gibt die E-Mail Adresse zur Autorisierung des Nutzers bei der RESTful-API
141:         * zurÃ¼ck. Hat der Nutzer seine E-Mail Adresse geÄndert, muss bei der
142:         * Ã\204nderung gegenÃ¼ber der RESTful-API einmalig die alte E-Mail zur
143:         * Autorisierung mitgegeben werden.
144:         *
145:         * @return E-Mail zur Autorisierung der RESTful-API, nach Ã\204nderung der
146:         *         Benutzerdaten
147:         */
```

```
148:     private String getUserIdentification() {
149:         String userEmail = userData.getString("old_user_email", null);
150:         if (userEmail != null) {
151:             editor = userData.edit();
152:             editor.remove("old_user_email");
153:             editor.commit();
154:             return userEmail;
155:         } else {
156:             return userData.getString("user_email", null);
157:         }
158:     }
159: }
```

```
1: package com.HFUwerdMobil.sync;
2:
3: import android.app.Activity;
4: import android.content.Context;
5:
6: import com.HFUwerdMobil.memory.UserData;
7:
8: /**
9:  * Fordert fÃ¼r eine unverifizierte E-Mail Adresse bei den externen Services
10: * (RESTful API) eine Verifizierungscode an
11: *
12: * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
13: *         Veronika Kinzel
14: * @version 1.0
15: */
16: public class DataSyncVerifycode extends Activity implements Runnable {
17:     /**
18:      * Unverifizierte E-Mail Adresse des Benutzers
19:      */
20:     private String unverifiedEmail;
21:
22:     /**
23:      * Aktuell hinterlegte E-Mail Adresse des Benutzers
24:      */
25:     private String userEmail;
26:
27:     /**
28:      * Von der API zurÃ¼ckgegebener Verifizierungscode
29:      */
30:     public static String[] VERIFYCODE_RESPONSE = null;
31:
32:     /**
33:      * Speichert die unverifizierte E-Mail Adresse des Benutzers und das Context
34:      * Objekt zwischen
35:      *
36:      * @param unverifiedEmail
37:      *          Unverifizierte E-Mail Adresse des Benutzers
38:      * @param context
39:      *          Context Objekt
40:      */
41:     public DataSyncVerifycode(String unverifiedEmail, Context context) {
42:         this.unverifiedEmail = unverifiedEmail;
43:         this.userEmail = new UserData(context).getUserMail();
44:     }
45:
46:     /**
47:      * Schickt eine zu verifizierende E-Mail Adresse an die RESTful-API, und
48:      * wartet auf einen Response Code und den zugehÃ¶rigen Verifizierungscode.
49:      */
```

```
./com/HFUwerdMobil/sync/DataSyncVerifycode.java      Tue Jan 20 22:35:15 2015      2
50:          * @return String Array mit Response Code (Index 0) und Verifizierungscode
51:          *           (Index 1)
52:          */
53:         @Override
54:         public void run() {
55:             VERIFYCODE_RESPONSE = new DataSyncHandler(userEmail).receiveData("/verify/" + unverifiedEmail);
56:         }
57:     }
```

```
1: package com.HFUwerdMobil.util;
2:
3: import java.io.IOException;
4: import java.io.InputStream;
5:
6: import org.apache.http.client.ClientProtocolException;
7: import org.apache.http.client.methods.HttpGet;
8: import org.apache.http.impl.client.DefaultHttpClient;
9: import org.json.JSONException;
10: import org.json.JSONObject;
11:
12: import android.os.AsyncTask;
13:
14: /**
15:  * Diese Klasse codiert Postleitzahlen oder GPS Daten (Latitude, Longitude) zu
16:  * Ortsnamen
17:  *
18:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
19:  *         Veronika Kinzel
20:  * @version 1.0
21:  */
22: public class Geocoding extends AsyncTask<Void, Void, String> {
23:     /**
24:      * Geocoding API URI
25:      */
26:     String uri = null;
27:
28:     /**
29:      * Generiert die API URI, von der die kodierten Daten gelesen werden sollen
30:      *
31:      * @param lat
32:      *        Latitude
33:      * @param lng
34:      *        Longitude
35:      */
36:     public Geocoding(String lat, String lng) {
37:         this.uri = "http://maps.googleapis.com/maps/api/geocode/json?latlng=" + lat + "," + lng + "&sensor=false";
38:     }
39:
40:     /**
41:      * Generiert die API URI, von der die kodierten Daten gelesen werden sollen
42:      *
43:      * @param zipCode
44:      *        Postleitzahl
45:      */
46:     public Geocoding(String zipCode) {
47:         this.uri = "http://maps.googleapis.com/maps/api/geocode/json?address=" + zipCode + "&region=de&sensor=false";
48:     }
49:
```

```
48:         }
49:
50:        /**
51:         * Liest anhand der generierten URI, das augerufene Dokument aus und erhält
52:         * somit den kodierten Ortsnamen und gibt diesen zurück an den Task Aufrufer
53:         */
54:        @Override
55:        protected String doInBackground(Void... params) {
56:            try {
57:                StringBuilder stringBuilder = new StringBuilder();
58:                InputStream stream = new DefaultHttpClient().execute(new HttpGet(uri)).getEntity().getContent();
59:
60:                int b;
61:                while ((b = stream.read()) != -1) {
62:                    stringBuilder.append((char) b);
63:                }
64:                JSONObject jsonObject = new JSONObject(stringBuilder.toString());
65:                try {
66:                    int i = 0;
67:                    do {
68:                        if (jsonObject.getJSONArray("results").getJSONObject(0).getJSONArray("address_components").getJSONObject(i).getString("types")
69:                            .equals("[\"locality\", \"political\"]")) {
70:                                return new String(jsonObject.getJSONArray("results").getJSONObject(0).g
etJSONArray("address_components").getJSONObject(i)
71:                                    .getString("long_name")).getBytes("ISO-8859-1"), "UTF-8"
72:                            };
73:                            i++;
74:                        } while (true);
75:                    } catch (JSONException e) {
76:                        e.printStackTrace();
77:                    }
78:                } catch (ClientProtocolException e) {
79:                    e.printStackTrace();
80:                } catch (IOException e) {
81:                    e.printStackTrace();
82:                } catch (JSONException e) {
83:                    e.printStackTrace();
84:                }
85:            }
86:        }
87:    }
```

```
1: package com.HFUwerdMobil.util;
2:
3: import android.app.AlertDialog;
4: import android.app.DialogFragment;
5: import android.content.Context;
6: import android.content.DialogInterface;
7: import android.content.SharedPreferences;
8:
9: import com.HFUwerdMobil.main.R;
10:
11: /**
12:  * Diese Klasse stellt Methoden bereit, um dem Nutzer Informationen über
13:  * Probleme und schwerwiegende Fehler in der App zu informieren.
14:  *
15:  * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
16:  *         Veronika Kinzel
17:  * @version 1.0
18:  */
19: public class IssueNotification extends DialogFragment {
20:     /** {@link android.content.Context} Objekt **/
21:     static Context context;
22:
23:     /** Neue E-Mail Adresse des Google Konto (Primärkonto) des Android Device **/
24:     static String email;
25:
26:     /**
27:      * Informiert den Nutzer per Dialog, dass ein schwerwiegender Fehler
28:      * aufgetreten ist. Beim Bestätigen des Dialoges wird der
29:      * Applikations-Prozess gekillt.
30:      *
31:      * @param context
32:      *          {@link android.content.Context} Objekt
33:      */
34:     public static void fatalError(Context context) {
35:         AlertDialog.Builder builder = new AlertDialog.Builder(context);
36:         builder.setTitle(R.string.dialog_fatalError_title);
37:         builder.setMessage(R.string.dialog_fatalError_message);
38:         builder.setNegativeButton(R.string.dialog_fatalError_terminate, new DialogInterface.OnClickListener() {
39:
40:             @Override
41:             public void onClick(DialogInterface dialog, int which) {
42:                 dialog.cancel();
43:
44:                     // Kill App
45:                     android.os.Process.killProcess(android.os.Process.myPid());
46:                 }
47:             });
48:             builder.create().show();
49:     }
}
```

```
50:
51:        /**
52:         * Informiert den Nutzer per Dialog, dass die E-Mail Adresse im PrimÄarkonto
53:         * des Android Device geÄndert wurde. Weiterhin soll der Nutzer angeben, ob
54:         * die geÄnderte E-Mail Adresse als neue user_email gesetzt werden soll oder
55:         * ob die alte beibehalten werden soll.
56:         *
57:         * @param context
58:             {@link android.content.Context} Objekt
59:         * @param email
60:             Die neue E-Mail Adresse des Android Device PrimÄarkonto
61:         */
62:     public static void primaryMailChanged(Context context, String email) {
63:         IssueNotification.context = context;
64:         IssueNotification.email = email;
65:
66:         AlertDialog.Builder builder = new AlertDialog.Builder(context);
67:         builder.setTitle(R.string.dialog_primaryMailChanged_title);
68:         builder.setMessage(R.string.dialog_primaryMailChanged_message);
69:         builder.setPositiveButton(R.string.dialog_primaryMailChanged_yes, new DialogInterface.OnClickListener()
{
70:
71:             @Override
72:             public void onClick(DialogInterface dialog, int which) {
73:                 dialog.cancel();
74:                 SharedPreferences userData = IssueNotification.context.getSharedPreferences("userData",
Context.MODE_PRIVATE);
75:                 SharedPreferences.Editor editor = userData.edit();
76:                 editor.putString("user_email", IssueNotification.email);
77:                 editor.commit();
78:                 IssueNotification.primaryMailChangedSuccessfully();
79:             }
80:         });
81:         builder.setNegativeButton(R.string.dialog_primaryMailChanged_no, new DialogInterface.OnClickListener()
{
82:
83:             @Override
84:             public void onClick(DialogInterface dialog, int which) {
85:                 dialog.cancel();
86:                 SharedPreferences userData = IssueNotification.context.getSharedPreferences("userData",
Context.MODE_PRIVATE);
87:                 SharedPreferences.Editor editor = userData.edit();
88:                 editor.putString("user_email_lastQueried", IssueNotification.email);
89:                 editor.commit();
90:             }
91:         });
92:         builder.create().show();
93:     }
94:
```

```
95:         /**
96:          * Informiert den Nutzer per Dialog, dass der eingegebene Verifizierungscode
97:          * nicht korrekt ist
98:          *
99:          * @param context
100:             Context Objekt
101:        */
102:       public static void verifycodeInputFailed(Context context) {
103:           IssueNotification.context = context;
104:
105:           AlertDialog.Builder builder = new AlertDialog.Builder(context);
106:           builder.setTitle(R.string.dialog_verifycodeInputFailed_title);
107:           builder.setMessage(R.string.dialog_verifycodeInputFailed_message);
108:           builder.setPositiveButton(R.string.dialog_verifycodeInputFailed_okay, new DialogInterface.OnClickListener() {
109:
110:               @Override
111:               public void onClick(DialogInterface dialog, int which) {
112:                   dialog.cancel();
113:               }
114:           });
115:           builder.create().show();
116:       }
117:
118:       /**
119:          * Informiert den Nutzer per Dialog, dass die die App-E-Mail-Adresse durch
120:          * die E-Mail Adresse des PrimÄärkontos des Android Device substituiert
121:          * wurde.
122:          */
123:       private static void primaryMailChangedSuccessfully() {
124:           AlertDialog.Builder builder = new AlertDialog.Builder(context);
125:           builder.setTitle(R.string.dialog_primaryMailChangedSuccessfully_title);
126:           builder.setMessage(R.string.dialog_primaryMailChangedSuccessfully_message);
127:           builder.setPositiveButton(R.string.dialog_primaryMailChangedSuccessfully_okay, new DialogInterface.OnClickListener() {
ickListener() {
128:
129:               @Override
130:               public void onClick(DialogInterface dialog, int which) {
131:                   dialog.cancel();
132:               }
133:           });
134:           builder.create().show();
135:       }
136:
137:       /**
138:          * Informiert den Nutzer per Dialog, dass die Route nicht weiter
139:          * aufgezeichnet wird.
140:          *
141:          * @param context
```

```
142:             * {@link android.content.Context} Objekt
143:             */
144:         public static void locationProviderNotAvailable(Context context) {
145:             AlertDialog.Builder builder = new AlertDialog.Builder(context);
146:             builder.setTitle(R.string.dialog_locationProviderNotAvailable_title);
147:             builder.setMessage(R.string.dialog_locationProviderNotAvailable_message);
148:             builder.setNegativeButton(R.string.dialog_locationProviderNotAvailable_okay, new DialogInterface.OnClickListener() {
149:                 @Override
150:                 public void onClick(DialogInterface dialog, int which) {
151:                     dialog.cancel();
152:                 }
153:             });
154:             builder.create().show();
155:         }
156:     }
157:
158: /**
159: * Informiert den Nutzer per Dialog, dass die Verbindung zum Netzwerk
160: * fehlerhaft ist.
161: *
162: * @param context
163: *          Context Objekt
164: */
165: public static void networkFailed(Context context) {
166:     AlertDialog.Builder builder = new AlertDialog.Builder(context);
167:     builder.setTitle(R.string.dialog_networkFailed_title);
168:     builder.setMessage(R.string.dialog_networkFailed_message);
169:     builder.setNegativeButton(R.string.dialog_networkFailed_okay, new DialogInterface.OnClickListener() {
170:         @Override
171:         public void onClick(DialogInterface dialog, int which) {
172:             dialog.cancel();
173:             // Kill App
174:             android.os.Process.killProcess(android.os.Process.myPid());
175:         }
176:     });
177:     builder.create().show();
178: }
179:
180:
181: /**
182: * Informiert den Nutzer per Dialog, dass die vom Nutzer eingegebene E-Mail
183: * nicht korrekt ist.
184: *
185: * @param context
186: *          {@link android.content.Context} Objekt
187: */
188: public static void emailNotValid(Context context) {
189:     AlertDialog.Builder builder = new AlertDialog.Builder(context);
```

```
190:         builder.setTitle(R.string.dialog_emailNotValid_title);
191:         builder.setMessage(R.string.dialog_emailNotValid_message);
192:         builder.setNegativeButton(R.string.dialog_emailNotValid_okay, new DialogInterface.OnClickListener() {
193:
194:             @Override
195:             public void onClick(DialogInterface dialog, int which) {
196:                 dialog.cancel();
197:             }
198:         });
199:         builder.create().show();
200:     }
201:
202: /**
203: * Informiert den Nutzer per Dialog, dass die vom Nutzer eingegebene
204: * Handynummer nicht korrekt ist.
205: *
206: * @param context
207: *          {@link android.content.Context} Objekt
208: */
209: public static void deviceNumberNotValid(Context context) {
210:     AlertDialog.Builder builder = new AlertDialog.Builder(context);
211:     builder.setTitle(R.string.dialog_deviceNumberNotValid_title);
212:     builder.setMessage(R.string.dialog_deviceNumberNotValid_message);
213:     builder.setNegativeButton(R.string.dialog_deviceNumberNotValid_okay, new DialogInterface.OnClickListener() {
214:
215:         @Override
216:         public void onClick(DialogInterface dialog, int which) {
217:             dialog.cancel();
218:         }
219:     });
220:     builder.create().show();
221: }
222: }
```



```
1: package com.HFUwerdMobil.util;
2:
3: import android.content.Context;
4: import android.util.AttributeSet;
5: import android.widget.ImageView;
6:
7: /**
8:  * Diese Klasse handelt das About Grid.
9:  *
10: * @author Tobias Straub, Susanne Roos, Matthias Feichtmeier, Thomas Lesinski,
11: *         Veronika Kinzel
12: * @version 1.0
13: */
14: public class SquareImageView extends ImageView {
15:
16:     public SquareImageView(Context context) {
17:         super(context);
18:     }
19:
20:     public SquareImageView(Context context, AttributeSet attrs) {
21:         super(context, attrs);
22:     }
23:
24:     public SquareImageView(Context context, AttributeSet attrs, int defStyle) {
25:         super(context, attrs, defStyle);
26:     }
27:
28:     @Override
29:     protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
30:         super.onMeasure(widthMeasureSpec, heightMeasureSpec);
31:         setMeasuredDimension(getMeasuredWidth(), getMeasuredWidth()); // width
32:     }
33:
34: }
```

**./drawable/gradient.xml**

**Sat Nov 22 10:17:02 2014**

**1**

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <shape xmlns:android="http://schemas.android.com/apk/res/android" >
3:     <gradient android:angle="270" android:endColor="#484848" android:startColor="#00000000" />
4: </shape>
```



```
1: <?xml version="1.0" encoding="utf-8"?>
2: <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent">
5:
6:     <GridView
7:         android:id="@+id/gvMain"
8:         android:layout_width="match_parent"
9:         android:layout_height="match_parent"
10:        android:horizontalSpacing="1dp"
11:        android:numColumns="2"
12:        android:padding="0dp"
13:        android:stretchMode="columnWidth"
14:        android:verticalSpacing="1dp" >
15:
16:     </GridView>
17:
18: </FrameLayout>
```



./layout/activity\_choose\_route.xml Sat Jan 17 16:26:20 2015 1

```
1: <GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
2:     xmlns:tools="http://schemas.android.com/tools"
3:     android:id="@+id/GridLayout1"
4:     android:layout_width="match_parent"
5:     android:layout_height="wrap_content"
6:     android:columnCount="2"
7:     android:orientation="horizontal"
8:     android:paddingLeft="5sp"
9:     android:paddingTop="10sp"
10:    android:rowCount="5" >
11:
12:    <ImageView
13:        android:id="@+id/lm1"
14:        android:src="@drawable/location_marker" />
15:
16:    <TextView
17:        android:id="@+id/current_location"
18:        android:text="@string/aktueller_standort"
19:        android:textSize="25sp"
20:        android:layout_marginTop="5dp" />
21:
22:    <ImageView
23:        android:id="@+id/lm2"
24:        android:src="@drawable/location_marker" />
25:
26:    <Spinner
27:        android:id="@+id/locations"
28:        android:layout_width="248dp"
29:        android:entries="@array/selectable_locations"
30:        android:spinnerMode="dropdown" />
31:
32:    <Space
33:        android:layout_column="0"
34:        android:layout_width="50dp"
35:        android:layout_height="10dp" />
36:
37:    <Button
38:        android:id="@+id/start_route"
39:        android:layout_gravity="left|top"
40:        android:text="@string/start_route" />
41:
42:    <Space
43:        android:layout_column="0"
44:        android:layout_width="50dp"
45:        android:layout_height="10dp" />
46:
47:    <TextView
48:        android:id="@+id/recent_routes"
49:        android:text="@string/recent_routes"
```

./layout/activity\_choose\_route.xml Sat Jan 17 16:26:20 2015 2

```
50:        android:textSize="15sp"
51:        android:layout_marginTop="25dp" />
52:
53:    <Space
54:        android:layout_column="0"
55:        android:layout_width="50dp"
56:        android:layout_height="10dp" />
57:
58:    <ListView
59:        android:id="@+id/list"
60:        android:layout_width="248dp"
61:        android:layout_height="230dp" />
62:
63: </GridLayout>
```

**./layout/activity\_help.xml**

**Tue Jan 20 09:21:05 2015**

**1**

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent"
5:     android:orientation="vertical" >
6:
7:     <WebView
8:         android:id="@+id/help_content"
9:         android:layout_width="fill_parent"
10:        android:layout_height="fill_parent" />
11:
12: </LinearLayout>
```



```
1: <?xml version="1.0" encoding="utf-8"?>
2: <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent"
5:     android:orientation="vertical" >
6:
7:     <WebView
8:         android:id="@+id/imprint_content"
9:         android:layout_width="fill_parent"
10:        android:layout_height="fill_parent" />
11:
12:
13: </LinearLayout>
```



```
1: <?xml version="1.0" encoding="utf-8"?>
2: <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="fill_parent"
4:     android:layout_height="fill_parent" >
5:
6:     <LinearLayout
7:         android:layout_width="match_parent"
8:         android:layout_height="wrap_content"
9:         android:orientation="vertical"
10:        android:weightSum="3" >
11:
12:         <RelativeLayout
13:             android:layout_width="match_parent"
14:             android:layout_height="0px"
15:             android:layout_weight="1"
16:             android:focusableInTouchMode="true"
17:             android:orientation="vertical" >
18:
19:             <ImageView
20:                 android:id="@+id/img_background1"
21:                 android:layout_width="match_parent"
22:                 android:layout_height="match_parent"
23:                 android:layout_alignParentLeft="true"
24:                 android:layout_alignParentTop="true"
25:                 android:contentDescription="@string/background1"
26:                 android:gravity="center_horizontal|center_vertical"
27:                 android:scaleType="centerCrop"
28:                 android:src="@drawable/welt" />
29:
30:             <Button
31:                 android:id="@+id/btn_no_hfu_student"
32:                 style="?android:attr/buttonStyle"
33:                 android:layout_width="wrap_content"
34:                 android:layout_height="wrap_content"
35:                 android:layout_alignParentBottom="true"
36:                 android:layout_centerHorizontal="true"
37:                 android:layout_marginBottom="14dp"
38:                 android:text="@string/no_hfu_student"
39:                 android:textColor="#FFFFFF"
40:                 android:textSize="25sp" />
41:
42:             <Button
43:                 android:id="@+id/btn_hfu_student"
44:                 style="?android:attr/buttonStyle"
45:                 android:layout_width="wrap_content"
46:                 android:layout_height="wrap_content"
47:                 android:layout_above="@+id/btn_no_hfu_student"
48:                 android:layout_centerHorizontal="true"
49:                 android:paddingBottom="16dp"
```

```
50:         android:text="@string/hfu_student"
51:         android:textColor="#FFFFFF"
52:         android:textSize="25sp" />
53:
54:     <TextView
55:         android:id="@+id/text_welcome_back"
56:         android:layout_width="wrap_content"
57:         android:layout_height="wrap_content"
58:         android:layout_centerHorizontal="true"
59:         android:layout_centerVertical="true"
60:         android:text="@string/welcome_back"
61:         android:textAppearance="?android:attr/textAppearanceMedium"
62:         android:textColor="#FFFFFF"
63:         android:visibility="gone" />
64:
65:     <RelativeLayout
66:         android:id="@+id/relativeLayout1"
67:         android:layout_width="wrap_content"
68:         android:layout_height="83dp"
69:         android:layout_alignParentLeft="true"
70:         android:layout_alignParentTop="true"
71:         android:focusableInTouchMode="true"
72:         android:orientation="vertical" >
73:
74:         <TextView
75:             android:id="@+id/editText2"
76:             android:layout_width="wrap_content"
77:             android:layout_height="wrap_content"
78:             android:layout_alignParentLeft="true"
79:             android:layout_centerVertical="true"
80:             android:paddingLeft="20dp"
81:             android:text="@string/app_name1"
82:             android:textColor="@color/primary"
83:             android:textSize="20sp" />
84:
85:         <TextView
86:             android:id="@+id/editText1"
87:             android:layout_width="wrap_content"
88:             android:layout_height="wrap_content"
89:             android:layout_alignParentRight="true"
90:             android:layout_centerVertical="true"
91:             android:paddingRight="10dp"
92:             android:text="@string/app_name2"
93:             android:textColor="@color/primary"
94:             android:textSize="20sp" />
95:
96:         <ImageView
97:             android:id="@+id/img_werdmobil"
98:             android:layout_width="match_parent"
```

```
99:             android:layout_height="wrap_content"
100:            android:layout_alignParentLeft="true"
101:            android:layout_alignParentTop="true"
102:            android:contentDescription="@string/logo"
103:            android:paddingBottom="16dp"
104:            android:paddingLeft="36dp"
105:            android:paddingRight="40dp"
106:            android:paddingTop="20dp"
107:            android:src="@drawable/ic_launcher"
108:            android:visibility="visible" />
109:        </RelativeLayout>
110:
111:        <TextView
112:            android:id="@+id/text_email"
113:            android:layout_width="wrap_content"
114:            android:layout_height="wrap_content"
115:            android:layout_below="@+id/relativeLayout1"
116:            android:layout_centerHorizontal="true"
117:            android:gravity="center"
118:            android:text="@string/your_email_address"
119:            android:textAppearance="?android:attr/textAppearanceMedium"
120:            android:textColor="#FFFFFF"
121:            android:textSize="25sp"
122:            android:visibility="gone" />
123:
124:        <EditText
125:            android:id="@+id/input_email"
126:            android:layout_width="wrap_content"
127:            android:layout_height="wrap_content"
128:            android:layout_above="@+id/btn_parse_mail"
129:            android:layout_alignParentLeft="true"
130:            android:layout_marginBottom="30dp"
131:            android:background="#88676767"
132:            android:ems="10"
133:            android:hint="@string/maxmustermann"
134:            android:inputType="textEmailAddress"
135:            android:textColor="#FFFFFF"
136:            android:visibility="gone" />
137:
138:        <TextView
139:            android:id="@+id/text_email_hfu_string"
140:            android:layout_width="wrap_content"
141:            android:layout_height="wrap_content"
142:            android:layout_alignBottom="@+id/input_email"
143:            android:layout_alignParentRight="true"
144:            android:layout_alignTop="@+id/input_email"
145:            android:background="#88676767"
146:            android:text="@string/hsfurtwangende"
147:            android:textAppearance="?android:attr/textAppearanceMedium"
```

```
148:             android:textColor="#FFFFFF"
149:             android:visibility="gone" />
150:
151:             <Button
152:                 android:id="@+id/btn_parse_mail"
153:                 android:layout_width="wrap_content"
154:                 android:layout_height="wrap_content"
155:                 android:layout_alignParentBottom="true"
156:                 android:layout_centerHorizontal="true"
157:                 android:layout_marginBottom="141dp"
158:                 android:text="@string/send"
159:                 android:textColor="#FFFFFF"
160:                 android:visibility="gone" />
161:
162:             </RelativeLayout>
163:
164:         </LinearLayout>
165:
166:     </RelativeLayout>
```

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent"
5:     android:orientation="vertical" >
6:
7:     <TextView
8:         android:id="@+id/textView1"
9:         android:layout_width="match_parent"
10:        android:layout_height="wrap_content"
11:        android:layout_weight="0.18"
12:        android:gravity="center"
13:        android:text="@string/record_route_info"
14:        android:textAppearance="?android:attr/textAppearanceMedium"
15:        android:layout_marginLeft="10dp"
16:        android:layout_marginRight="10dp" />
17:
18:     <ProgressBar
19:         android:id="@+id/progressBar1"
20:         style="?android:attr/progressBarStyleLarge"
21:         android:layout_width="match_parent"
22:         android:layout_height="wrap_content"
23:         android:layout_weight="0.68" />
24:
25:     <Button
26:         android:id="@+id/endRoute"
27:         android:layout_width="wrap_content"
28:         android:layout_height="wrap_content"
29:         android:layout_gravity="center_vertical|center_horizontal"
30:         android:text="@string/endRouteDestination" />
31:
32: </LinearLayout>
```



```
1: <?xml version="1.0" encoding="utf-8"?>
2: <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="fill_parent" android:layout_height="fill_parent"
4:     android:fillViewport="true">
5:     <LinearLayout
6:         android:layout_width="match_parent"
7:         android:layout_height="wrap_content"
8:         android:orientation="vertical"
9:         android:weightSum="3">
10:
11:         <LinearLayout
12:             android:layout_width="match_parent"
13:             android:layout_height="0dp"
14:             android:layout_weight="1">
15:
16:             <fragment
17:                 android:id="@+id/map"
18:                 android:name="com.google.android.gms.maps.MapFragment"
19:                 android:layout_width="match_parent"
20:                 android:layout_height="352dp" />
21:
22:         </LinearLayout>
23:
24:         <RelativeLayout
25:             android:layout_width="match_parent"
26:             android:layout_height="0dp"
27:             android:orientation="vertical"
28:             android:layout_marginTop="10dp"
29:             android:layout_marginLeft="10dp"
30:             android:layout_weight="2" >
31:
32:             <!-- OBD-INFOS -->
33:             <TextView
34:                 android:id="@+id/route_details_obd"
35:                 android:layout_width="wrap_content"
36:                 android:layout_height="wrap_content"
37:                 android:text="@string/routen_infos"
38:                 android:textSize="27sp"
39:                 android:layout_centerHorizontal="true" />
40:
41:             <!-- START+DESTINATION -->
42:             <TextView
43:                 android:id="@+id/start_destination"
44:                 android:layout_width="wrap_content"
45:                 android:layout_height="wrap_content"
46:                 android:text="@string/start_and_destination"
47:                 android:textSize="20sp"
48:                 android:layout_below="@+id/route_details_obd"
49:                 android:layout_centerHorizontal="true" />
```

```
50:  
51:        <!-- Date -->  
52:        <TextView  
53:            android:id="@+id/route_details_date"  
54:            android:layout_width="wrap_content"  
55:            android:layout_height="wrap_content"  
56:            android:text="@string/no_date"  
57:            android:textSize="20sp"  
58:            android:layout_below="@+id/start_destination"  
59:            android:layout_centerHorizontal="true"  
60:            android:layout_marginBottom="10dp" />  
61:  
62:        <!-- Desc: Fahrtzeit in Min -->  
63:        <TextView  
64:            android:id="@+id/route_details_min"  
65:            android:layout_width="wrap_content"  
66:            android:layout_height="wrap_content"  
67:            android:text="@string/travel_time"  
68:            android:textSize="20sp"  
69:            android:layout_below="@+id/route_details_date"  
70:            android:visibility="gone" /><!-- delete visibility, after the function exist -->  
71:  
72:        <!-- Fahrtzeit in Min -->  
73:        <TextView  
74:            android:id="@+id/rd_min"  
75:            android:layout_width="wrap_content"  
76:            android:layout_height="wrap_content"  
77:            android:text="@string/no_detail"  
78:            android:textSize="20sp"  
79:            android:layout_below="@+id/route_details_date"  
80:            android:layout_toRightOf="@+id/route_details_min"  
81:            android:visibility="gone" /><!-- delete visibility, after the function exist -->  
82:  
83:        <!-- Desc: Fahrtstrecke in KM -->  
84:        <TextView  
85:            android:id="@+id/route_details_km"  
86:            android:layout_width="wrap_content"  
87:            android:layout_height="wrap_content"  
88:            android:text="@string/route_length"  
89:            android:textSize="20sp"  
90:            android:layout_below="@+id/route_details_min"  
91:            android:visibility="gone" /><!-- delete visibility, after the function exist -->  
92:  
93:        <!-- Fahrtstrecke in KM -->  
94:        <TextView  
95:            android:id="@+id/rd_km"  
96:            android:layout_width="wrap_content"  
97:            android:layout_height="wrap_content"  
98:            android:text="@string/no_detail"
```

```
./layout/activity_route_details.xml      Mon Jan 19 10:31:40 2015      3

99:                               android:textSize="20sp"
100:                              android:layout_below="@+id/route_details_min"
101:                              android:layout_toRightOf="@+id/route_details_kmh"
102:                              android:visibility="gone" /> <!-- delete visibility, after the function exist -->
103:
104:                              <!-- Desc: Durchschnittsgeschw. -->
105:                              <TextView
106:                                  android:id="@+id/route_details_kmh"
107:                                  android:layout_width="wrap_content"
108:                                  android:layout_height="wrap_content"
109:                                  android:text="@string/average_speed"
110:                                  android:textSize="20sp"
111:                                  android:layout_below="@+id/route_details_km" />
112:
113:                              <!-- Durchschnittsgeschw. -->
114:                              <TextView
115:                                  android:id="@+id/average_speed"
116:                                  android:layout_width="wrap_content"
117:                                  android:layout_height="wrap_content"
118:                                  android:text="@string/no_detail"
119:                                  android:textSize="20sp"
120:                                  android:layout_below="@+id/route_details_km"
121:                                  android:layout_toRightOf="@+id/route_details_kmh" />
122:
123:                              <!-- Desc: Verbrauch -->
124:                              <TextView
125:                                  android:id="@+id/route_details_consumption"
126:                                  android:layout_width="wrap_content"
127:                                  android:layout_height="wrap_content"
128:                                  android:text="@string/wastage"
129:                                  android:textSize="20sp"
130:                                  android:layout_below="@+id/route_details_kmh" />
131:
132:                              <!-- Verbrauch -->
133:                              <TextView
134:                                  android:id="@+id/average_consumption"
135:                                  android:layout_width="wrap_content"
136:                                  android:layout_height="wrap_content"
137:                                  android:text="@string/no_detail"
138:                                  android:textSize="20sp"
139:                                  android:layout_below="@+id/route_details_kmh"
140:                                  android:layout_toRightOf="@+id/route_details_consumption" />
141:                              </RelativeLayout>
142:                          </LinearLayout>
143: </ScrollView>
```



```
1: <?xml version="1.0" encoding="utf-8"?>
2: <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent"
5:     android:orientation="vertical"
6:     android:background="#FF000000" >
7:
8:     <Button
9:         android:id="@+id/b_refresh_btn"
10:        android:layout_width="match_parent"
11:        android:layout_height="wrap_content"
12:        android:text="@string/refresh"
13:        android:background="#dddddd"
14:        android:layout_margin="20dp"
15:        android:layout_marginBottom="5dp"
16:        android:onClick="refresh" />
17:
18:     <ListView
19:         android:id="@+id/b_listView"
20:         android:layout_width="match_parent"
21:         android:layout_height="wrap_content"
22:         android:layout_margin="20dp"
23:         android:background="#ffffff" >
24:     </ListView>
25:
26: </LinearLayout>
```



```
1: <?xml version="1.0" encoding="utf-8"?>
2: <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="fill_parent" android:layout_height="fill_parent">
4:
5:     <LinearLayout
6:         android:layout_width="match_parent"
7:         android:layout_height="wrap_content"
8:         android:orientation="vertical">
9:
10:        <LinearLayout
11:            android:layout_width="match_parent"
12:            android:layout_height="fill_parent">
13:
14:            <!-- User_Picture -->
15:            <ImageView
16:                android:id="@+id/img_user_picture"
17:                android:layout_width="wrap_content"
18:                android:layout_height="fill_parent"
19:                android:contentDescription="@string/contentDesc_user_picture"
20:                android:src="@drawable/user_picture"
21:                android:scaleType="centerCrop" />
22:        </LinearLayout>
23:
24:        <RelativeLayout
25:            android:layout_width="match_parent"
26:            android:layout_height="fill_parent"
27:            android:paddingLeft="16dp"
28:            android:paddingRight="16dp"
29:            android:paddingBottom="16dp"
30:            android:paddingTop="20dp"
31:            android:orientation="vertical">
32:
33:            <!-- Firstname -->
34:            <ImageView
35:                android:id="@+id/img_firstname"
36:                android:layout_width="wrap_content"
37:                android:layout_height="wrap_content"
38:                android:contentDescription="@string/firstname"
39:                android:src="@drawable/ic_firstname"
40:                android:layout_alignBottom="@+id/firstname" />
41:
42:            <EditText
43:                android:id="@+id/firstname"
44:                android:layout_width="match_parent"
45:                android:layout_height="wrap_content"
46:                android:hint="@string/firstname"
47:                android:inputType="text"
48:                android:layout_toRightOf="@+id/img_firstname"
49:                android:layout_marginLeft="10dp"
```

```
50:                     android:paddingRight="16dp" />
51:                     <!-- Lastname -->
52:             <ImageView
53:                 android:id="@+id/img_lastname"
54:                 android:layout_width="wrap_content"
55:                 android:layout_height="wrap_content"
56:                 android:contentDescription="@string/lastname"
57:                 android:src="@drawable/ic_lastname"
58:                 android:layout_alignBottom="@+id/lastname"
59:                 android:layout_marginTop="7dp" />
60:
61:             <EditText
62:                 android:id="@+id/lastname"
63:                 android:layout_width="fill_parent"
64:                     android:layout_height="wrap_content"
65:                 android:hint="@string/lastname"
66:                 android:inputType="text"
67:                 android:layout_below="@+id/firstname"
68:                 android:layout_toRightOf="@+id/img_lastname"
69:                 android:layout_marginLeft="10dp"
70:                 android:layout_marginTop="7dp" />
71:
72:             <!-- Hometown -->
73:             <ImageView
74:                 android:id="@+id/img_hometown"
75:                 android:layout_width="wrap_content"
76:                 android:layout_height="wrap_content"
77:                 android:contentDescription="@string/hometown"
78:                 android:src="@drawable/ic_hometown"
79:                 android:layout_alignBottom="@+id/hometown"
80:                 android:layout_marginTop="7dp"
81:                 android:inputType="numberSigned" />
82:
83:             <EditText
84:                 android:id="@+id/hometown"
85:                 android:layout_width="fill_parent"
86:                     android:layout_height="wrap_content"
87:                 android:hint="@string/hometown"
88:                 android:inputType="text"
89:                 android:layout_below="@+id.lastname"
90:                 android:layout_toRightOf="@+id/img_hometown"
91:                 android:layout_marginLeft="10dp"
92:                 android:layout_marginTop="7dp" />
93:
94:             <!-- Phone -->
95:             <ImageView
96:                 android:id="@+id/img_phone"
97:                 android:layout_width="wrap_content"
98:                 android:layout_height="wrap_content"
```

```
99:                     android:contentDescription="@string/phone"
100:                    android:src="@drawable/ic_phone"
101:                    android:layout_alignBottom="@+id/phone"
102:                    android:layout_marginTop="7dp" />
103:
104:            <EditText
105:                android:id="@+id/phone"
106:                android:layout_width="fill_parent"
107:                    android:layout_height="wrap_content"
108:                android:hint="@string/phone"
109:                android:inputType="phone"
110:                android:layout_below="@+id/hometown"
111:                android:layout_toRightOf="@+id/img_phone"
112:                android:layout_marginLeft="10dp"
113:                android:layout_marginTop="7dp" />
114:
115:            <!-- Email -->
116:            <ImageView
117:                android:id="@+id/img_email"
118:                android:layout_width="wrap_content"
119:                android:layout_height="wrap_content"
120:                android:contentDescription="@string/email"
121:                android:src="@drawable/ic_email"
122:                android:layout_alignBottom="@+id/email"
123:                android:layout_marginTop="7dp" />
124:
125:            <EditText
126:                android:id="@+id/email"
127:                android:layout_width="fill_parent"
128:                    android:layout_height="wrap_content"
129:                android:hint="@string/email"
130:                android:inputType="textWebEmailAddress"
131:                android:layout_below="@+id/phone"
132:                android:layout_toRightOf="@+id/img_email"
133:                android:layout_marginLeft="10dp"
134:                android:layout_marginTop="7dp" />
135:
136:            <!-- OBD -->
137:            <TextView
138:                android:id="@+id/tv_oobd"
139:                android:layout_width="wrap_content"
140:                android:layout_height="wrap_content"
141:                android:text="@string/oobd"
142:                android:textSize="20sp"
143:                android:layout_below="@+id/img_email"
144:                android:layout_marginTop="25dp" />
145:
146:            <Switch
147:                android:id="@+id/tBtn_oobd"
```

```
148:             android:layout_width="wrap_content"
149:             android:switchMinWidth="150dp"
150:             android:layout_height="wrap_content"
151:             android:layout_alignBottom="@+id/tv_oob"
152:             android:textOn="An"
153:                 android:textOff="Aus"
154:             android:layout_centerHorizontal="true" />
155:
156:             <Button
157:                 android:id="@+id/btn_save"
158:                 android:layout_width="wrap_content"
159:                 android:layout_height="wrap_content"
160:                 android:text="@string/save"
161:                 android:layout_below="@+id/tBtn_oob"
162:                 android:layout_centerHorizontal="true"
163:                 android:layout_marginTop="20dp"
164:                 android:onClick="saveSettings" />
165:             </RelativeLayout>
166:         </LinearLayout>
167:     </ScrollView>
```

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent"
5:     android:orientation="vertical" >
6:
7:     <TextView
8:         android:id="@+id/textView_verifyUser"
9:         android:layout_width="wrap_content"
10:        android:layout_height="wrap_content"
11:        android:text="@string/note_verify_code"
12:        android:textAppearance="?android:attr/textAppearanceMedium"
13:        android:layout_marginLeft="10dp"
14:        android:layout_marginRight="10dp"
15:        android:gravity="center" />
16:
17:     <EditText
18:         android:id="@+id/input_verify_code"
19:         android:layout_width="match_parent"
20:         android:layout_height="wrap_content"
21:         android:ems="10"
22:         android:inputType="numberSigned" />
23:
24:     <Button
25:         android:id="@+id/btn_send_verify_code"
26:         android:layout_width="wrap_content"
27:         android:layout_height="wrap_content"
28:         android:text="@string/check_code" />
29:
30:     <TextView
31:         android:id="@+id/textView_info_number"
32:         android:layout_width="wrap_content"
33:         android:layout_height="wrap_content"
34:         android:text="@string/input_your_device_number"
35:         android:layout_marginTop="18dp"
36:         android:layout_marginLeft="5dp"
37:         android:textSize="20sp"
38:         android:textStyle="bold"
39:         android:visibility="gone"
40:         android:textAppearance="?android:attr/textAppearanceMedium" />
41:
42:     <LinearLayout
43:         android:layout_width="fill_parent"
44:         android:layout_height="wrap_content"
45:         android:orientation="horizontal">
46:
47:         <TextView
48:             android:id="@+id/textView_desc_number"
49:             android:layout_width="wrap_content"
```

./layout/activity\_user\_verify.xml Sun Jan 18 09:37:11 2015 2

```
50:             android:layout_height="wrap_content"
51:             android:text="@string/your_device_number"
52:             android:layout_marginTop="18dp"
53:             android:layout_marginLeft="5dp"
54:             android:visibility="gone"
55:             android:textAppearance="?android:attr/textAppearanceMedium" />
56:
57:             <EditText
58:                 android:id="@+id/input_number"
59:                 android:layout_width="0dp"
60:                 android:layout_height="wrap_content"
61:                 android:layout_weight="1"
62:                 android:layout_marginTop="15dp"
63:                 android:layout_marginLeft="5dp"
64:                 android:hint="@string/example_number"
65:                 android:visibility="gone"
66:                 android:inputType="number" />
67:         </LinearLayout>
68:
69:         <Button
70:             android:id="@+id/btn_send_number"
71:             android:layout_width="wrap_content"
72:             android:layout_height="wrap_content"
73:             android:layout_marginTop="10dp"
74:             android:layout_gravity="center_horizontal"
75:             android:visibility="gone"
76:             android:text="@string/send" />
77:
78:     </LinearLayout>
```

./layout/item.xml Sun Nov 23 12:17:28 2014 1

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent"
5:     android:layout_margin="1dp" >
6:
7:     <com.HFUwerdMobil.util.SquareImageView
8:         android:id="@+id/picture"
9:         android:layout_width="match_parent"
10:        android:layout_height="match_parent"
11:        android:layout_margin="0dp"
12:        android:padding="0dp"
13:        android:scaleType="centerCrop" />
14:
15:     <TextView
16:         android:id="@+id/text"
17:         android:layout_width="match_parent"
18:         android:layout_height="wrap_content"
19:         android:layout_gravity="bottom"
20:         android:background="@drawable/gradient"
21:         android:paddingBottom="15dp"
22:         android:paddingLeft="10dp"
23:         android:paddingRight="10dp"
24:         android:paddingTop="15dp"
25:         android:textColor="@android:color/white" />
26:
27: </FrameLayout>
```



**./layout/recent\_routes.xml**

**Sat Jan 17 13:46:40 2015**

**1**

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <TextView xmlns:android="http://schemas.android.com/apk/res/android"
3:     android:id="@+id/label"
4:     android:layout_width="fill_parent"
5:     android:layout_height="fill_parent"
6:     android:padding="10dip"
7:     android:textSize="25sp" />
```



```
1: <?xml version="1.0" encoding="utf-8"?>
2: <menu xmlns:android="http://schemas.android.com/apk/res/android" >
3:     <item
4:         android:id="@+id/menu_home"
5:         android:orderInCategory="100"
6:         android:showAsAction="never"
7:         android:title="@string/menu_home"/>
8:     <item
9:         android:id="@+id/menu_settings"
10:        android:orderInCategory="100"
11:        android:showAsAction="never"
12:        android:title="@string/menu_settings"/>
13:     <item
14:         android:id="@+id/menu_about"
15:         android:orderInCategory="100"
16:         android:showAsAction="never"
17:         android:title="@string/menu_about"/>
18:     <item
19:         android:id="@+id/menu_imprint"
20:         android:orderInCategory="100"
21:         android:showAsAction="never"
22:         android:title="@string/menu_imprint"/>
23:     <item
24:         android:id="@+id/menu_help"
25:         android:orderInCategory="100"
26:         android:showAsAction="never"
27:         android:title="@string/menu_help"/>
28: </menu>
```



./values/colors.xml

Sat Jan 17 11:33:24 2015

1

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <resources>
3:   <color name="primary">#4CAF50</color>
4:   <color name="primary_dark">#1B5E20</color>
5:   <color name="accent">#00E676</color>
6:   <color name="text">#01579B</color>
7:   <color name="windowBackground">#E8F5E9</color>
8: </resources>
```



```
1: <?xml version="1.0" encoding="utf-8"?>
2: <resources>
3:   <string name="app_name">HFUwerdMobil!</string>
4:   <string name="authentification">HFUwerdMobil!</string>
5:   <string name="choose_route">Route wÄhlen</string>
6:   <string name="record_route">Route wird aufgezeichnet..</string>
7:   <string name="route_details">Routendetails</string>
8:   <string name="about">Ã\234ber</string>
9:   <string name="driven_routes">gefahrenen Routen</string>
10:  <string name="help">Hilfe</string>
11:  <string name="imprint">Impressum</string>
12:  <string name="settings">Einstellungen</string>
13:  <string name="user_verify">Verifizierung</string>
14:
15:  <string name="menu_home">Route wÄhlen</string>
16:  <string name="menu_driven_routes">Routendetails</string>
17:  <string name="menu_settings">Einstellungen</string>
18:  <string name="menu_about">Ã\234ber</string>
19:  <string name="menu_imprint">Impressum</string>
20:  <string name="menu_help">Hilfe</string>
21:
22:  <string name="firstname">Vorname</string>
23:  <string name="lastname">Nachname</string>
24:  <string name="hometown">PLZ</string>
25:  <string name="phone">Telefon</string>
26:  <string name="obd">OBD</string>
27:  <string name="save">Speichern</string>
28:  <string name="savedSettings">Einstellungen gespeichert</string>
29:  <string name="contentDesc_user_picture">Benutzerbild</string>
30:
31:  <string name="route_start">Startort</string>
32:  <string name="route_seperator">---</string>
33:  <string name="route_end">Zielort String</string>
34:  <string name="route_date">Datum</string>
35:  <string name="routen_infos"><b>Routen-Infos</b></string>
36:  <string name="start_and_destination">Startort - Zielort</string>
37:  <string name="no_date">Kein Datum vorhanden</string>
38:  <string name="no_detail"> k.A.</string>
39:  <string name="travel_time">Fahrzeit:</string>
40:  <string name="route_length">Fahrstrecke:</string>
41:  <string name="average_speed">Durchschnittsgeschwindigkeit:</string>
42:  <string name="wastage">Verbrauch:</string>
43:
44:  <string name="dialog_fatalError_title">Kritischer Fehler</string>
45:  <string name="dialog_fatalError_message">Es ist ein schwerwiegender Fehler aufgetreten. Die App muss beendet werden
.</string>
46:  <string name="dialog_fatalError_terminate">Okay.</string>
47:  <string name="dialog_primaryMailChanged_title">E-Mail Ändern</string>
48:  <string name="dialog_primaryMailChanged_message">Die E-Mail Adresse Ihres Android GerÃtes hat sich verÃndert. So
```

```
11 die neue E-Mail Adresse auch als neue App-E-Mail-Adresse verwendet werden?</string>
49:    <string name="dialog_primaryMailChanged_yes">Ja</string>
50:    <string name="dialog_primaryMailChanged_no">Nein danke</string>
51:    <string name="dialog_primaryMailChangedSuccessfully_title">E-Mail geÄndert</string>
52:    <string name="dialog_primaryMailChangedSuccessfully_message">Die App-E-Mail Adresse wurde durch die neue E-Mail Adr
esse Ihres Android GerÃ¤tes ersetzt.</string>
53:    <string name="dialog_primaryMailChangedSuccessfully_okay">Okay</string>
54:    <string name="dialog_locationProviderNotAvailable_title">Standorterfassung fehlgeschlagen</string>
55:    <string name="dialog_locationProviderNotAvailable_message">Wir kÃ¶nnen deinen aktuellen Standort weiterhin nicht me
hr erfassen. Bitte sorge dafÃ¼r, dass Du GPS wÃ¤hrend der Routen Aufzeichnung nicht deaktivierst. Wir beenden nun die aktuelle
Routen Aufzeichnung.</string>
56:    <string name="dialog_locationProviderNotAvailable_okay">Okay</string>
57:    <string name="dialog_inputManuallyMail_title">Deine E-Mail Adresse</string>
58:    <string name="dialog_inputManuallyMail_message">Wir konnten die E-Mail Adresse deines Google Accounts auf dem Andro
id GerÃ¤t nicht auslesen. Bitte gib deine E-Mail Adresse ein:</string>
59:    <string name="dialog_inputManuallyMail_okay">Okay</string>
60:    <string name="dialog_verifycodeInputFailed_title">Verifizierung fehlerhaft</string>
61:    <string name="dialog_verifycodeInputFailed_message">Der eingegebene Verifizierungscode stimmt leider nicht mit dem
Code Ã¼berein, den wir dir per E-Mail geschickt haben.</string>
62:    <string name="dialog_verifycodeInputFailed_okay">Erneut versuchen</string>
63:    <string name="dialog_networkFailed_title">Fehlerhafter Vorgang</string>
64:    <string name="dialog_networkFailed_message">Keine Verbindung mit dem Netzwerk mÃ¶glich. Bitte prÃ¼fe deine Internet
verbindung. (developer note: no internet connectior or no connection to server available)</string>
65:    <string name="dialog_networkFailed_okay">Okay</string>
66:    <string name="dialog_emailNotValid_title">E-Mail nicht korrekt</string>
67:    <string name="dialog_emailNotValid_message">Die eingegebene E-Mail ist nicht korrekt. Bitte Ã¼berprÃ¼fe deine Einga
be.</string>
68:    <string name="dialog_emailNotValid_okay">Weiter</string>
69:    <string name="dialog_deviceNumberNotValid_title">Handynummer nicht korrekt</string>
70:    <string name="dialog_deviceNumberNotValid_message">Bitte eine korrekte Handynummer (mit mindestens 6 Zahlen) eingeb
en.</string>
71:    <string name="dialog_deviceNumberNotValid_okay">Are you serious?</string>
72:
73:
74:    <string name="activate_gps">Bitte GPS aktivieren</string>
75:
76:    <string name="aktueller_standort">aktueller Standort</string>
77:    <string name="start_route">Route starten</string>
78:    <string name="destination_chosen">Startpunkt ausgewÃ¤hlt</string>
79:
80:    <string name="background1">Hintergrund1</string>
81:        <string name="hfu_student">Mit HFU Mailkonto anmelden</string>
82:        <string name="no_hfu_student">Mit externem Mailkonto anmelden</string>
83:        <string name="logo">HFUWerdmobil!</string>
84:        <string name="app_name1">HFU</string>
85:        <string name="app_name2">werdMobil!</string>
86:
87:        <string name="furtwangen">HFU Furtwangen</string>
88:        <string name="schwenningen">HFU Schwenningen</string>
```

./values/strings.xml        Tue Jan 20 22:48:50 2015        3

```
89:      <string name="tuttlingen">HFU Tuttlingen</string>
90:      <string name="home">nach Hause</string>
91:
92:      <string-array name="selectable_locations">
93:          <item>@string/furtwangen</item>
94:          <item>@string/schwenningen</item>
95:          <item>@string/tuttlingen</item>
96:          <item>@string/home</item>
97:      </string-array>
98:
99:      <string name="refresh">Refresh</string>
100:     <string name="bluetooth">Bluetooth</string>
101:     <string name="recent_routes">Zuletzt gefahrene Routen</string>
102:     <string name="record_route_info">Deine Fahrt wird nun aufgezeichnet und in Echtzeit an unsere Server Ã¼bertrage
n. Solange Du die Ã34bertragung nicht beendest, kÃ¶nnen Dich potenzielle Mitfahrer anrufen.</string>
103:    <string name="endRouteDestination">Ich habe mein Ziel erreicht!</string>
104:    <string name="maxmustermann">max.mustermann</string>
105:    <string name="your_email_address">Deine E-Mail Adresse:</string>
106:    <string name="hsfurtwangende">@\hs-furtwangen.de</string>
107:    <string name="welcome_back">Willkommen zurÃ¼ck, </string>
108:    <string name="send">Absenden</string>
109:    <string name="email">E-Mail</string>
110:
111:    <string name="input_your_device_number">Bitte Handynummer angeben</string>
112:        <string name="your_device_number">Deine Handynummer:</string>
113:        <string name="example_number">01673235287</string>
114:
115:        <string name="note_verify_code">Wir haben dir soeben einen Verifizierungscode an deine E-Mail geschickt. Bitte
gib diesen Code hier ein, damit wir Dir deine E-Mail Adresse zuordnen kÃ¶nnen.</string>
116:        <string name="check_code">Verifizierung abschlieÃ37en</string>
117:
118:        <string name="hfu">HFU</string>
119:        <string name="tobiasstraub">Tobias Straub</string>
120:        <string name="susannereroos">Susanne Roos</string>
121:        <string name="matthiasfeichtmeier">Matthias Feichtmeier</string>
122:        <string name="thomaslesinski">Thomas Lesinski</string>
123:        <string name="veronikakinzel">Veronika Kinzel</string>
124:        <string name="hfuerdmobillogo">HFUerdmobil!</string>
125:        <string name="no_routes_to_show">bisher keine Route</string>
126:        <string name="is_destination_yet">Sie sind bereits am Ziel</string>
127:        <string name="is_destination">Sie haben das Ziel erreicht</string>
128:        <string name="no_data_available">Keine Routendaten aufgezeichnet</string>
129:
130:        <string name="help_content">
131:            <![CDATA[<html>
132:                <head>
133:                    <title>FAQ</title>
134:                </head><body>
135:                    <p><b>Frequently Asked Questions</b></p>
```

136: <p><b>Wie funktioniert die Anmeldung?</b></p>  
137: <p>Die App HFUwerdMobil! ist f&uuml;r Studierende der Hochschule Furtwangen University konzipiert.  
138: Bei der ersten Anmeldung fragt die App ob ein HFU Account vorhanden ist oder nicht. Falls das  
139: der Fall ist wird der Nutzer aufgefordert seine HFU E-Mailadresse einzugeben.  
140: Falls kein HFU Account vorhanden ist pr&uuml;ft die App ob ein Google Account vorhanden ist und  
141: f&uuml;gt diesen dann als User-E-Mailadresse hinzu. Ist kein Google Account vorhanden wird der  
142: Nutzer zur Eingabe einer validen E-Mailadresse aufgefordert.  
143: Derzeit ist es unbedingt notwendig, dass du deinen HFU webmail Account verwendest, da der volle  
144: Funktionsumfang der Anwendung nur f&uuml;r HFU Studierende zur Verf&uuml;gung steht.</p>  
145:  
146: <p><b>Wo gebe ich meine pers&ouml;nlichen Daten an?</b></p>  
147: <p>Unter der Option "Einstellungen" kannst du deinen Vor- und Nachnamen, sowie deine Postleitzahl  
148: und Mobilfunknummer hinterlegen (f&uuml;r Community-Funktionen ben&ouml;tigt).  
149: Hier kannst du auch die Bluetooth Verbindung zu OBD ein/auszuschalten.</p>  
150:  
151: <p><b>Wo trifft sich die HFUwerdmobil! Community?</b></p>  
152: <p>Unter www.hfuwerdmobil.de kannst du dich registrieren und Teil der Community werden!</p>  
153:  
154: <p><b>Wie w&auml;hle ich eine Route?</b></p>  
155: <p>Nach erfolgreicher Anmeldung auf der Startseite gelangt der Nutzer auf "Route w&auml;hlen".  
156: Das erste Feld in der Anzeige ist der aktuelle mittels GPS ermittelte Standort des Nutzers.  
157: Er dient als Ausgangslage f&uuml;r seine Fahrt. Im zweiten Feld kann nun eine Auswahl zum Ziel getroffen werden.  
158: Dies geschieht mittels einer Dropdownliste, in welcher zwischen folgenden Optionen gew&auml;hlt werden kann:  
159: HFU Furtwangen, HFU Villingen-Schwenningen, HFU Tuttlingen und Heimatort.  
160: Das Feld Heimatort bezieht sich auf den in den Benutzereinstellungen hinterlegten Wohnort.  
161: Alternativ kann in der Anzeige der zuletzt gefahrenen Routen eine Auswahl getroffen werden.</p>  
162:  
163: <p><b>Wo sehe ich meine bereits gefahrenen Routen?</b></p>  
164: <p>Die &Uuml;bersicht der zuletzt gefahrenen Routen befindet sich unter "Route w&auml;hlen". Dort werden diese dem Nutzer  
165: in Textform aufgelistet. Hier ist neben Start- und Endpunkt, Datum und Uhrzeit aufgef&uuml;hrt.  
166: Durch Klick auf eine dort aufgef&uuml;hrte Route werden die zugeh&ouml;rigen Routendetails aufgerufen.</p>  
167:  
168: <p><b>Wie starte ich die Aufzeichnung?</b></p>  
169: <p>Ebenfalls unter "Route w&auml;hlen" findest du den Button &bdquo;Route starten&ldquo;. Damit beginnt die Aufzeichnung nun  
170: mit der Aufzeichnung der gefahrenen Route. Im Hintergrund werden kontinuierlich  
171: die aktuellen GPS-Koordinaten ermittelt und zur sp&auml;teren Darstellung der Route in der  
172: serverseitigen Datenbank abgespeichert.</p>  
173:  
174: <p><b>Wie wird die Aufzeichnung beendet?</b></p>  
175: <p>Die Aufzeichnung der Fahrtstrecke terminieren: Die Route wird beendet wenn der Nutzer in den n&auml;chsten Umkreis  
176: der entsprechend gew&auml;hlten Hochschule gelangt ist. In diesem Fall erkennt die App automatisch, dass das Ziel  
177: erreicht wurde. Sofern der Heimatort des Fahrers das Ziel ist, muss der Nutzer die App manuell &uuml;ber den  
178: Button &bdquo;Ich habe mein Ziel erreicht!&ldquo; stoppen.</p>  
179:  
180: <p><b>Wo finde ich die Routendetails?</b></p>  
181: <p>Nachdem die Aufzeichnung der Route beendet wurde, zeigt dir die App nun abschlie&szlig;end einen Google Maps  
182: Kartenausschnitt der gefahrenen Route inklusive Markern an Start und Ziel.

183: Außerdem werden dir Durchschnittsgeschwindigkeit und -verbrauch angezeigt, wenn du  
184: während der Aufzeichnung OBD aktiviert hastest.</p>  
185:  
186: <p><b>Was ist eigentlich dieses OBD?</b></p>  
187: <p>On-Board-Diagnose (OBD) ist ein Fahrzeugdiagnosesystem. Die 16polige OBD-2-Diagnosebuchse im Fahrzeug  
188: (seit 2001 für Benziner bzw. 2004 für Diesel Neuzulassungen verpflichtend) kann mittels eines entsprechenden  
189: Adapters ausgelesen werden. Dieser kommuniziert über Bluetooth mit der App und übermittelt Daten zur  
190: Fahrgeschwindigkeit und zum Treibstoffverbrauch.</p>]]></string>  
191: <string name="imprint\_content">  
192: <![CDATA[<html>  
193: <head>  
194: <title>Impressum</title>  
195: </head>  
196: <body>  
197:  
198: <p>Angaben gemäß § 5 TMG:</p>  
199: <p><b>Hochschule Furtwangen University</b><br />  
200: Robert-Gerwig-Platz 1<br />  
201: 78120 Furtwangen<br />  
202: Vertreten durch:</p>  
203:  
204: <p>[Vertreten durch: Name, Anschrift]<br />  
205: Kontakt:</p>  
206:  
207: <p>Telefon: xxx<br />  
208: Telefax: xxx<br />  
209: E-Mail: t.straub@hs-furtwangen.de</p>  
210: <p><b>Quellenangaben für die verwendeten Bilder und Grafiken:</b></p>  
211:  
212: <p><b>Grafik:</b> "The Travel Bug" <br />  
213: Autor: Pascal Mau (Fakultät DM)</b><br />  
214: unter Verwendung der unter Freier Lizenz stehenden Vektorgrafik "World" von <br />  
215: MacDaddy/Vecteezy.com</p>  
216:  
217:  
218: <b>Haftungsausschluss</b> (Disclaimer)</p>  
219: <p>Quelle: <http://www.e-recht24.de></p>  
220: <b>Haftung für Inhalte</b></p>  
221: <p>Als Diensteanbieter sind wir gemäß § 7 Abs.1 TMG für eigene Inhalte auf diesen Seiten nach den allgemeinen Gesetzen verantwortlich. Nach §§ 8 bis 10 TMG sind wir als Diensteanbieter jedoch nicht verpflichtet, übermittelte oder gespeicherte fremde Informationen zu überwachen oder nach Umständen zu forschen, die auf eine rechtswidrige Täuschigkeit hinweisen. Verpflichtungen zur Entfernung oder Sperrung der Nutzung von Informationen nach den allgemeinen Gesetzen bleiben hiervon unberücksichtigt. Eine diesbezügliche Haftung ist jedoch erst ab dem Zeitpunkt der Kenntnis einer konkreten Rechtsverletzung möglich. Bei Bekanntwerden von entsprechenden Rechtsverletzungen werden wir diese Inhalte umgehend entfernen.</p>  
222: <p><b>Haftung für Links</b></p>  
223: <p>Unser Angebot enthält Links zu externen Webseiten Dritter, auf deren Inhalte wir keinen Einfluss haben. Deshalb können wir für diese fremden Inhalte auch keine Gewähr übernehmen. Für die Inhalte der verlinkten Seiten ist stets der jeweilige Anbieter oder Betreiber der Seiten verantwortlich. Die verlinkten Seiten wurden zum Zeitpunkt der Verlinkung inhaltlich korrekt erachtet.</p>

unkt der Verlinkung auf m&ouml;gliche Rechtsverst&ouml;&szlig;e &uuml;berpr&uuml;ft. Rechtswidrige Inhalte waren zum Zeitpunkt der Verlinkung nicht erkennbar. Eine permanente inhaltliche Kontrolle der verlinkten Seiten ist jedoch ohne konkrete Anhaltspunkte einer Rechtsverletzung nicht zumutbar. Bei Bekanntwerden von Rechtsverletzungen werden wir derartige Links umgehend entfernen.</p>

224:       <p><b>Urheberrecht</b></p>

225:       <p>Die durch die Seitenbetreiber erstellten Inhalte und Werke auf diesen Seiten unterliegen dem deutschen Urheberrecht. Die Vervielf&auml;ltigung, Bearbeitung, Verbreitung und jede Art der Verwertung au&szlig;erhalb der Grenzen des Urheberrechtes bed&uuml;rfen der schriftlichen Zustimmung des jeweiligen Autors bzw. Erstellers. Downloads und Kopien dieser Seite sind nur f&uuml;r den privaten, nicht kommerziellen Gebrauch gestattet. Soweit die Inhalte auf dieser Seite nicht vom Betreiber erstellt wurden, werden die Urheberrechte Dritter beachtet. Insbesondere werden Inhalte Dritter als solche gekennzeichnet. Sollten Sie trotzdem auf eine Urheberrechtsverletzung aufmerksam werden, bitten wir um einen entsprechenden Hinweis. Bei Bekanntwerden von Rechtsverletzungen werden wir derartige Inhalte umgehend entfernen.</p>

226:

227:

228:       <p><b>Datenschutzerkl&auml;rung:</b><br />

229:       <b>Datenschutz</b></p>

230:       <p>Die Nutzung unserer Webseite ist in der Regel ohne Angabe personenbezogener Daten m&ouml;glich. Soweit auf unseren Seiten personenbezogene Daten (beispielsweise Name, Anschrift oder eMail-Adressen) erhoben werden, erfolgt dies, soweit m&ouml;glich, stets auf freiwilliger Basis. Diese Daten werden ohne Ihre ausdr&uuml;ckliche Zustimmung nicht an Dritte weitergegeben.

231:       Wir weisen darauf hin, dass die Daten&uuml;bertragung im Internet (z.B. bei der Kommunikation per E-Mail) Sicherheitsl&uuml;cken aufweisen kann. Ein l&uuml;ckenloser Schutz der Daten vor dem Zugriff durch Dritte ist nicht m&ouml;glich.

232:       Der Nutzung von im Rahmen der Impressumspflicht ver&ouml;ffentlichten Kontaktdataen durch Dritte zur &uuml;berseitung von nicht ausdr&uuml;cklich angeforderter Werbung und Informationsmaterialien wird hiermit ausdr&uuml;cklich widersprochen. Die Betreiber der Seiten behalten sich ausdr&uuml;cklich rechtliche Schritte im Falle der unverlangten Zusendung von Werbeinformationen, etwa durch Spam-Mails, vor.</p>

233:

234:       <p><b>Auskunft, L&ouml;schung, Sperrung</b></p>

235:       <p>Sie haben jederzeit das Recht auf unentgeltliche Auskunft &uuml;ber Ihre gespeicherten personenbezogenen Daten, deren Herkunft und Empf&auml;nger und den Zweck der Datenverarbeitung sowie ein Recht auf Berichtigung, Sperrung oder L&ouml;schung dieser Daten. Hierzu sowie zu weiteren Fragen zum Thema personenbezogene Daten k&ouml;nnen Sie sich jederzeit &uuml;ber die im Impressum angegeben Adresse des Seitenbetreibers an uns wenden.</p>]]>

236:       </string>

237:

238:

239:       </resources>

**./values/styles.xml**

**Sat Jan 17 12:38:23 2015**

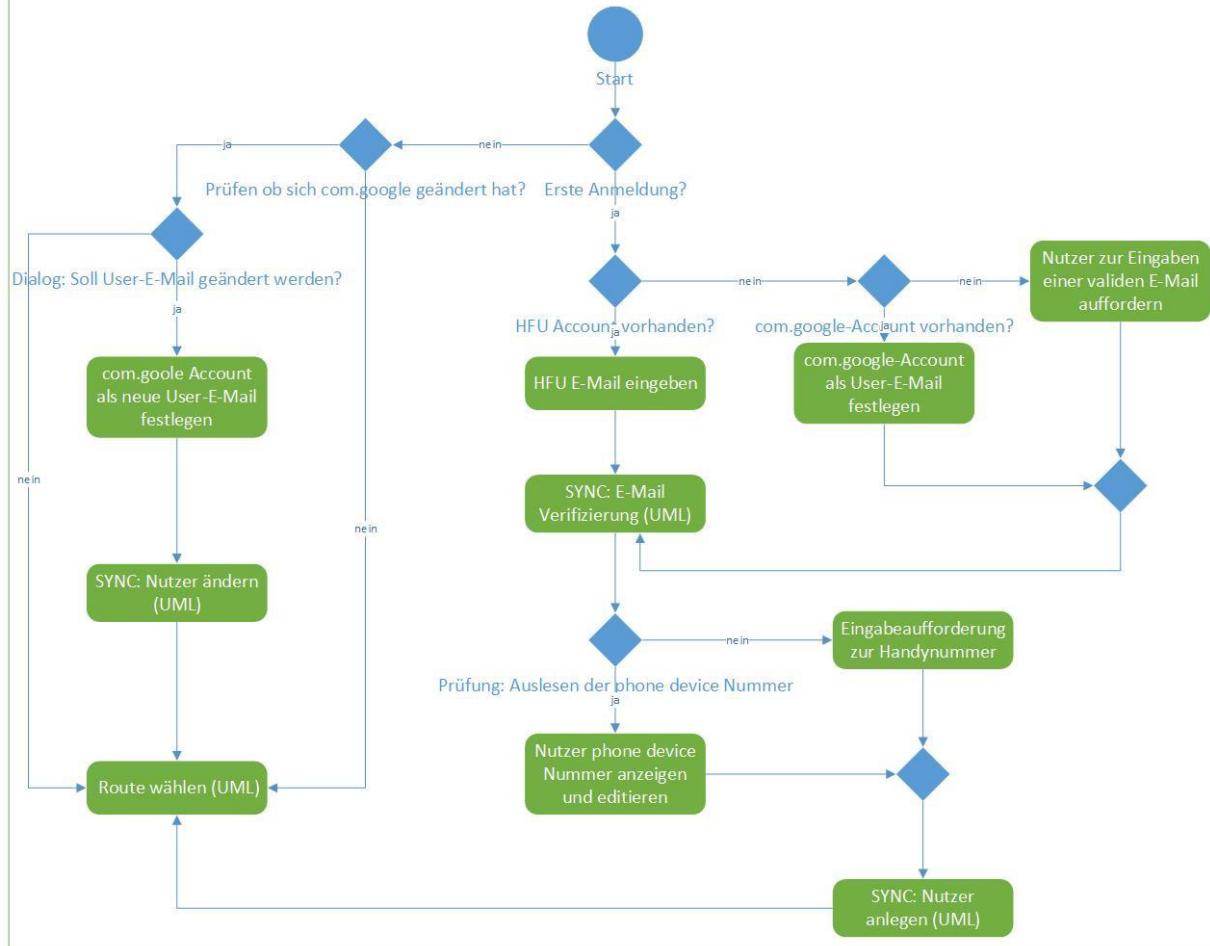
**1**

```
1: <resources xmlns:android="http://schemas.android.com/apk/res/android">
2:     <style name="HFUwerdMobil" parent="android:Theme.Holo.Light">
3:         <item name="android:textColor">@color/text</item>
4:         <item name="android:spinnerItemStyle">@style/spinnerItemStyle</item>
5:     </style>
6:
7:     <style name="spinnerItemStyle">
8:         <item name="android:textSize">25sp</item>
9:     </style>
10: </resources>
```

```
1: <?xml version="1.0" encoding="utf-8"?>
2: <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3:   package="com.HFUwerdMobil.main"
4:   android:versionCode="1"
5:   android:versionName="1.0" >
6:
7:     <!-- 14 statt 11, wegen OBD-Switch -->
8:     <!-- 15 statt, 15, wegen BluetoothActivity getUuids -->
9:   <uses-sdk
10:     android:minSdkVersion="15"
11:     android:targetSdkVersion="21" />
12:   <uses-permission android:name="android.permission.GET_ACCOUNTS"/>
13:   <uses-permission android:name="android.permission.READ_PHONE_STATE" />
14:   <uses-permission android:name="android.permission.INTERNET" />
15:   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
16:   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
17:   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
18:   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
19:   <uses-permission android:name="android.permission.BLUETOOTH"/>
20:   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
21:   <uses-feature
22:     android:glEsVersion="0x00020000"
23:     android:required="true"/>
24:
25:   <application
26:     android:allowBackup="true"
27:     android:icon="@drawable/ic_launcher"
28:     android:label="@string/app_name"
29:     android:theme="@style/HFUwerdMobil" >
30:     <meta-data
31:       android:name="com.google.android.maps.v2.API_KEY"
32:       android:value="AIzaSyAa40q6eLgwQsFcD9EQOborhhINDhkrYQU"/>
33:
34:     <meta-data android:name="com.google.android.gms.version"
35:       android:value="@integer/google_play_services_version" />
36:     <activity
37:       android:name="com.HFUwerdMobil.activity.LoginActivity"
38:       android:label="@string/authentification" >
39:       <intent-filter>
40:         <action android:name="android.intent.action.MAIN" />
41:
42:         <category android:name="android.intent.category.LAUNCHER" />
43:       </intent-filter>
44:     </activity>
45:     <service android:name="com.HFUwerdMobil.service.GPSListener"></service>
46:     <service android:name="com.HFUwerdMobil.obd2.OBDService"></service>
47:     <activity android:name="com.HFUwerdMobil.activity.ChooseRouteActivity" android:label="@string/choose_route"></a
ctivity>
48:     <activity android:name="com.HFUwerdMobil.activity.RecordRouteActivity" android:label="@string/record_route"></a
```

```
ctivity>
 49:           <activity android:name="com.HFUwerdMobil.activity.RouteDetailsActivity" android:label="@string/route_details" a
ndroid:parentActivityName="com.HFUwerdMobil.activity.ChooseRouteActivity">
 50:               <meta-data android:name="android.support.PARENT_ACTIVITY" android:value="com.HFUwerdMobil.activity.ChooseR
outeActivity" />
 51:           </activity>
 52:           <activity android:name="com.HFUwerdMobil.activity.AboutActivity" android:label="@string/about"></activity>
 53:           <activity android:name="com.HFUwerdMobil.activity.HelpActivity" android:label="@string/help"></activity>
 54:           <activity android:name="com.HFUwerdMobil.activity.ImprintActivity" android:label="@string/imprint"></activity>
 55:           <activity android:name="com.HFUwerdMobil.activity.UserVerifyActivity" android:label="@string/user_verify" andro
id:parentActivityName="com.HFUwerdMobil.activity.LoginActivity" android:noHistory="true">
 56:               <meta-data android:name="android.support.PARENT_ACTIVITY" android:value="com.HFUwerdMobil.activity.LoginAc
tivity" />
 57:           </activity>
 58:           <activity android:name="com.HFUwerdMobil.activity.SettingsActivity" android:label="@string/settings" android:pa
rentActivityName="com.HFUwerdMobil.activity.ChooseRouteActivity">
 59:               <meta-data android:name="android.support.PARENT_ACTIVITY" android:value="com.HFUwerdMobil.activity.Choose
RouteActivity" />
 60:           </activity>
 61:           <activity android:name="com.HFUwerdMobil.obd2.BluetoothActivity" android:label="@string/bluetooth" >
 62:               </activity>
 63:           </application>
 64:
 65: </manifest>
```

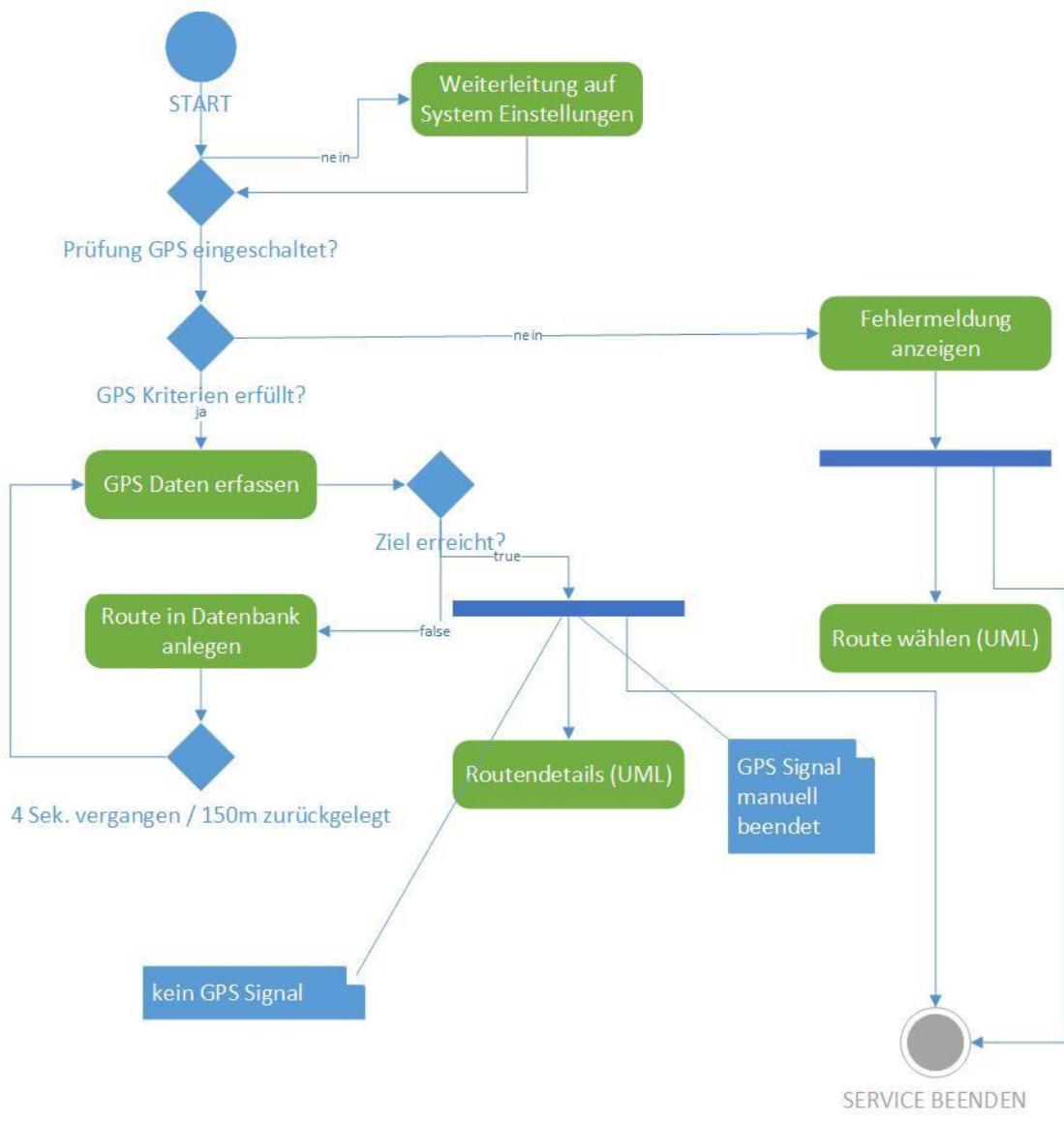
## Authentifizierung und Authorisierung



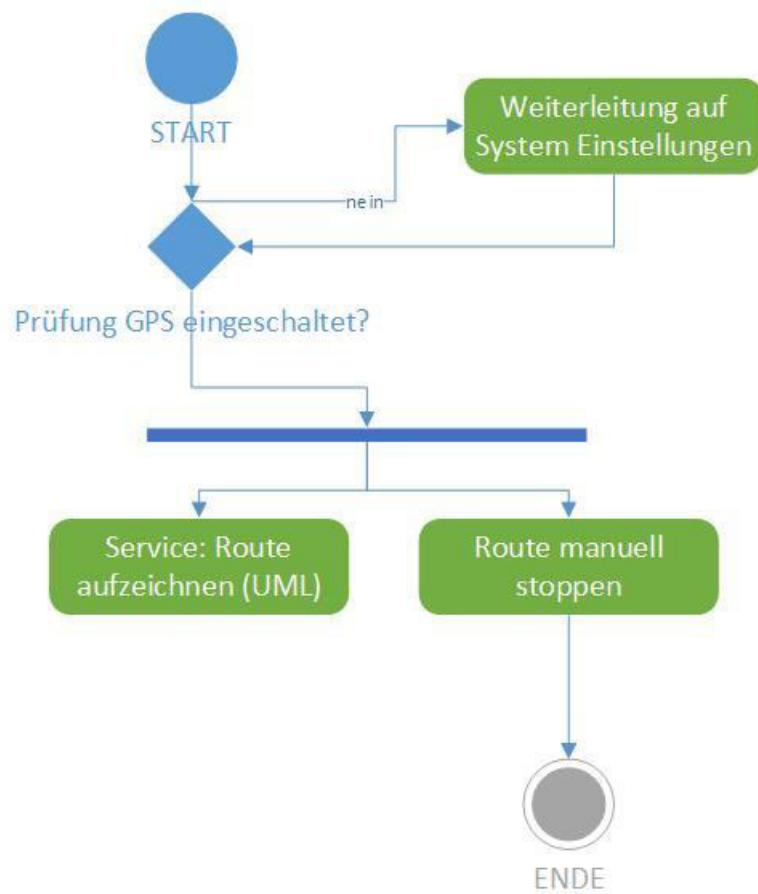
## Benutzereinstellungen verwalten



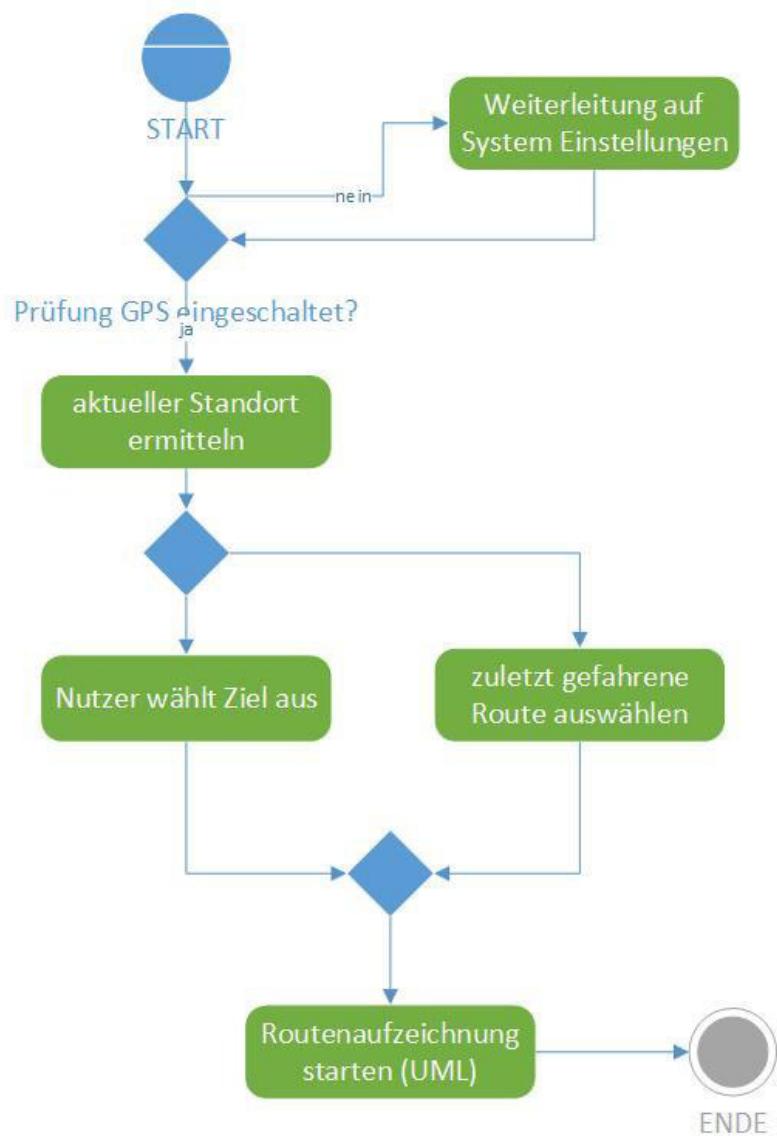
Service: Route aufzeichnen



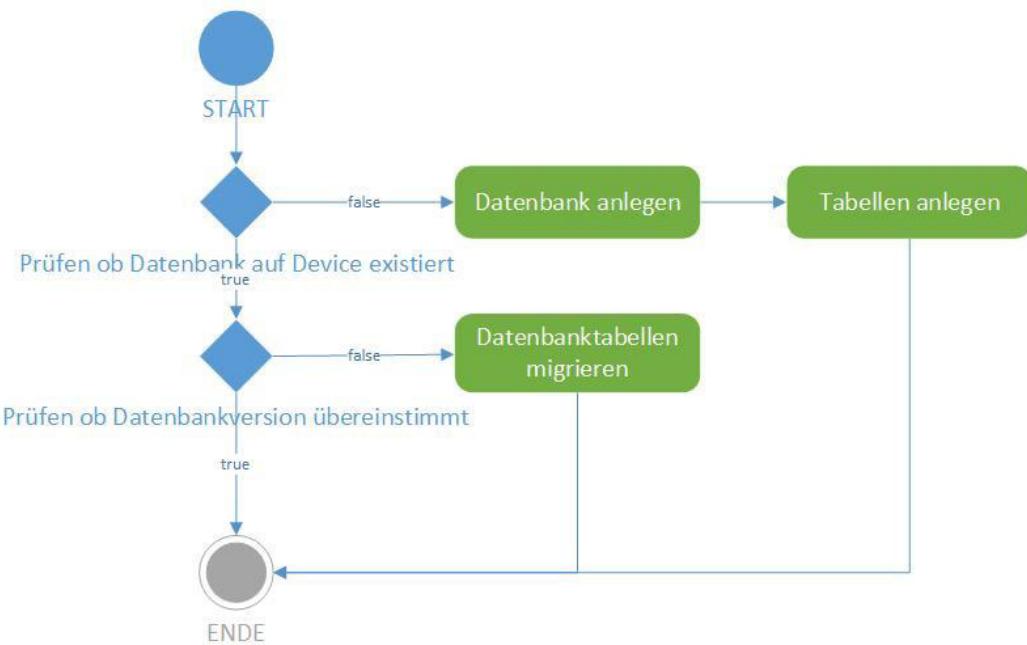
## Route aufzeichnen



## Route wählen



## Datenbank anlegen/migrieren



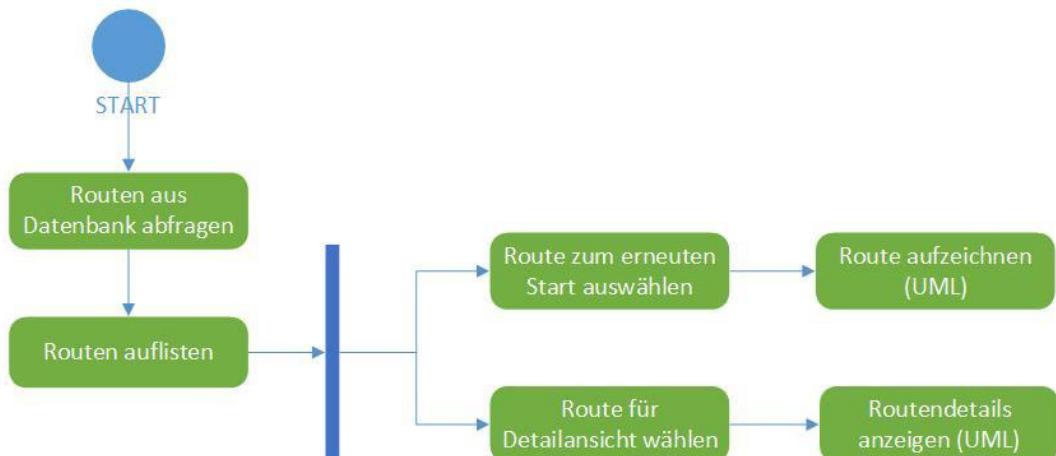
### Routendetails anzeigen



### About anzeigen



### Gefahrene Routen anzeigen



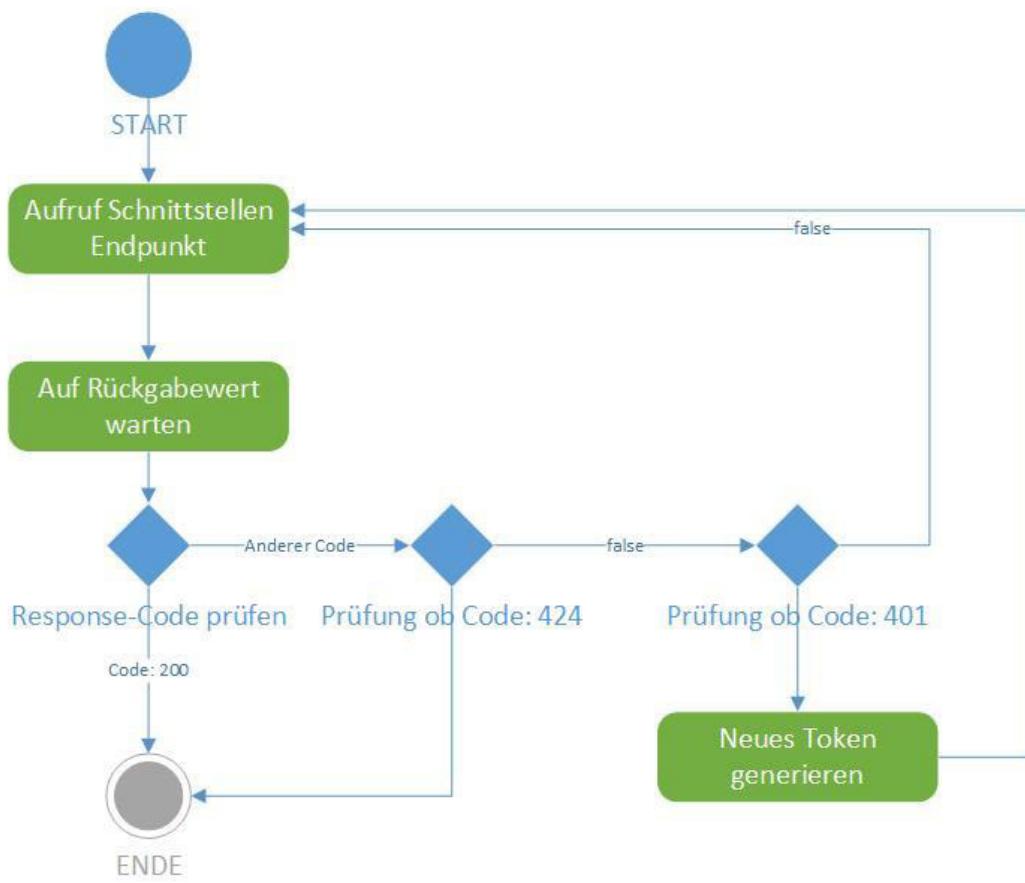
### Hilfe anzeigen



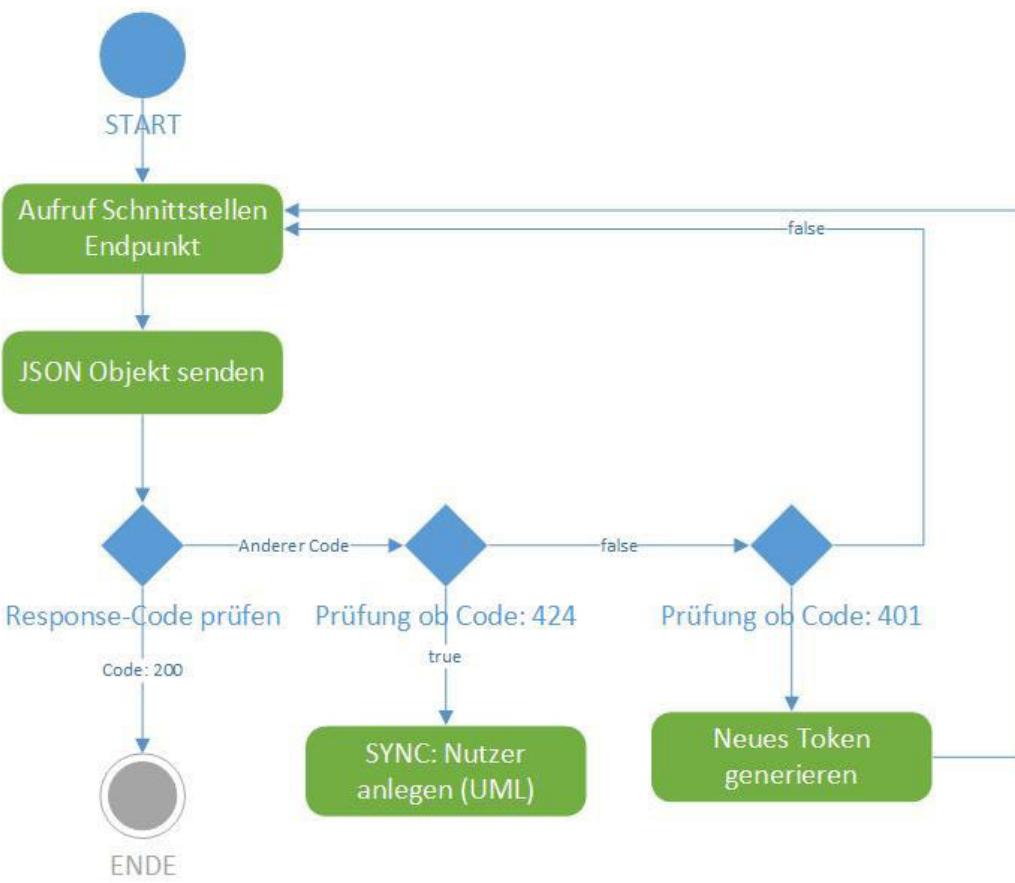
### Impressum anzeigen



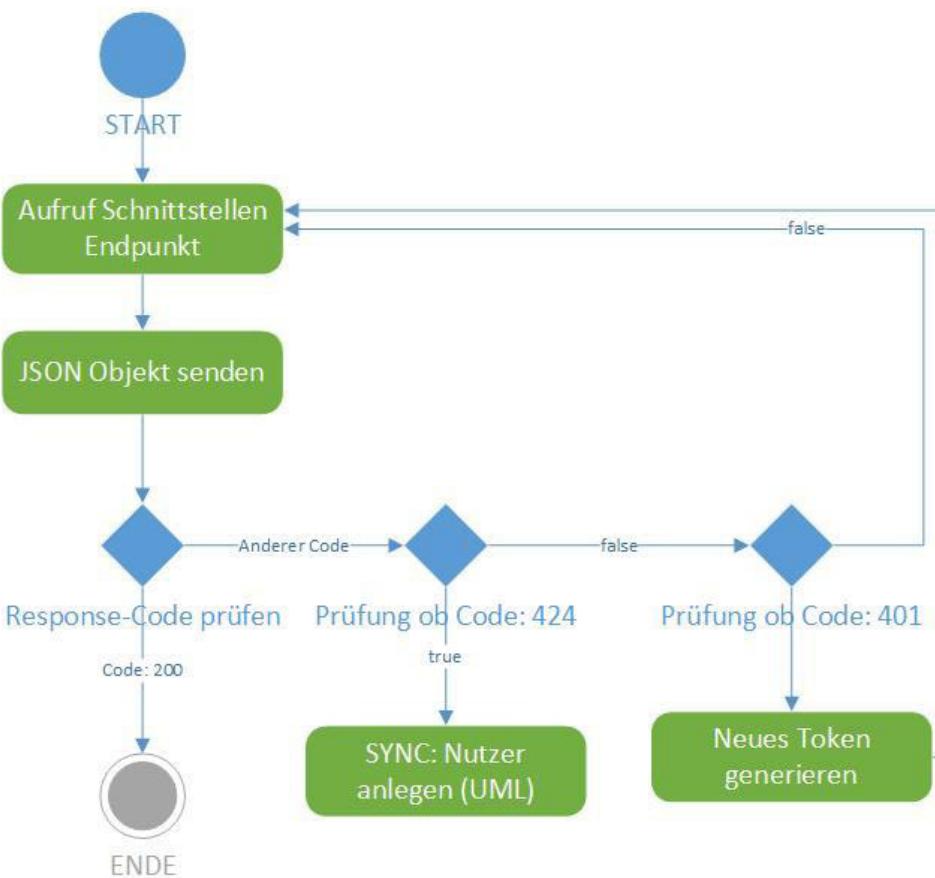
### SYNC: Daten abrufen



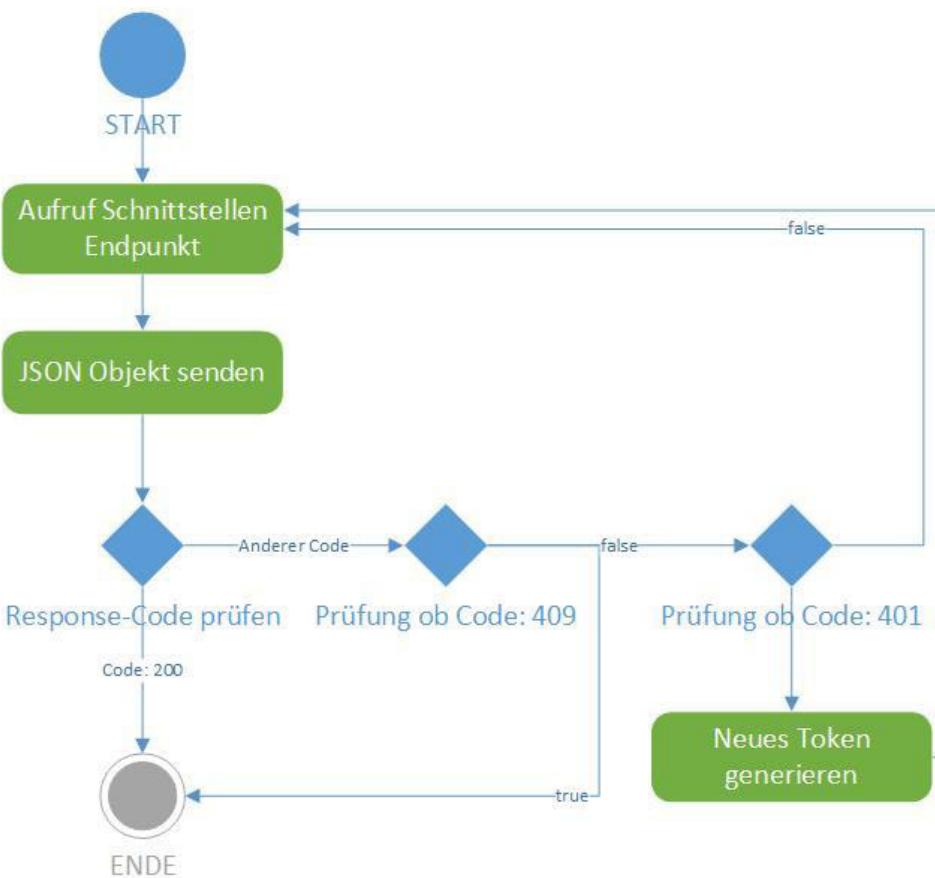
### SYNC: GPS-Daten der Route hinzufügen



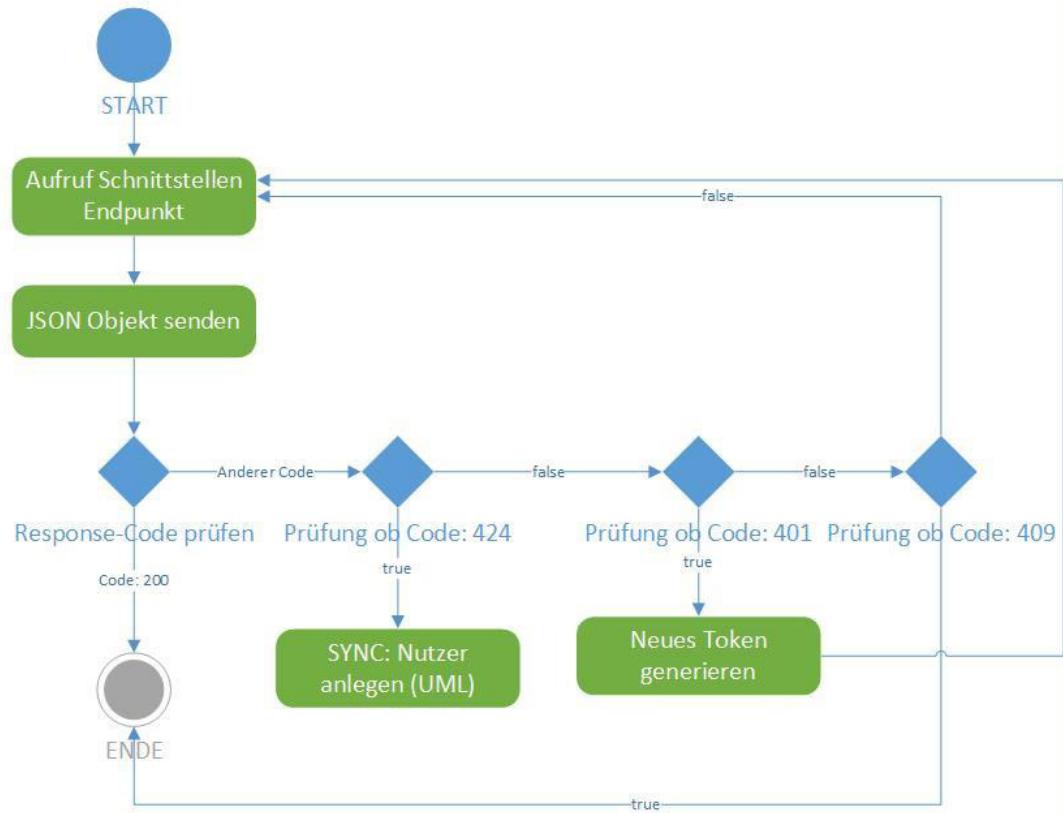
### SYNC: User ändern



### SYNC: User anlegen



### SYNC: Neue Route anlegen



### SYNC: Route beenden

