ulm university universität
uulm

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme (DBIS)

Konzeption und Implementierung einer mobilen Anwendung zu Unterstützung des Fahrstils am Beispiel des Apple iOS und Google Firebase

Masterarbeit

Vorgelegt von:
Ryad Guerroudj
ryad.guerroudj@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert
Dr. Rüdiger Pryss

Betreuer:
Dr. Rüdiger Pryss

2018

Fassung 22. Dezember 2018

Inhaltsverzeichnis

1 Einleitung

Der Straßenverkehr spielt in unserem Leben eine bedeutende Rolle. Nicht nur, um Ziele schnell und bequem zu erreichen, sondern auch, um Güter zu transportieren. Laut Statistischem Bundesamt waren in Deutschland im Jahr 2015 mehr als 56 Millionen Kraftfahrzeuge zugelassen. Den größten Anteil bildeten die Pkw mit mehr als 45 Millionen, gefolgt von mehr als drei Millionen Lkw. Jedes Fahrzeug legte hierbei durchschnittlich 14.000 Kilometer zurück [1].

Diese Zahlen verdeutlichen, dass jedes Jahr Milliarden Kilometer gefahren werden. Dies bringt aber auch Nachteile mit sich. Rund 366.448 Verkehrsunfälle mit Personenschaden ereigneten sich in Deutschland im Jahr 2015, das entspricht einem Zuwachs von 1,2 Prozent gegenüber 2014. Dabei kamen 3.459 Menschen ums Leben. Mit 86 Prozent ist menschliches Fehlverhalten mit großem Abstand die vorherrschende Ursache der Verkehrsunfälle, wobei die falsche Auswahl der Fahrgeschwindigkeit ganz oben auf der Liste steht [1].

Der Verbrauch von Kraftstoff für Personenkraftwagen in Deutschland lag im Jahr 2015 bei 13.877 Millionen Liter Dieselkraftstoff und 23.460 Millionen Liter Ottokraftstoff [2]. Offiziellen Angaben zufolge verursachte allein der Pkw-Verkehr in Deutschland im Jahr 2015 rund 112,3 Millionen Tonnen des Treibhausgases Kohlendioxid (CO2) in der Luft [2]. Der Gesetzgeber verpflichtet deshalb die Hersteller der Automobilindustrie, zum Schutze der Umwelt einen CO2-Emissionen-Standardtest zu bestehen, bevor ein Modell zugelassen wird. Dabei sind die Ergebnisse aus der spezifischen Testumgebung nicht unbedingt identisch mit denen aus der realen Umwelt.

1.1 Motivation

Unfälle mit Personen- oder Sachschaden und ein gestiegener Kraftstoffverbrauch beziehungsweise erhöhte CO2-Emissionen könnten durch ein optimiertes Fahrerverhalten vermieden werden.

Laut einer Studie von Kumar, kennen 19 Prozent der Fahrer die Geschwindigkeitsgrenze oder ihre eigene Geschwindigkeit nicht [3]. Dies kann riskantes Verhalten mit sich bringen, wie zum Beispiel unangebrachtes Beschleunigen [4]. Eine Erhöhung der Durchschnittsgeschwindigkeit um 1,6 km/h führt zu einem Anstieg der Unfallzahlen um 5 Prozent [3]. 40 Prozent der befragten 25 Probanden geben an, dass sie manchmal überrascht sind, dass das Tempolimit anders ist als das von ihnen angenommene. 68 Prozent geben an, dass sie manchmal unbeabsichtigt über ihrer Wunschgeschwindigkeit fahren [3].

In der Studie wird zum einen die Fähigkeit der Fahrer untersucht, Ratschlägen zu folgen. Zum anderen konzentriert sie sich darauf, wie man Fahrer dazu bewegen kann, sicheres Fahren zu priorisieren. Es wurde festgestellt, dass jede Art von Ratschlägen zur sicheren Lenkung die Leistung verbesserte [5]. Dabei erweist sich kontinuierliches visuelles Feedback in Echtzeit als effektiver gegenüber akustischem oder akkumuliertem Feedback. Als Nachteil reduziert diese Modalität die Aufmerksamkeit auf die Vorwärtsansicht und erhöht die subjektive Arbeitsbelastung [6]. Es muss ein Kompromiss zwischen der Bereitstellung von Umgebungsinformationen und der visuellen Ablenkung des Fahrers gefunden werden.

Das Bewusstsein für treibstoffeffiziente Fahrtechniken hat sowohl im öffentlichen als auch im kommerziellen Bereich an Popularität gewonnen. Evaluierte visuelle Eco-Driving-Feedback-Systeme haben gezeigt, dass Eco-Drive, ob durch finanzielle oder ökologische Gründe motiviert, das Potenzial hat, die Produktion von Treibhausgasen deutlich zu reduzieren [6]. Eine Reihe von Faktoren können den tatsächlichen Kraftstoffverbrauch beeinflussen, zum Beispiel Außentemperatur, Fahrzeuglast und Verwendung von Hilfssystemen wie Klimaanlagen. Den individuellen Fahrstil zu beeinflussen ist ein sinnvoller Ansatz. Doch Studien zufolge kann unabhängig vom Fahrzeug und von der Technologie an Bord der individuelle Fahrstil signifikanten Einfluss auf die Kraftstoffeffizienz ausüben. Van der Voort beschreibt, dass durch eine Verbesserung des Fahrstils 10–15 Prozent Kraftstoffeinsparungen beim Pkw erreicht werden könnten [7]. Diese Einsparungen könnten sogar ohne einen starken Anstieg der Reisezeiten erreicht werden [8]. Für Nutzfahrzeuge und Lastkraftwagen mit längeren Fahrstrecken wäre eine solche Einsparung noch bedeutender. Daher werden spezielle Schulungen für Lkw-Fahrer angeboten. Einige Studien haben belegt, dass Fahrer von Kraftfahrzeugen in den ersten Monaten gute Einsparungen erreichen können, mit der Zeit aber zu ihren alten Fahrmustern zurückkehren.

1.2 Ziel

Bei der Mobile-Anwendungsentwicklung müssen viele Facetten berücksichtigt werden und ist deshalb nicht einfach [9]. Daher ist dieses Projekt anspruchsvoll. Mit dieser Arbeit wird mit Hilfe von digitalen Technologien und von Gamifizierung ein Ansatz zur Verbesserung der primären Fahraufgabe gezeigt. Es wird eine iOS-App entwickelt, die anhand verschiedener Leistungskriterien die Nutzung des Fahrzeuges überwacht, analysiert und dem Fahrer in Echtzeit Feedback zur Verbesserung des Fahrstils gibt. Ziel ist es, dadurch eine möglichst sichere und effiziente Fahrweise zu erreichen. Um die Nutzung eines Fahrzeuges zu erfassen, wird eine OBD2 Schnittstelle verwendet. Darin sind die wichtigen Kenndaten definiert und an einer frei zugänglichen Schnittstelle zur Verfügung gestellt. Zusätzlich werden die Google-Map-API benutzt, um die statischen Eigenschaften der Route wie die maximale zulässige Fahrgeschwindigkeit zu erfassen; solche Daten ermöglichen eine objektive Bewertung des Fahrerverhaltens.

2 Grundlagen

Im Verlauf dieses Kapitels wird an das Themenfeld der Fahrdynamik herangeführt. Anschließend werden die Daten ermittelt, die als Grundbasis für die Bewertung des Fahrstils von Bedeutung sind. Um dies an praktischen Beispielen zu verdeutlichen, werden weiterhin vier mobile Anwendungen mit verschiedenen Feedback-Modalitäten vorgestellt.

2.1 Fahrzeugzustand

Jeder Fahrzeugmotor hat ein ähnliches Diagramm wie in Abbildung 2.1 dargestellt. Hier ist zu erkennen, dass der maximale Wirkungsgrad beziehungsweise der niedrigste Kraftstoffverbrauch durch Fahren unter Teillast im möglichst hohen Getriebegang in einem bestimmten Drehzahlbereich erreicht wird.

Abhängig von der jeweiligen Gaspedalstellung und dem eingelegten Gang errechnet sich die Motorleistung aus der Drehzahl und dem Drehmoment. Solange die produzierte Leistung eine Kraft erzeugt, die größer ist als die der Fahrwiderstände, wird das Fahrzeug in Bewegung gesetzt.

2.2 Fahrwiderstände

Während der Fahrt eines Kraftfahrzeuges treten diverse Widerstände auf, die eine bestimmte Motorleistung erfordern, um das Fahrzeug in Bewegung zu setzen. Diese sind der Luftwiderstand, der Rollwiderstand, der Beschleunigungswiderstand und der Steigungswiderstand, vergleiche Abbildung 2.2.

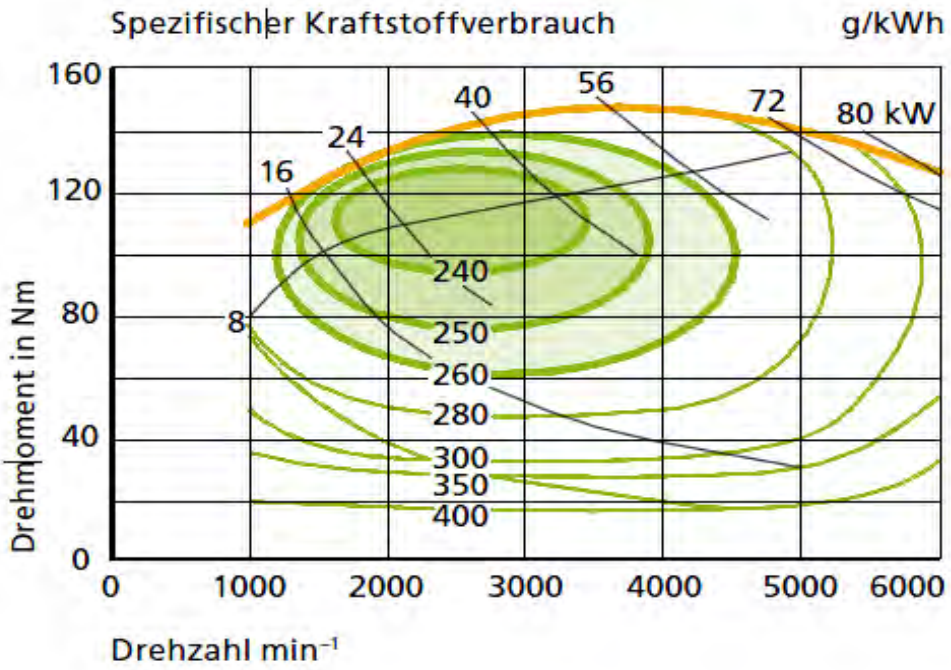

Abbildung 2.1: Verbrauchskennfeld bei Benzinmotor 1,8 Liter [10].

2.2.1 Luftwiderstand

Der am stärksten wirkende Widerstand für ein fahrendes Kraftfahrzeug ist der Luftwiderstand. Die Formel 2.1 verdeutlicht die Wirkung der verschiedenen Faktoren (Fahrzeugstirnfläche A, Strömungswiderstand, Luftwiderstandsbeiwert c, Geschwindigkeit v und Dichte) auf den Luftwiderstand, wobei zu erkennen ist, dass die Geschwindigkeit im Gegensatz zu allen anderen Faktoren potenziell wirkt. Bei doppelter Geschwindigkeit vervierfacht sich der Luftwiderstand.

$$W_{Luft} = \frac{\rho}{2} * c * A * v^2 \tag{2.1}$$

2.2.2 Rollwiderstand

Mit zunehmender Geschwindigkeit steigt nicht nur der Luftwiderstand, sondern auch der Rollwiderstand und infolgedessen der Kraftstoffverbrauch nehmen zu. Bei doppelter Geschwindigkeit verdoppelt sich der Rollwiderstand gemäß Formel 2.2. Der Rollwiderstand hängt von der Fahrzeugmasse m, der Erdbeschleunigung g, dem Rollwiderstandsbeiwert f und dem Steigungswinkel ab.

$$W_{Roll} = m * g * f * cos(\alpha) \tag{2.2}$$

2.2.3 Beschleunigungswiderstand

Um die Geschwindigkeit eines Fahrzeuges zu erhöhen, wird zusätzliche Kraft benötigt. Dabei entsteht ein Beschleunigungswiderstand, der von Fahrzeugmasse und Geschwindigkeitserhöhung abhängt. Die Formel 2.3 legt dar, dass mit zunehmender Fahrzeugmasse m auch mehr Kraft F zur Geschwindigkeitserhöhung benötigt wird.

$$W_{Beschleunigung} = m * \frac{\Delta v}{\Delta t} \tag{2.3}$$

$$v_2 = v_1 + (\frac{F}{m} * (t2 - t1)) \tag{2.4}$$

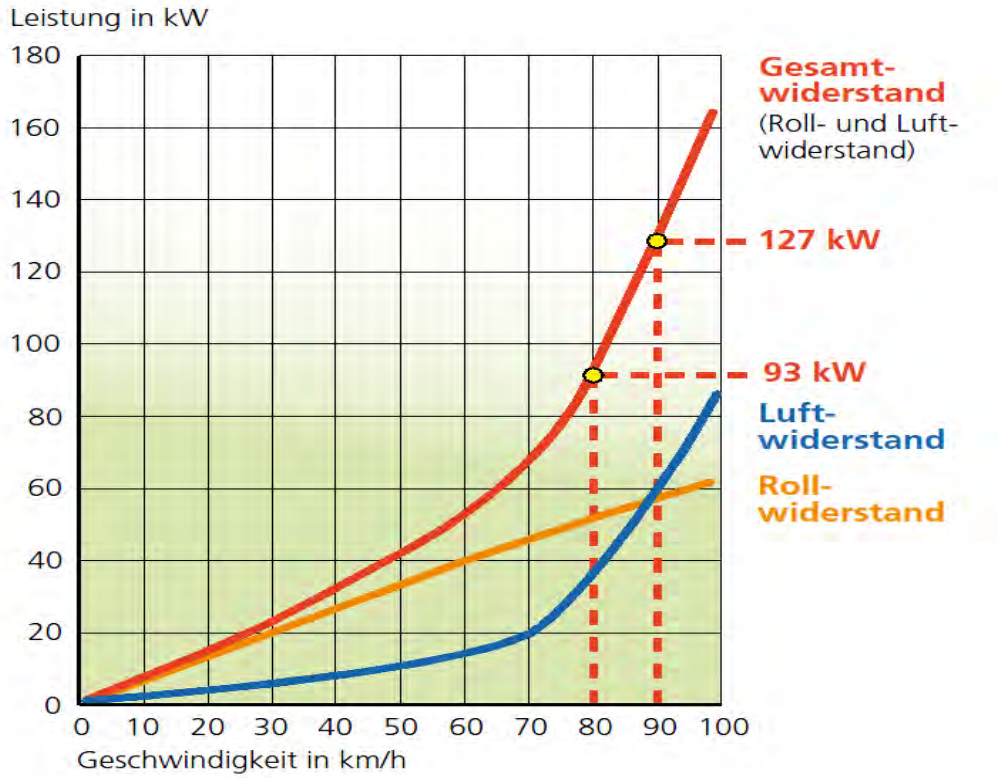

Abbildung 2.2: Luft- & Rollwiderstand [10].

2.2.4 Steigungswiderstand

Bei Bergfahrt wird zusätzliche Kraftstoffenergie benötigt, die von der Fahrzeugmasse m und der Schwerkraft der Erde g abhängt, vergleiche Formel 2.5.

$$W_{Steigung} = m * g * sin(\alpha) \tag{2.5}$$

2.3 Fahrzeugdaten

Im Verlauf dieses Kapitels wird die OBD2 Schnittstelle vorgestellt. Anschließend werden die Methoden für die Gangerkennung und die momentane Kraftstoffverbrauchsberechnung beschrieben.

2.3.1 OnBoard-Diagnose

OBD steht für OnBoard-Diagnose. Die Schnittstelle wurde 1988 von der California Air Ressource Board eingeführt. Seit 2001 gibt es OBD auch in Europa. Das Ziel von OBD ist die kontinuierliche Überwachung der abgasbeeinflussenden Systeme [11]. Dazu zählen:

- Bereitstellung definierter Signale
- Signalisierung und Speicherung von auftretenden abgasrelevanten Fehlern
- Ansteuerung der Motorkontrollleuchte

OBD unterstützt zehn verschiedene Modi, die auf verschiedene Methoden zurückgreifen, um mit dem Fahrzeug zu interagieren.

Modus	Beschreibung
1	Gibt Live-Daten zurück.
2	Gibt momentane Daten eines Fehlers (freeze frame) an. Wenn ein Fehler vom Steuergerät erkannt wird, werden die Sensordaten zum Fehlerzeitpunkt aufgezeichnet.
3	Dieser Modus zeigt die gespeicherten Diagnosefehlercodes an. Diese Fehlercodes gehören zum Standard für alle Fahrzeugmarken und sind in 4 Kategorien unterteilt: P0xxx: für Standardwerte, die mit dem Antrieb verbunden sind (Motor und Getriebe), C0xxx: für Standardfehler am Chassis, B0xxx: für Standardfehler an der Karosserie.
4	In diesem Modus werden aufgezeichnete Fehlercodes gelöscht und die Motorfehleranzeige ausgeschaltet. Hinweis: Meist ist es nicht notwendig, einen Fehler zu beheben, der nicht diagnostiziert oder repariert wurde. Die MIL leuchtet während des nächsten Fahrzyklus wieder auf.
5	Dieser Modus gibt die Ergebnisse der Selbstdiagnose an den Sauerstoff-/Lambda-Sensoren an. Er betrifft überwiegend Benzinfahrzeuge. Bei neuen Steuergeräten mit CAN wird dieser Modus nicht mehr verwendet. Modus 6 ersetzt die Funktionen, die in Modus 5 verfügbar waren.
6	Dieser Modus gibt die Ergebnisse der Selbstdiagnose von Systemen an, die keiner ständigen Überwachung unterliegen.
7	Dieser Modus liefert unbestätigte Fehlercodes. Nach einer Reparatur ist es sehr hilfreich zu überprüfen, ob der Fehlercode erneut angezeigt wird, ohne einen langen Testlauf durchführen zu müssen. Die verwendeten Codes sind identisch mit denen in Modus 3
8	Dieser Modus gibt die Ergebnisse der Selbstdiagnose auf anderen Systemen an. Er wird in Europa kaum verwendet.
9	9 Dieser Modus gibt die Informationen über das Fahrzeug an, wie zum Beispiel: die VIN (Fahrzeugidentifikationsnummer), Kalibrierungswerte.
10	Dieser Modus gibt die permanenten Fehlercodes an. Die verwendeten Codes sind mit denen in den Modi 3 und 7 identisch. Im Gegensatz zu den Modi 3 und 7 können diese Codes nicht im Modus 4 gelöscht werden. Nur mehrere Straßenzyklen ohne Auftreten des Problems können den Fehler beheben.

Tabelle 2.1: OBD2-Modi

Im Rahmen dieser Arbeit wird Mode 1 benutzt. Hierbei werden Echtzeit-Daten ermittelt. Nicht alle verfügbaren und definierten PID werden auch wirklich vom Fahrzeug unterstützt. Um herauszufinden, auf welche PID das Fahrzeug antworten wird, gibt es eine Auslesefunktion [12].

Über Reizung der PID werden vier Datenbytes zurückgegeben. Jedes der 32 Bits defi-

niert, welche der nächsten 32 PID vom Fahrzeug unterstützt werden. Jeweils das letzte Bit sagt aus, ob 32 weitere PID unterstützt werden.

Beispiel: Anreizung mit der Frage: 01 00 gibt die Antwort: 41 00 BE 1F A8 13.

- 01: Frage in Modus 1
- 00: angefragte PID
- 41: Antwort in Modus 1
- BE 1F A8 13: Antwort Datenbytes

In diesem Fall werden also noch weitere der nächsten 32 PID unterstützt.

PID(HEX)	PID(Dec)	Bytes returned	Description	Units
0	00	4	supported [01-20]	
0C	12	2	Engine RPM	rpm
0D	13	1	Vehicle speed	km/h

Tabelle 2.2: OBD2-Interpretation

2.3.2 Schaltanalyse

Der gewählte Gang ist nicht per OBD auslesbar und muss aus den Fahrzeugdaten berechnet werden. Die Gänge 3, 4, 5, 6 und gegebenenfalls 7 sind gut durch das Verhältnis Gangübersetzung = Motordrehzahl/Fahrzeuggeschwindigkeit gekennzeichnet. Der erste und der zweite Gang werden im Normalfall nur kurz beim Beschleunigen gefahren und können mit dem oben genannten Verhältnis nicht robust detektiert werden. Hier ist eine genauere Betrachtung der Motorlast und des Gaspedaltimings notwendig. Eine entsprechende Umrechnungstabelle von Motordrehzahl/Fahrzeuggeschwindigkeit wird von der App mitgeloggt, um die Getriebeübersetzung zu lernen.

2.3.3 Momentane Kraftstoffverbrauch-Berechnung

Wie oben erwähnt, werden nicht alle PID von OBD unterstützt. Dies ist herstellerabhängig und betrifft auch den momentanen Kraftstoffverbrauch. Es wird die folgende Verfahren für die Verbrauchsberechnung benutzt.

Immer wenn ein neuer Geschwindigkeitswert oder ein neuer Luftmassenmesserwert (MAF) hereinkommt, wird der Momentane Verbrauch bestimmt. Für ein Benzinfahrzeug gilt die folgende Formel:

$$fuelRate[L/100km] = fuelFlow/vehicleSpeed \tag{2.6}$$

$$fuelFlow[L/sec] = massAirFlow/(AFRactual * fuelDensity) \tag{2.7}$$

$$fuelDensity[gfuel/L] = 0,75[kgFuel/L] * 1000[gFuel/kgFuel] \tag{2.8}$$

Wobei:

- massAirflow [gAir/sec] lesbar mit Manufacturer specific PID oder mit PID(hex)= 0110
- vehicleSpeed [km/h] lesbar mit PID(hex)= 010D
- loadAbs lesbar mit PID(hex)= 0143
- engineSpeed lesbar mit PID(hex)= 010C
- AFRactual [gAir/gFuel]=14,7
- airDensity [gAir/L] =1,184

2.4 Fahrstil Feedback

In diesem Kapitel werden vier Systeme mit verschiedenen Feedback-Modalitäten präsentiert, die den Fahrer dazu bewegen sollen, sein Fahrverhalten zu ändern.

2.4.1 Dynamic Speedometer

Dynamic Speedometer ist ein Visualisierungssystem, das den Tachographen mit Informationen über die aktuelle maximale zulässige Geschwindigkeit erweitert, vergleiche 2.3. Damit wird angestrebt, das momentane Fahrverhalten des Fahrers durch sofortige Rückmeldung zu verändern. Im Falle einer Überschreitung der zulässigen Geschwindigkeit erhält der Fahrer visuelle und akustische Signale. Dies entlastet den Fahrer von

Abbildung 2.3: Speedometer erweitert den Tachographen mit Informationen über die aktuelle maximale zulässige Geschwindigkeit [3].

der Aufgabe des Suchens nach Verkehrszeichen, um die aktuelle Geschwindigkeitsbegrenzung zu bestimmen. Die Funktion ist persistent und integriert als Bestandteil des Tachographen, sodass der Fahrer von seiner Hauptaufgabe nicht abgelenkt wird.

2.4.2 CoastMaster

CoastMaster ist eine Smartphone-Anwendung, die als Fahrspielanzeige dient. Sie ist motiviert durch die Notwendigkeit, den Fahrer in Situationen wieder in die Fahraufgabe einzubinden, in denen das Manövrieren des Fahrzeugs einfach ist und kein hohes Maß an Engagement erfordert. Sie unterstützt den Fahrer bei Änderungen der Geschwindigkeitsbegrenzungen, und gibt ein Feedback zum Fahrverhalten. Im Rahmen CoastMaster Porjekt wurde in einer Studie zwischen eine Gamified- und eine Nicht-Gamified-Schnittstelle verglichen in Bezug auf Benutzererfahrung, Fahrleistung und visuelle Ablenkung.Die Ergebnisse deuten auf eine Steigerung der Fahrerbeteiligung sowie auf eine Abnahme der Geschwindigkeitsverletzungen hin. Gleichzeitig führt die Gamified-Version zu längeren Blicken auf das Display, was die visuelle Ablenkung erhöht [13].

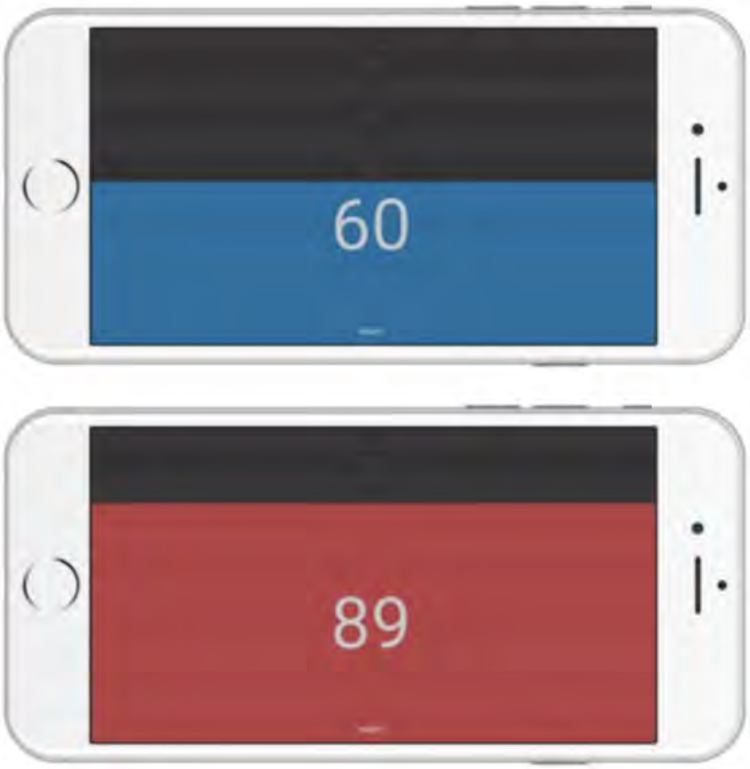

Abbildung 2.4: CocoaMaster - Spielerischer Ansatz für das Fahrstil-Feedback [13].

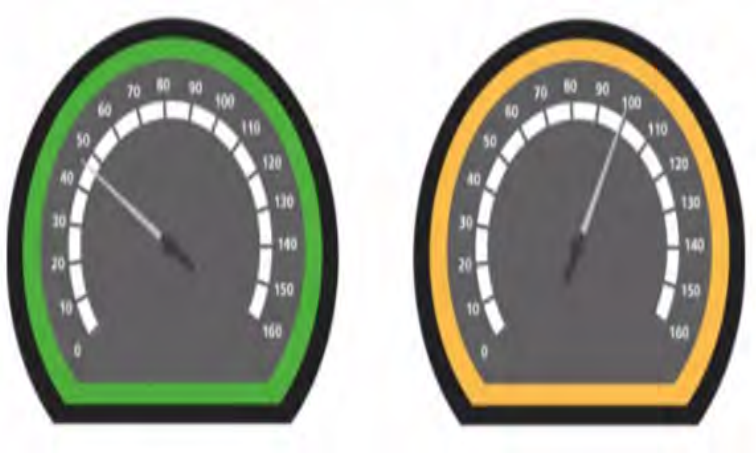

Abbildung 2.5: EcoSpeedometer - Leuchtet grün bei sparsamer Fahrt, orange bei ineffizienter Fahrt [14].

2.4.3 EcoSpeedometer

Das EcoSpeedometer bietet in Echtzeit eine treibstoffeffiziente Fahranleitung. Es handelt sich dabei um ein Display, das nahtlos in den traditionellen Tacho integriert ist und eine visuelle Rückmeldung dazu gibt, ob der Fahrer aktuell treibstoffsparend fährt. Bei sparsamer Fahrt leuchtet das EcoSpeedometer grün, bei ineffizienter Fahrt orange. Analog zum Speedometer ist es persistent und integriert als Bestandteil des Tachographen, sodass der Fahrer beim Fahren nicht abgelenkt wird. Die abstrakte Form des Feedbacks könnte jedoch andererseits das intuitive Erfassen eines kraftstoffsparenden Stils erschweren.

2.4.4 EcoScore

Das EcoScore-Display visualisiert die kumulative Kraftstoffeffizienz für die aktuelle Fahrt durch die Anzeige einer Reihe von grünen Bäumen, siehe Abbildung 2.6. Je effizienter die Fahrt ist, desto größer werden die Bäume dargestellt. Die Fahrstilanalyse ermittelt dazu aus den drei Komponenten Beschleunigung, gleichmäßige Fahrweise und Ausrollen einen Punktwert zwischen 0 und 100, der angibt, wie umweltverträglich der Fahrstil ist. Alle drei Werte haben direkte Auswirkungen auf Kraftstoffverbrauch und Schadstoffausstoß, und tragen indirekt zur Sicherheit im Straßenverkehr bei. Bei einem sehr niedrigen EcoScore erscheint eine Aufforderung auf dem Bildschirm, die Fahrweise anzupassen. EcoScore hat dasselbe Problem wie EcoSpeedometer und ist für den Fahrer möglicherweise nicht intuitiv erfassbar.

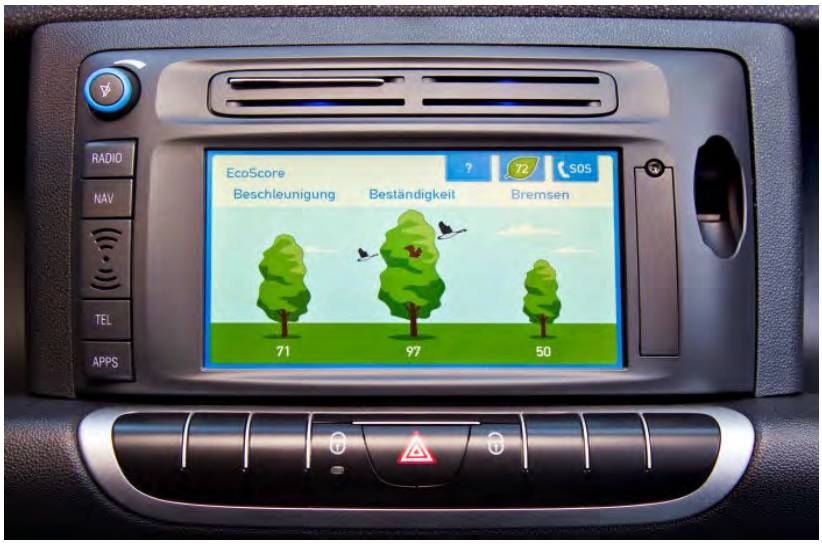

Abbildung 2.6: EcoScore - Im Gegensatz zum EcoSpeedometer reagiert das EcoDisplay nicht sofort auf das Fahrverhalten, sondern zeigt die Kraftstoffeffizienz für einen bestimmten Zeitraum an – in diesem Fall für die Dauer der letzten Fahrt. Es zielt daher auf eine langfristige Verhaltensänderung ab, indem Informationen über einen längeren Fahrzyklus bereitgestellt werden [15].

3 Anforderungsdefinition

Die Anforderungen wurden aus der Analyse in Kapitel 2 herausgearbeitet.

3.1 Funktionale Anforderungen

3.1.1 Authentifizierung

Die App soll die Identität eines Benutzers kennen. Dies ermöglicht, die Nutzerdaten sicher zu speichern und dieselben personalisierten Nutzererfahrungen auf allen Geräten bereitzustellen.

Title	Beschreibung
Registrieren	Benutzer muss sich registrieren, indem er eine gültige E-Mail-Adresse und ein validiertes Passwort wiederholend anlegt.
Passwort Validieren	Ein Passwort muss mindestens aus 8 Zeichen bestehen sowie eine Zahl und ein Sonderzeichen enthalten.
Bestätigungs-E-Mail	Nach erfolgreicher Eingabe der E-Mail-Adresse und des Passworts wird ein Link an die angegebene E-Mail-Adresse gesendet, um die Registrierung abzuschließen.
Passwort Zurücksetzen	Benutzer kann sein Passwort zurücksetzen, indem er seine E-Mail-Adresse eingibt. Ein Link wird an die angegebene E-Mail-Adresse gesendet, um das Passwort zurückzusetzen.
Anmelden	Benutzer muss eine gültige E-Mail-Adresse und ein Passwort eingeben, um sich anzumelden. Bei falscher Eingabe wird der Benutzer angewiesen, seine Eingabedaten zu korrigieren.

Tabelle 3.1: Authentifizierung

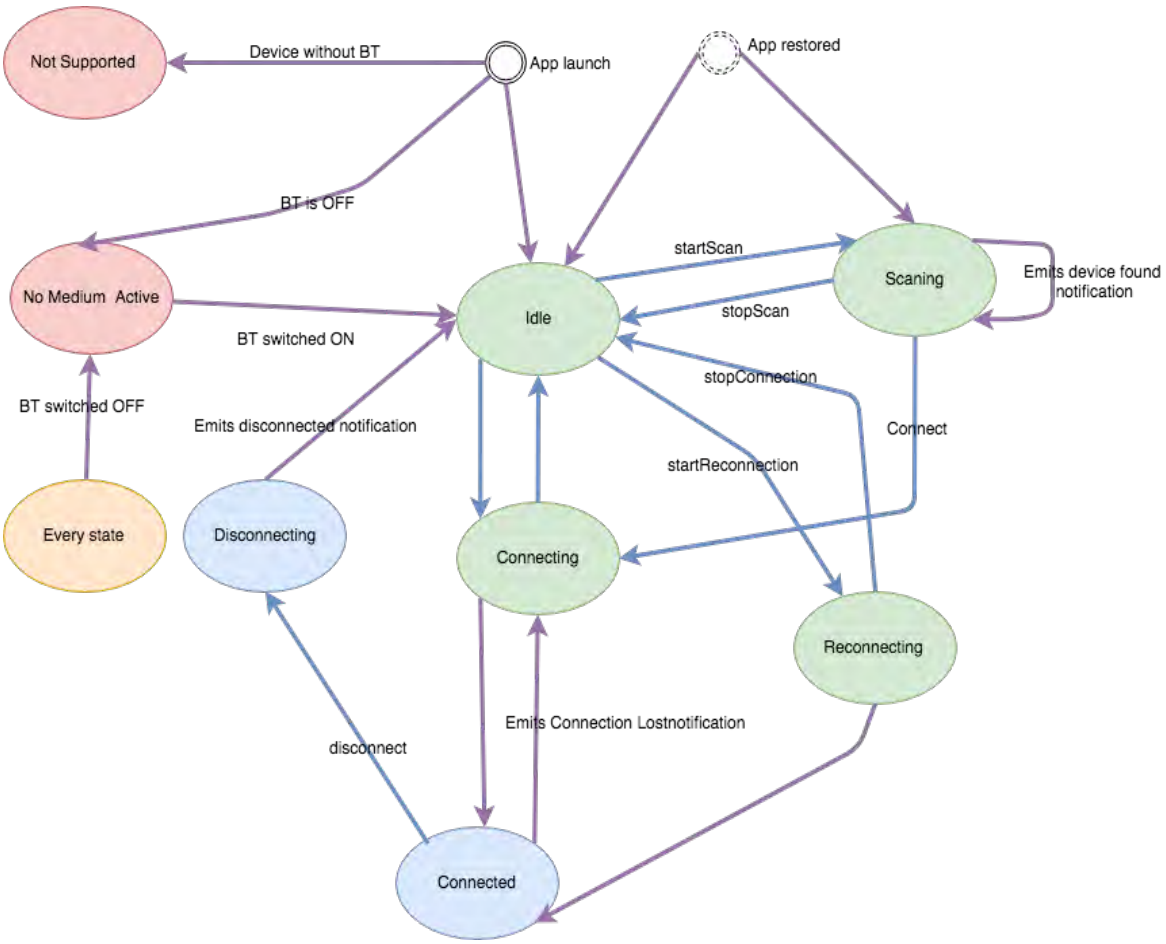

Abbildung 3.1: Zustandsdiagramm der Bluetooth Verbindung zu Dongle.

3.1.2 Dongle Verbindung

Abbildung 3.1 stellt das Zustandsdiagramm der Bluetooth-Verbindung zu Dongle[1] dar. Es zeigt die Verwaltung der Kommunikation mit dem Dongle. Die App muss alle dargestellten Zustände und Übergänge erkennen und gegebenenfalls den Benutzer zum nächsten Schritt leiten. Die Kommunikation mit dem Gerät soll automatisch stattfinden. Findet eine automatische Verbindung nicht statt, ist die manuelle Verbindung zum Dongle möglich.

[1] OBD2-Bluetooth Adapter

3.1.3 Fahrstil Bewertung

In der Tabelle 3.2 ist das Soll-Verhalten aufgelistet. Jede Abweichung davon wird mit schlechteren Punktzahlen oder gegebenenfalls negativ bewertet.

Title	Beschreibung
Motorstart 1	Den Motorstart erst kurz vor dem Fahrtbeginn durchführen.
Motorstart 2	Insbesondere im Kaltbetrieb hohe Motordrehzahl vermeiden.
Standphasen 1	Fahrzeugstandphasen möglichst vermeiden oder minimieren.
Standphasen 2	Motorleerlaufverbrauch durch Motorabstellen weitgehend vermeiden.
Beschleunigung 1	Hochschalten mit teilweisem Gangüberspringen durchführen.
Beschleunigung 2	Normale Beschleunigungen möglichst niedertourig im hohen Gang durchführen.
Verzögerung	Optimal wäre, die Entwicklung des Verkehrsablaufs zu beobachten und frühzeitig den eigenen Fahrtablauf danach auszurichten. Vor der Bremspedalbetätigung das Fahrzeug ohne eingelegten Getriebegang ausrollen lassen oder die Motorbremse einsetzen, dann sanft das Bremspedal betätigen.
Reisegeschwendigkeit 1	Gleichmäßiges Geschwindigkeitsprofil anstreben.
Reisegeschwindigkeit 2	Konstantgeschwindigkeiten immer im höchstmöglichen Gang fahren.

Tabelle 3.2: Fahrstil-Bewertung

3.1.4 Fahrtenbuch

Die App zeichnet die Daten jeder Fahrt auf. Eine Fahrt fängt an ab dem Zeitpunkt des Motoreinschaltens und endet mit dem Ausschalten. Diese Fahrten werden für den Benutzer aufgelistet.

4 Entwurf

Im Verlauf dieses Kapitels werden die wichtigsten Entwurfsmuster, die in dieser Arbeit eingesetzt werden, vorgestellt. Anschließend wird die gesamte App-Architektur erläutet.

4.1 Entwurfsmuster

Entwurfsmuster beschreiben generische Lösungen für Probleme, auf die andere Entwickler bereits gestoßen sind. Sie sind keine konkreten Implementierungen, sondern dienen als Ausgangspunkt für das Schreiben von Code.

Es gibt drei Haupttypen von Entwurfsmustern:

- Strukturelles Designmuster: Beschreibt, wie Objekte zu größeren Strukturen zusammengesetzt und kombiniert werden. Beispiele für strukturelle Entwurfsmuster sind Model-View-Controller (MVC), Model-View-ViewModel (MVVM) und Facade.
- Verhaltensmuster: Beschreibt, wie Objekte miteinander kommunizieren. Beispiele für Verhaltensmuster sind Delegation, Strategie und Beobachter.
- Gestaltungsmuster: Beschreibt, wie Objekte erstellt oder instanziiert werden. Beispiele für Gestaltungsmuster sind Builder, Singleton und Prototype.

4.1.1 Modell-View-Controller

Der Einsatz des Model-View-Controller-Musters ermöglicht die Trennung des Interface-Entwurfs vor seiner Implementierung. Das hat den Vorteil der einfachen Wiederverwendung von derselben Implementierung für eine neue Benutzerschnittstelle und erleichtert auch eine spätere Änderung oder Erweiterung.

MVC verwaltet die Kommunikation zwischen den Objekten aus den drei Bereichen: Datenmodell (Model), Programmsteuerung (Controller) und Präsentation (Ansicht). Das

Vorteile	Nachteile
Erstellen eine gemeinsame Sprache.	kann die Komplexität das Projekt erhöhen.
Vereinfachen den Einstieg von Entwicklern.	Viele Entwurfsmuster werden durch moderne Programmiersprachen überflüssig gemacht.
Mit Entwurfsmuster werden Ähnlichkeiten zwischen Codes schneller erkannt.	Objektorientierte Prinzipien werden nicht richtig gelernt.

Tabelle 4.1: Entwurfsmuster

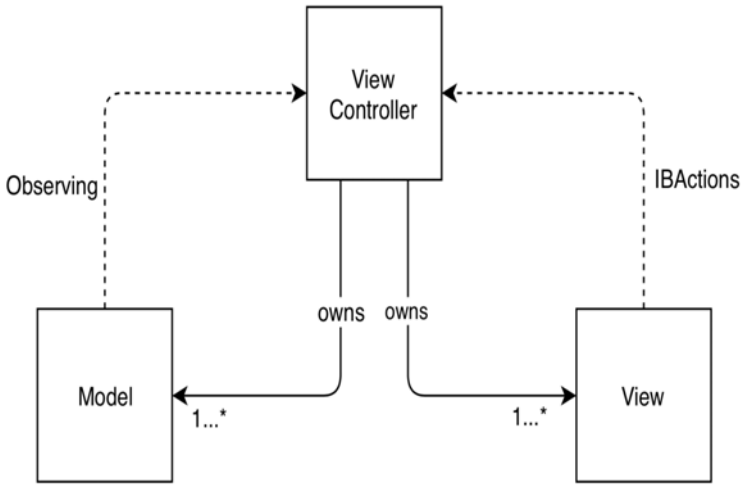

Abbildung 4.1: Modell-View-Controller.

Datenmodell enthält die Logik der Applikation, der Controller die User Interface Logik und der Bereich Ansicht die graphischen Elemente. Das Datenmodell und Ansicht dürfen nie direkt miteinander kommunizieren. Der Controller kann direkt auf Model zugreifen. Falls das Datenmodell Änderungen für den Controller hat, kann es die sogenannte „radio station“ anbieten, auf der der Controller sich registrieren kann. Der Controller kann direkt auf Ansicht zugreifen. Der Controller kann Ziele (Targets) in sich definieren, die von Ansicht durch Aktionen beeinflusst werden können, sodass der Controller die Änderungen an das User-Interface mitbekommt. Falls Ansicht sich mit dem Controller synchronisieren möchte, kann das Konzept von Delegieren benutzt werden. Der Controller setzt sich selbst als Delegaten von Ansicht durch die Implementierung der Protokolle. Ansicht besitzt die Daten nicht, kann sie aber vom Protokoll abfragen.

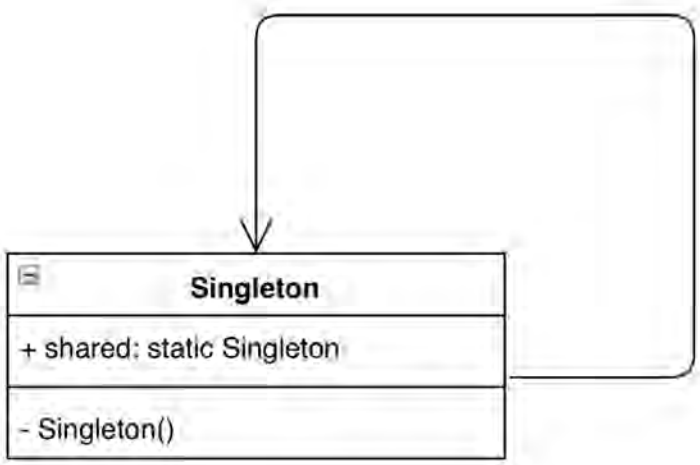

Abbildung 4.2: Singleton.

4.1.2 Singleton

Das Singleton-Muster beschränkt eine Klasse auf nur eine Instanz. Jeder Verweis auf die Klasse bezieht sich auf die gleiche zugrunde liegende Instanz. Dieses Muster ist sehr verbreitet in der Entwicklung von iOS-Apps, da Apple es in großem Umfang nutzt.

Das Singleton-Muster soll verwendet werden, wenn es problematisch ist, mehr als eine Instanz einer Klasse zu haben. Es ist zu beachten, dass das Singleton-Muster nicht in jeder Situation sinnvoll ist; es ist zum Beispiel nicht dafür geeignet, Informationen von einem View-Controller zu einem anderen zu übertragen. Damit entsteht schnell der Global Variablen-Effekt.

4.1.3 Delegation

Das Delegationsmuster ermöglicht es einem Objekt, ein anderes Objekt zu verwenden, um Daten bereitzustellen oder eine Aufgabe durchführen zu lassen. Dieses Muster besteht aus drei Teilen:

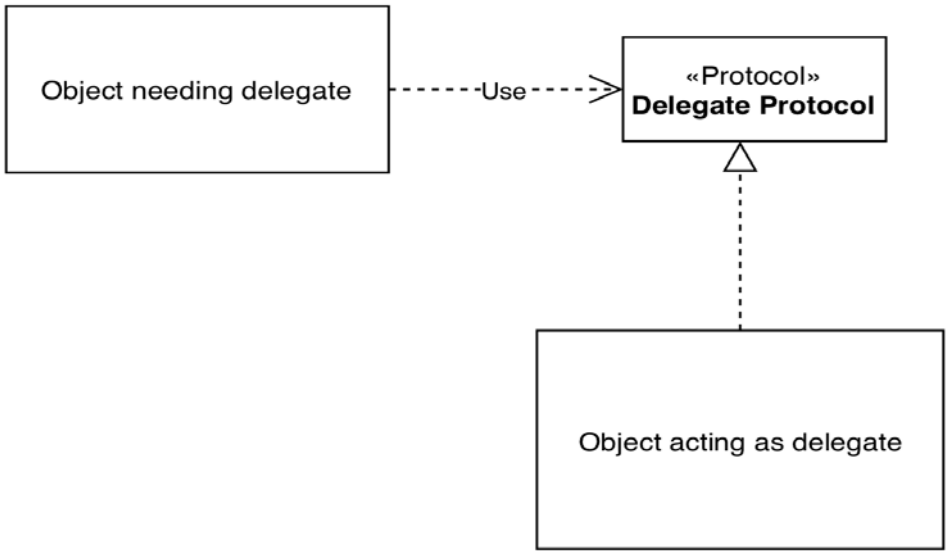

Abbildung 4.3: Delegation.

- Ein Objekt, das einen Delegaten benötigt.
- Ein Delegaten-Protokoll, das die Methoden definiert, die ein Delegat implementieren kann oder sollte.
- Ein Delegat, bei dem es sich um das Hilfsobjekt handelt, das das Delegaten-Protokoll implementiert.

Durch die Verwendung eines Delegaten-Protokolls anstelle eines konkreten Objekts ist die Implementierung flexibler: Jedes Objekt, das das Protokoll implementiert, kann als Delegat verwendet werden.

Dieses Muster soll verwendet werden, um große Klassen aufzuteilen oder generische, wiederverwendbare Komponenten zu erstellen. Delegierten-Beziehungen sind in allen Apple-Frameworks üblich, insbesondere in UIKit. Sowohl Objekte mit DataSource- als auch Delegaten-Namen folgen dem Delegations-Muster, da jedes ein Objekt betrifft, das von einem anderen aufgefordert wird, Daten bereitzustellen oder etwas zu tun. Delegaten sind nützlich, können aber auch zu häufig verwendet werden. Wenn ein Objekt mehrere Delegaten benötigt, kann dies ein Hinweis darauf sein, die Funktionalität des Objekts für bestimmte Anwendungsfälle aufzuteilen, anstatt eine einzige große Klasse zu bilden.

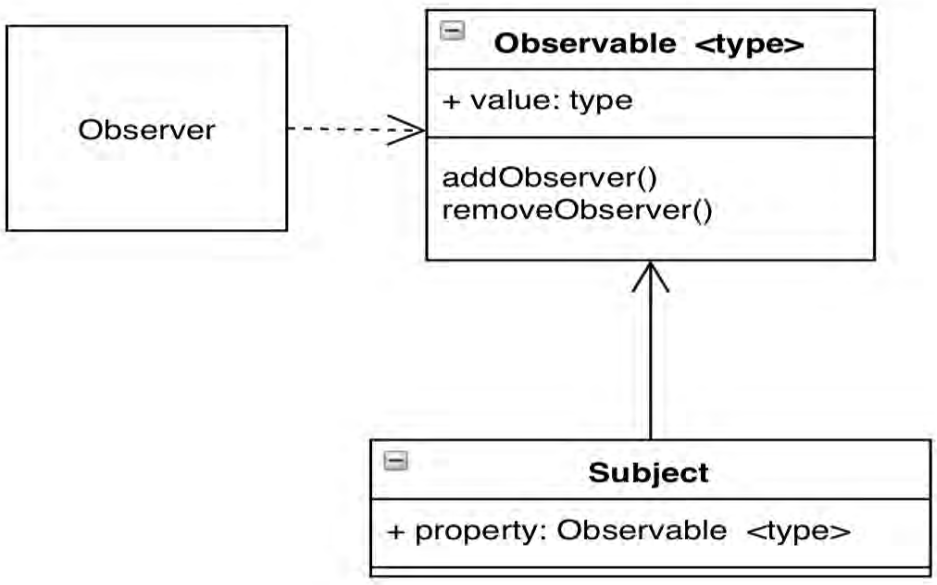

Abbildung 4.4: Observer.

4.1.4 Observer

Das Beobachtermuster lässt ein Objekt die Änderungen eines anderen Objekts beobachten. Zur Implementierung gibt es verschiedene Methoden, beispielsweise die Schlüsselwertbeobachtung (Key-Value-Observation, KVO) oder die Observable-Wrappers.

Dieses Muster hat zwei Hauptobjekte:

- Das Subjekt: ist das Objekt, das beobachtet wird.
- Das Beobachter-Objekt.

Das Beobachtermuster wird immer benutzt, wenn Änderungen an einem anderen Objekt erhalten werden sollen. Dieses Muster wird häufig bei MVC verwendet, wobei der View-Controller der Beobachter und das Modell das Subjekt ist. Dadurch kann das Modell seine Änderungen an den View-Controller übermitteln, ohne dass etwas über den Typ des View-Controllers bekannt ist. Somit können verschiedene View-Controller Änderungen an einem gemeinsam genutzten Modelltyp verwenden und beobachten.

4.2 App-Architektur

In diesem Abschnitt erfolgt ein vereinfachter Überblick über die Softwarearchitektur der App. Die wichtigsten Module sind: `DriverAwarness`, `Datamanager`, `Location`, `MapClient`, `FirebaseClient`, `DongleManager` und `DatabaseController`.

`DongleManager` ist verantwortlich für die Bluetooth-Kommunikation mit dem Dongle und dient als Einstiegspunkt für ankommende Daten vom Fahrzeug. `MapClient` verwaltet die Kommunikation mit die GoogleMap RestAPI , Mithilfe von HTTP-Requests nach dem RestAPI-Modell können Eigenschaften der Fahrstrecke wie die maximal zulässige Geschwindigkeit sowie auch die Auflösung von GPS-Positionen als Adressen oder Straßen eingeholt werden. Positionsdaten werden vom Modul `Location` übernommen, das den Zugriff auf das GPS-Modul von iOS verwaltet. Alle diese Daten werden mithilfe der `Datamanager` fusioniert und als eine Datenstruktur erstellt, sodass das Modul `DriverAwareness` diese für die Fahrstilbewertung nutzen kann. Zur Persistierung der Daten dient dabei eine SQLite-Datenbank, genannt CoreData, die durch das Modul `DatabaseController` verwendet wird. Firebase wird dabei verwendet, um die Benutzerdaten zu verifizieren. Dies erfolgt mithilfe des Firebase Frameworks für iOS. Die Ankopplung an die Services geschieht dabei über `RestAPIClient`. Er liefert nach erfolgreicher Authentifizierung die Zugriffsrechte.

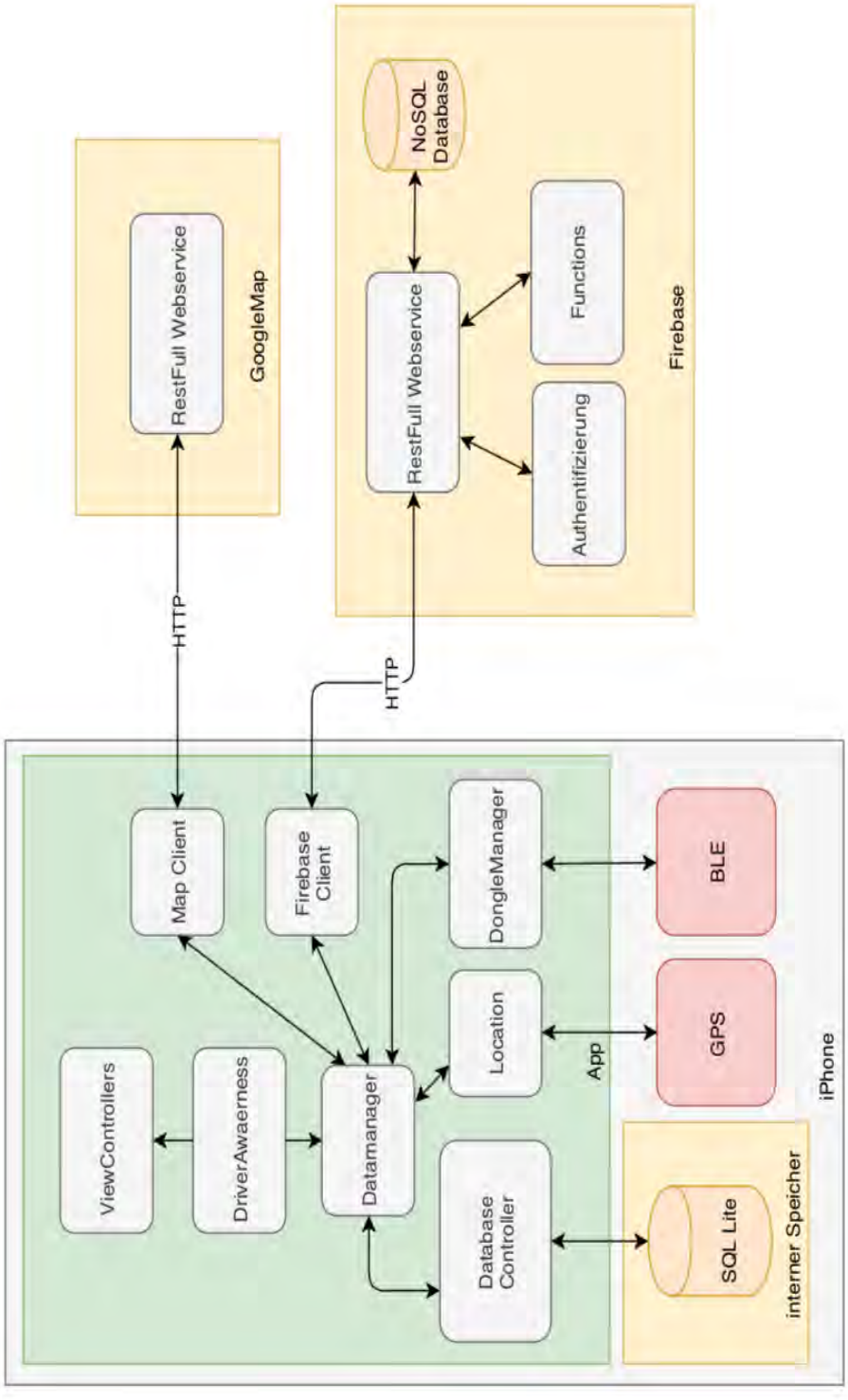

Abbildung 4.5: App-Architektur.

5 Realisierung

Im folgenden Abschnitt werden die Werkzeuge für die Entwicklung von iOS und Firebase Anwendungen erläutert.

5.1 iOS-Entwicklung

Um das iOS SDK und die Entwicklungsumgebung Xcode herunterzuladen, es ist zuerst einmal notwendig sich im Apple Developer Portal zu registrieren.

5.1.1 Xcode

Als Entwicklungsumgebung wird Xcode von Apple verwendet. Mit seinen Highlights wie dem neuen Editor, Interface Builder, Simulator und der Integration von Git bietet er das gesamte Spektrum für die iOS-Entwickler [16].

Einer der wichtigsten Punkte des Editors sind die Refactor- und Transfer-Features wie:

- Hinzufügen von fehlenden Protokollanforderungen
- Erzeugen von fehlenden Implementierungstubs
- Hinzufügen fehlender Überschreibungen für abstrakte Methoden
- Extrahieren von Variablen
- Extrahieren von Methoden
- Konvertieren von If-/else-Statement zu Switch-Statement
- Konvertieren von Switch-Statement zu If-/else-Statement

Interface Builder ermöglicht eine graphische Implementierung von Benutzerschnittstellen, ohne Codes schreiben zu müssen. Durch `Drag and Drop` können vordefinierte Objekte wie `Views`, `Buttons` und `Text Fields` zum Storyboard addiert werden. Danach

kann die Schnittstelle direkt mit dem Sourcecode verbunden werden, um die Aktionen zu behandeln oder Grafische Benutzeroberfläche zu konsumieren. Im Storyboard können dann die `Views` miteinander kombiniert werden, sodass der Benutzer durchnavigieren kann. Um die `Views` klar definiert zu halten, existiert ein Layout-System, das sogenannte Auto-Layout, das ermöglicht, die Einschränkungen von jedem einzelnen Objekt der Schnittstelle zu definieren, um zu bestimmen, wie die `Views` miteinander reagieren.

Der Simulator bietet eine gute Möglichkeit, die App schnell und effektiv zu testen. Die simulierte App interagiert mit der Geste der Benutzer und anderen Apps wie Maps analog zu echten Geräten. Des Weiteren können verschiedene Simulationen gleichzeitig durchgeführt werden.

Mit der Integration von Git und GitHub ist die Kontrolle und Versionisierung von Sourcecodes einfacher geworden. So unterstützt der Xcode Befehle wie `creating, commit, push, pull` und `merging`. Durch den Source Control Navigator ist ein schneller Zugriff auf `Branches, Tags` und `Server` möglich.

5.1.2 Swift

Swift ist eine objektorientierte Programmiersprache von Apple für iOS, macOS, tvOS, watchOS und Linux. Sie wird basierend auf Konzepten von modernen Programmiersprachen entwickelt, um eine sichere, schnelle und interaktive Programmiersprache anzubieten. Neben den Konzepten wie Klassen, Vererbung, Closures, Typinferenz, generischen Typen und Namensräumen sowie multiplen Rückgabetypen und -werten definiert Swift Klassen von bekannten Programmierfehlern, die schon beim Entwurf geprüft werden.

5.1.3 App Lebenszyklus

Der Eingangspunkt einer iOS-App ist die UIApplicationDelegate. Dieses ist ein Protokoll, das die App implementieren muss, um die verschiedenen Zustände wie ‚App-Start', ‚App im Hintergrund oder Vordergrund' oder ‚App wird beendet' zu erkennen Abbildung 5.2. Der Lebenszyklus der App ist wichtig, um die App und ihre Objekte ordnungsgemäß zu initialisieren. Die wesentlichen Methoden des Protokolls, die in der Regel von einer App implementiert werden, sind:

- `application(application: willFinishLaunchingWithOptions) -> Bool`: Diese Methode ist für den ersten Anwendungsaufbau vorgese-

hen. Storyboards wurden zu diesem Zeitpunkt bereits geladen.

- `application(application: didFinishLaunchingWithOptions) -> Bool`: wird als nächstes gerufen. Diese Callback-Methode wird aufgerufen, wenn der Start der Anwendung abgeschlossen ist oder wenn der Status wiederhergestellt wurde. Hier kann die Initialisierung abgeschlossen werden, z. B. das Erstellen der Benutzeroberfläche.
- `applicationWillResignActive(application: UIApplication)` wird aufgerufen, wenn die Anwendung vom aktiven in den inaktiven Status übergeht. Dies kann bei bestimmten Arten von vorübergehenden Unterbrechungen (z. B. eingehenden Telefonanrufen oder SMS-Nachrichten) auftreten oder wenn der Benutzer die Anwendung beendet und der Übergang in den Hintergrundstatus beginnt. Die Methode wird verwendet, um laufende Tasks anzuhalten und Timer zu deaktivieren.
- `applicationDidEnterBackground(application: UIApplication)` Die Methode wird verwendet, um Ressourcen freizugeben, Benutzerdaten zu speichern, Zeitgeber zu annullieren und genügend Informationen zum Anwendungsstatus zu speichern, um die Anwendung im aktuellen Status wiederherzustellen, falls sie später beendet wird.
- `applicationWillEnterForeground(application: UIApplication)` wird bei Übergang vom Hintergrund in den aktiven Zustand aufgerufen.
- `applicationDidBecomeActive(application: UIApplication)` Hier können alle Aufgaben neugestartet werden, die angehalten wurden, während die Anwendung inaktiv war. Wenn sich die Anwendung zuvor im Hintergrund befand, könnte die Benutzeroberfläche hier aktualisiert werden.
- `applicationWillTerminate(application: UIApplication)` wird aufgerufen, wenn die Anwendung kurz vor dem Abschluss steht. Sie wird für die Speicherung der Daten verwendet, bevor die Anwendung beendet wird.

5.1.4 Frameworks

Das Betriebssystem von Apple für mobile Geräte iOS ist auch eine Software-Plattform, die eine Schnittstelle bietet, um die Interaktion zwischen der Hardware und den Ap-

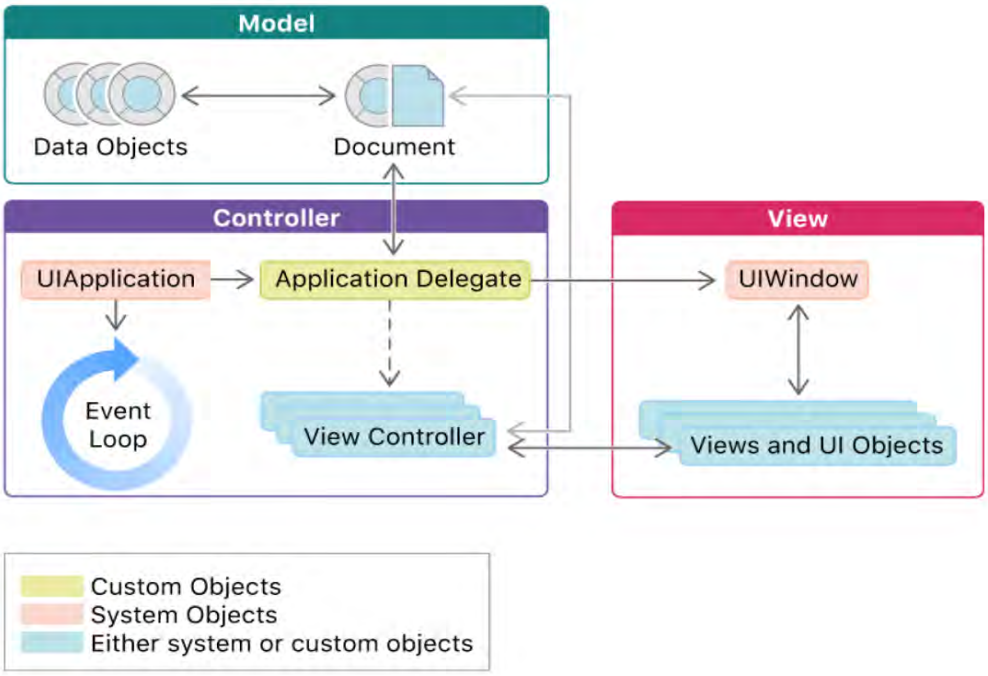

Abbildung 5.1: App-Lebenszyklus [17].

plikationen zu vereinfachen. So können native Apps mithilfe von iOS-Frameworks für unterschiedliche Hardware unkompliziert entwickelt, installiert und getestet werden.

Wie in der Abbildung 5.3 zu sehen ist, kann iOS als vier Layer veranschaulicht werden: Cocoa Touch, Media, Core Services und Core OS. Höhere Layer basieren auf den niedrigeren Layern und bieten anspruchsvollere Dienste. Apple empfiehlt, zuerst die obersten Layer zu nutzen und nur bei Bedarf auf die unteren Layer zuzugreifen.

Im Rahmen dieses Projekts werden die Frameworks Foundation, UIKit und CoreData verwendet. Foundation Framework ist eine Sammlung von Schnittstellen für Basis-Datentypen und Funkionalitäten. Das UIKit Framework unterstützt die Erstellung von graphischen Elementen sowie die Erkennung von verschiedenen Gesten (Touch). CoreData Framework fördert die Erstellung und die Verwaltung des Datenmodells. In diesem Projekt wurde es für die persistente Speicherung der Objekte in einer CoreData-Datenbank genutzt, die im Vergleich zu anderen Alternativen wie XML eine höhere Performance aufweist.

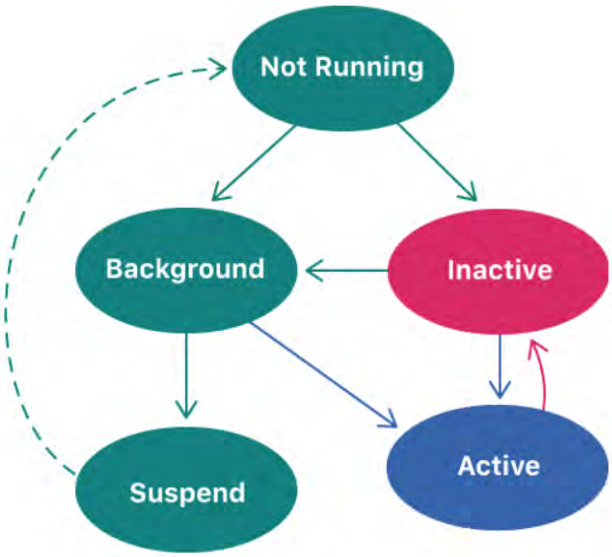

Abbildung 5.2: App-Zustände [17].

5.2 Firebase – Serverlose Lösung

Firebase is eine Plattform, die auf der Google-Infrastruktur basiert ist. Sie bietet Funktionen wie Authentifikation, Datenbanken, Analysen ,Messaging und Absturzberichte und wird automatisch skaliert. Damit wird bei der Entwicklung auf die Geschäftslogik konzentriert.

5.2.1 Authentifikation

Die Firebase-Authentifizierung bietet Backend-Services, Frameworks und vorgefertigte UI-Bibliotheken, um Benutzer für eine App zu authentifizieren. Sie unterstützt die Authentifizierung mit Kennwörtern, Telefonnummern und Anbietern wie Google, Facebook oder Twitter. Die Firebase-Authentifizierung lässt sich in andere Firebase-Services integrieren und nutzt Standards wie OAuth 2.0 und OpenID Connect, sodass ein eigenes benutzerdefiniertes Backend integriert werden kann. Es besteht die Möglichkeit, die Firebase Authentication SDK zu verwenden, um eine eigene Anmeldemethode manuell in die App zu integrieren.

- E-Mail- und passwortbasierte Authentifizierung
- Integrierte Identitätsanbieter-Integration: Google, Facebook, Twitter, GitHub

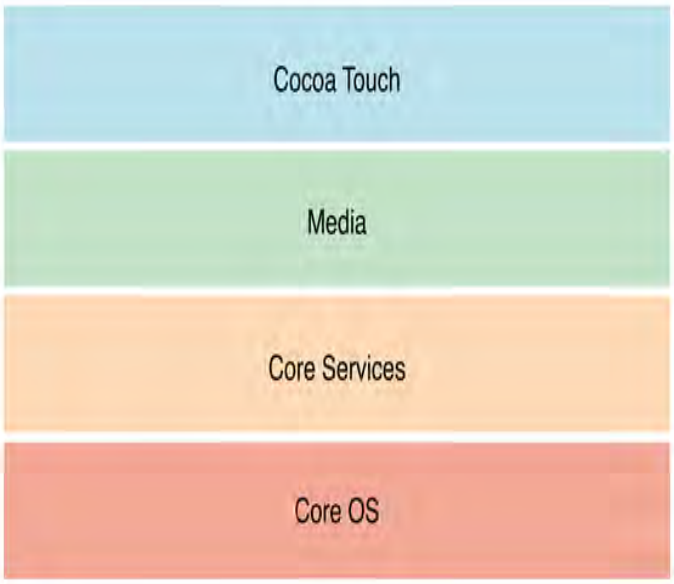

Abbildung 5.3: iOS-Schichten [16].

- Rufnummernauthentifizierung
- Benutzerdefinierte Authentifizierungssystemintegration
- Anonyme Authentifizierung

Damit sich ein Benutzer in der App anmelden kann, muss er Authentifizierungsinformationen eingeben. Diese Anmeldeinformationen können die E-Mail-Adresse und das Kennwort des Benutzers oder ein OAuth-Token sein. Anschließend werden die Anmeldeinformationen an das Firebase Authentication SDK übergeben. Die Backend-Services überprüfen die Anmeldeinformationen und senden eine Antwort an den Client zurück. Nach einer erfolgreichen Anmeldung werden grundlegende Profilinformationen des Benutzers zugänglich.

5.2.2 Cloud Firestore

Cloud Firestore ist eine flexible, skalierbare NoSQL-Datenbank für die Mobil-, Web- und Serverentwicklung von Firebase und Google Cloud Platform. Die Daten werden über Echtzeit-Listener der Client-Apps hinweg synchronisiert und bieten eine Offline-Unterstützung für mobile und Web-Lösungen, sodass responsive Apps unabhängig von der Netzwerklatenz oder Internetkonnektivität erstellt werden können

Die Daten werden in Dokumenten gespeichert, die Felder enthalten, denen die Werte zugeordnet werden. Diese Dokumente werden in Sammlungen gespeichert. Diese sind Container für die Dokumente, die zum Organisieren der Daten und zum Erstellen von Abfragen verwendet werden können. Dokumente unterstützen viele verschiedene Datentypen – von einfachen Zeichenfolgen und Zahlen bis hin zu komplexen, verschachtelten Objekten. Sie können auch Untersammlungen in Dokumenten enthalten und damit eine hierarchische Datenstruktur erstellen, die mit dem Wachstum ihrer Datenbank skaliert werden kann.

Mit Abfragen können die Daten auf Dokumentebene erscheinen, ohne die gesamte Samm-lung oder verschachtelte untergeordnete Sammlungen abrufen zu müssen. Sortierung, Filterung und Beschränkungen können hinzugefügt werden. Es sind Echtzeit-Listener vorhanden, um die Daten der App auf dem neuesten Stand zu halten, ohne bei jeder Aktualisierung die gesamte Datenbank abzurufen. Durch das Hinzufügen von Echtzeit-Listenern zu der App werden immer Benachrichtigungen mit einem Daten-Snapshot erstellt. Um den Zugriff auf die Daten im Firestore zu verwalten, eignen sich die Firestore-Sicherheitsregeln. Damit können die Zugriffsrechte flexibel angepasst werden.

5.2.3 Functions

Mit Cloud-Functions der Firebase wird der Backend-Code automatisch als Reaktion auf Ereignisse ausgeführt, die durch Firebase-Functions oder HTTPS-Anforderungen ausgelöst werden können. Der Code wird in der Google Cloud gespeichert und läuft in einer verwalteten Umgebung. So sind keine eigene Serververwaltung und Skalierung notwendig.

Nachdem eine Funktion geschrieben und bereitgestellt wurde, beginnen die Server von Google sofort mit der Verwaltung der Funktion. Diese kann direkt mit einer HTTP-Anforderung ausgelöst werden. Bei Hintergrundfunktionen überwachen die Server von Google Ereignisse und führen die Funktion durch, wenn sie ausgelöst werden. Wenn die Belastung zu- oder abnimmt, reagiert Google mit einer schnellen Skalierung der Anzahl der für die Ausführung der Funktion erforderlichen virtuellen Serverinstanzen. Jede Funktion wird isoliert in einer eigenen Umgebung mit einer eigenen Konfiguration ausgeführt.

- Es wird ein Code für eine neue Funktion geschrieben und ein Ereignisanbieter ausgewählt, z. B. eine Echtzeitdatenbank. Zudem werden die Bedingungen definiert, unter denen die Funktion ausgeführt werden soll.

- Nach der Bereitstellung der Funktion verbindet Firebase diese mit dem ausgewählten Ereignisanbieter.
- Wenn der Ereignisanbieter ein Ereignis generiert, das den Bedingungen der Funktion entspricht, wird der Code aufgerufen.
- Wenn die Funktion viele Ereignisse verarbeitet, erstellt Google mehr Instanzen, um die Arbeit schneller zu erledigen. Wenn die Funktion inaktiv ist, werden Instanzen bereinigt.
- Wenn der Entwickler die Funktion durch Bereitstellen des Codes aktualisiert, werden alle Instanzen für die alte Version bereinigt und durch neue Instanzen ersetzt.
- Wenn ein Entwickler die Funktion löscht, werden alle Instanzen bereinigt und die Verbindung zwischen der Funktion und dem Ereignisanbieter wird entfernt.

5.3 Implementierung

Die App ist nach dem Model-View-Controller-Pattern aufgebaut. Im Folgenden wird auf die wichtigen Klassen eingegangen, indem die Grafische Benutzeroberfläche aufgezeigt wird und die angebundene Implementierung beschrieben wird.

5.3.1 Authentifizierung

Die Klassen `SignUp` und `SignUPViewController` sind für den Registrierungsprozess zuständig. Abbildung 5.5 zeigt die Felder, die der Benutzer eingeben muss, um sich in der App anzumelden. E-Mail und Passwort werden nach regulären Ausdrücke validiert. Im Fall eine nicht erfolgreiche Validierung, wird die Methode ein Alert auflösen, damit der Benutzer seine Eingaben korrigieren kann. Ist die Eingabe erfolgreich, nutzt das Modell `SignUp` die Firebase-Methoden `createUser` und `sendEmailVerification`, um einen neuen Benutzer zu erstellen und anschließend eine Bestätigungsmail zu versenden. Der Benutzer muss auf den gesendeten Link klicken, um die Registrierung zu abzuschließen.

```
class SignInViewController {
  Auth.auth().signIn(withEmail:, password:) {
    (user , error) in
        if error != nil {
                ...
                self.present(alert, animated: true)
                return
        }
        if let user = user {
                let uid = user.user.uid
                let email = user.user.email
                if user.user.isEmailVerified {
                self.performSegue(withIdentifier:, sender:)
        }
  }
}
```

Listing 1: Die Klasse SignInViewController verwendet die Firebase Authentication SDK, um eine eigene Anmeldemethode in die App zu integrieren.

Die Klassen `SignIn` und `SignInViewController` sind für den Anmeldeprozess zuständig. 5.4. wird als Erstes angezeigt nach dem Appstart. Hier kann der Benutzer sich anmelden. Nach der Eingabe der E-Mail und des Passworts wird mithilfe der Firebase-Methode `Auth.auth().signIn(withEmail:,password:)` die Identität der Benutzer überprüft. Sind die Eingaben korrekt und hat der Benutzer vorher die Registrierung bestätigt, startet die App mit der Hauptansicht in Abbildung 5.10. Die Anmeldung bleibt auch nach der Beendigung der App vermerkt, sodass sich der Benutzer nicht immer wieder erneut anmelden muss. Die Anmeldung geht nur verloren, wenn der Benutzer sich abmeldet.

```
class SignUPViewController {
 Auth.auth().createUser(withEmail: em, password: pw1) {
    (user, error) in
    if error != nil {
        return
        }
    Auth.auth().currentUser?.sendEmailVerification {
        (error) in
        if error != nil {
                return
        }
        let alert = self.creatAlert(title:, message:, preferredStyle:)
                self.present(alert, animated: true,completion: nil)
        }
        navigationController!.popToRootViewController(animated: true)
 }
}
```

Listing 2: Die Klasse SignUPViewController verwendet die Firebase Authentication SDK für die Registrierung.

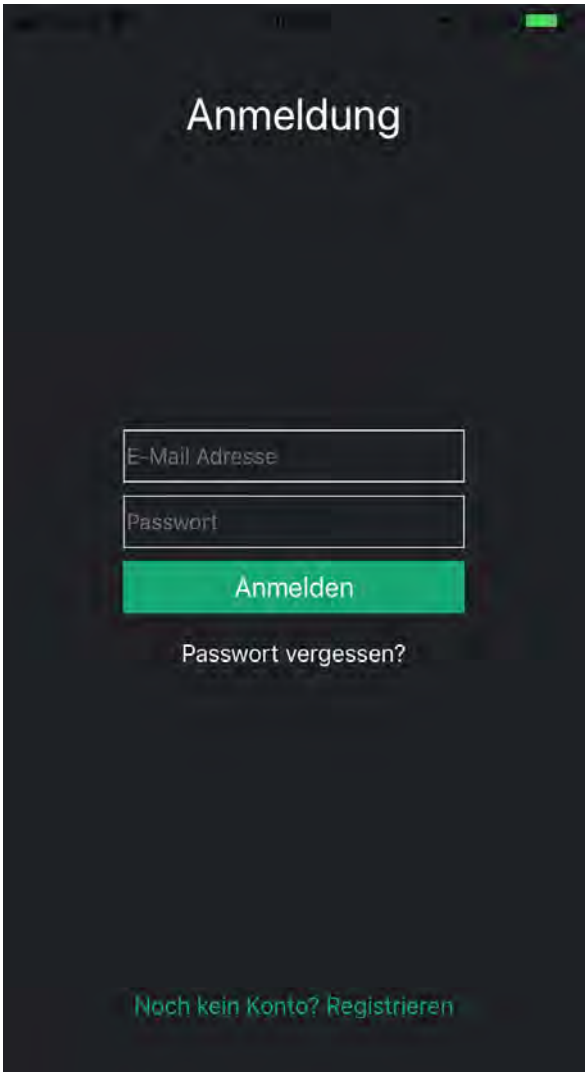

Abbildung 5.4: Login - Die App unterstützt nur die E-Mail- und passwortbasierte Authentifizierung.

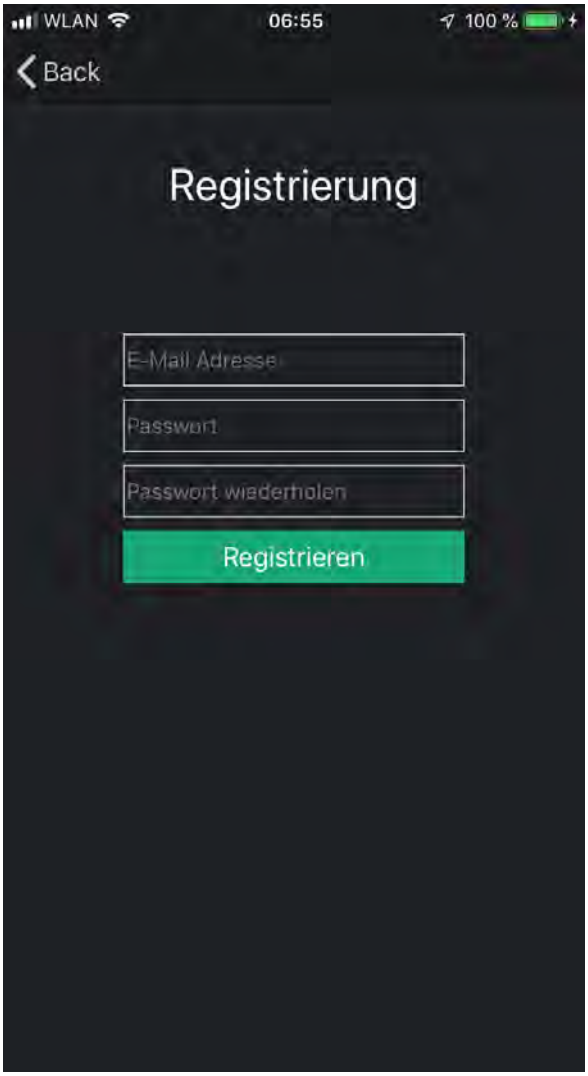

Abbildung 5.5: Email-Adresse und Passwort sind für die Registrierung notwendig.

Die Klassen `ResetPassword` und `ResetPasswordViewController` sind für die Passwort-Zurücksetzung zuständig (Abbildung 5.6). Ein Link wird durch die Firebase-Methode `Auth.auth().sendPasswordReset()` an eine eingegebene E-Mail-Adresse gesendet.

Abbildung 5.6: Passwort zurücksetzen.

5.3.2 Dongle Verbindung

Die Klassen `SyaraDongleConnectionViewController` und `SyaraDongleManager` sind für die Bluetooth-Kommunikation mit dem Dongle zuständig Abbildung 3.1. Dieser gibt dem Benutzer die Möglichkeit, nach dem Dongle zu suchen, falls es in einer Reichweite

liegt. Wird ein Dongle gefunden, aktiviert sich der Button für die Verbindung automatisch. `SyaraDongleConnectionViewController` muss sich als Observer registrieren bei dem Modell `SyaraDongleManager`. Dieses ist als Singletone realisiert. Mit den Methoden `profile()` und `communication()` überwacht die Verbindung Zustände und benachrichtigt die Observer bei Änderungen sofort.

```
var observers = [NSKeyValueObservation]()

func setupObserverDongleProfileStatus() -> NSKeyValueObservation {
        let observer = dongleManager.observe(\.dongleProfileStatus,
                                             options: [.initial]) {
        (dongleManager, _) in
        switch self.dongleManager.dongleProfileStatus {
                case "beginUpdateValues":
                self.dongleStatus = "dongleDataAvailable"
                self.dongleIndicator.stopAnimating()
                self.dongleIndicator.isHidden = true
                if let dongle = self.dongleManager.currentDevice?.name {
                        self.device.text = "Verbunden: " + dongle
                }
        default:
                break
        }
 }
return observer
}

func setupObserverDongleCommunicationStatus() -> NSKeyValueObservation {
        let observer = dongleManager.observe(\.dongleCommunicationStatus,
                                             options: [.initial]) {
           (dongleManager, _) in
                switch self.dongleManager.dongleCommunicationStatus {
                case "undefined", "notSupported", "noMediumActive":
                    self.dongleStatus = "notAvailable"
                case "idle":
                        self.dongleStatus = "idle"
                        self.scanButton.title = "Start scan"
                case "deviceFound":
                        self.dongleStatus = "deviceFound"
                ...
                default:
                        break
                }
        }
 return observer
}
```

Listing 3: Beobachter registrieren sich, um bei Änderung des Verbindung-Status benachrichtigt zu werden.

```
func profile(_ notification: Notification) {
        switch (notification as NSNotification).profileEventType {
        case .undefined:
                dongleProfileStatus = "undefined"
        case .incompleteDeviceInterface:
                dongleProfileStatus = "incompleteDeviceInterface"
        case .pairingError:
            dongleProfileStatus = "pairingError"
        case .authenticationError:
                dongleProfileStatus = "authenticationError"
        case .ready:
                dongleProfileStatus = "ready"
                if let updateValue =vehicle?.beginUpdateValues() {
                if updateValue {
                        dongleProfileStatus = "beginUpdateValues"
                    }
                }
        }
}

func profile(_ notification: Notification) {
        switch (notification as NSNotification).communicationEventType {
        case TXCommunicationEventType.noMediumActive:
             dongleCommunicationStatus = "noMediumActive"
        case TXCommunicationEventType.idle:
                 dongleCommunicationStatus = "idle"
        case TXCommunicationEventType.scanning:
                 dongleCommunicationStatus = "scaning"
        case TXCommunicationEventType.connecting:
                dongleCommunicationStatus = "connecting"
        case TXCommunicationEventType.deviceFound:
                dongleCommunicationStatus = "deviceFound"
        case TXCommunicationEventType.reconnecting:
                dongleCommunicationStatus = "reconnecting"
        case TXCommunicationEventType.connected:
                dongleCommunicationStatus = "connected"
        ...
        }
}
```

Listing 4: Verbindung-Status Handlung.

5.3.3 Benutzerprofil

Die Klassen `SyaraProfilViewController` und `SyaraUser` sind für die Benutzerdaten-Verwaltung zuständig Abbildung 5.7. Der Benutzer kann seinen Vornamen, Nachnamen sein Geburtsdatum, seine E-Mail-Adresse und Telefonnummer sowie ein Bild eingeben, ändern oder löschen. Das Modell `SyaraUser` nutzt die Klasse `DatabaseController` für die persistente Speicherung sowie die Firebase-Klasse `Auth`, um die Daten zu synchronisieren.

Die Klasse `SyaraProfilViewController` implementiert das Delegat `UIImagePicker`, um die Methoden `imagePickerController` und `imagePickerControllerDidCancel` zu verwenden. Dies ermöglicht den Zugriff auf die Kamera und die Fotos.

```
class SyaraProfilViewController: UITableViewController,
                                 UITextFieldDelegate,
                                 UIImagePickerControllerDelegate,
                                 UINavigationControllerDelegate {

    func imageButton(_ sender: UIButton) {
        let imagePickerController = UIImagePickerController()
        imagePickerController.delegate = self
        let actionSheet = UIAlertController(title: "Photo source",
                                            message: "Choose a source",
                                            preferredStyle: .actionSheet)
        actionSheet.addAction(UIAlertAction(title: "Camera",
                                            style: .default,
                                            handler: {
             (_: UIAlertAction) in
             imagePickerController.sourceType = .camera
             self.present(imagePickerController,
                          animated: true,
                          completion: nil)
        }))
        ...
        actionSheet.addAction(UIAlertAction(title: "Cancel",
                                            style: .cancel,
                                            handler: nil))
        self.present(actionSheet, animated: true, completion: nil)
    }

   func imagePickerController(picker: UIImagePickerController,
                              info: [.InfoKey : Any]) {
     let image = info[.originalImage] as? UIImage
     pictureImageView.image = image
     picker.dismiss(animated: true, completion: nil)
   }

   func imagePickerControllerDidCancel(picker: UIImagePickerController) {
     picker.dismiss(animated: true, completion: nil)
   }
}
```

Listing 5: Benutzerprofil.

Abbildung 5.7: Benutzerprofil.

5.3.4 Fahrzeugprofil

Die Klassen `SyaraCarViewController` und `SyaraCar` sind für die Fahrzeugdaten Verwaltung zuständig Abbildung 5.8. Hier werden dieselben Programmierungstechniken eingesetzt wie beim Benutzerprofil; lediglich die Eingabefelder unterscheiden sich. Die gefragten Daten sind: die Marke, das Modell und die Erstzulassung. Um ein genauere Bewertung zu geben, wird der Benutzer gefragt, auch die Daten über die Gang Übersetzungsverhältnisse, den durchschnittlichen Verbrauch sowie den Kraftstofftyp einzugeben.

Abbildung 5.8: Fahrzeugprofil.

5.3.5 Fahrtenbuch

Die Ansicht des Fahrtenbuchs besteht aus zwei Unteransichten: Karten und Tabelle 5.9. Beide werden von der Klasse `SyaraTripsViewController` verwaltet. Für die Kartenansicht wird das GoogleMap-Framework für iOS verwendet. Die Karte ist mithilfe der Methode `GMSMapStyle(contentsOfFileURL: styleURL)` und einer JSON-Datei konfigurierbar, die mit der Aufforderung `if let styleURL = Bundle.main.url(forResource: ßtyle", withExtension: "json")` geladen wird. Für die Positionsbestimmung benutzt die Klasse `SyaraTripManager` das Modul Location. Jede erkannte Fahrt von der Klasse `SyaraTripManager` wird an die Klasse `SyaraTripsViewController` übermittelt, sodass die Fahrt in der Tabelle `tableView` hinzugefügt wird. Um die im iOS vorgefertigte Implementierung für TabeView zu nutzen, muss `SyaraTripsViewController` die Delegaten `UITableViewDataSource` und `UITableViewDelegate` implementieren (siehe Listing 6).

```
class TripsViewController: UIViewController,
                           UITableViewDataSource,
                           UITableViewDelegate {

    var location = SyaraLocation.sharedInstance
        @IBOutlet weak var mapView: UIView!
        @IBOutlet weak var tableView: UITableView!

    func setupMapView() {
        let mapFrame = CGRect(x, y, width, height)
        self.gmsMapView = GMSMapView.map(mapFrame, camera)
        let tableFrame = CGRect(x, y, width, height)
        tableView.frame = tableFrame
        do {
            if let styleURL = Bundle.main.url(forResource: "style",
                                              withExtension: "json") {
                mapStyle = try GMSMapStyle(contentsOfFileURL: styleURL)
            }
        } catch {
            Logger.error("\(error)")
        }
        self.view.addSubview(gmsMapView)
    }

   func tableView(tableView: UITableView,
                  cellForRowAt indexPath: IndexPath) -> UITableViewCell {
     let cell = tableView.dequeueReusableCell(withIdentifier: "tripCell")
     as! TripTableViewCell
     let trip = tripManager.trips[indexPath.row]
     cell.date.text = trip.date
     cell.startTimestamp.text = trip.startTimestamp
     cell.endTimestamp.text = trip.endTimestamp
     cell.startAddress.text = trip.startAddress
     cell.endAddress.text = trip.endAddress
     return cell
   }
}
```

Listing 6: Fahrten.

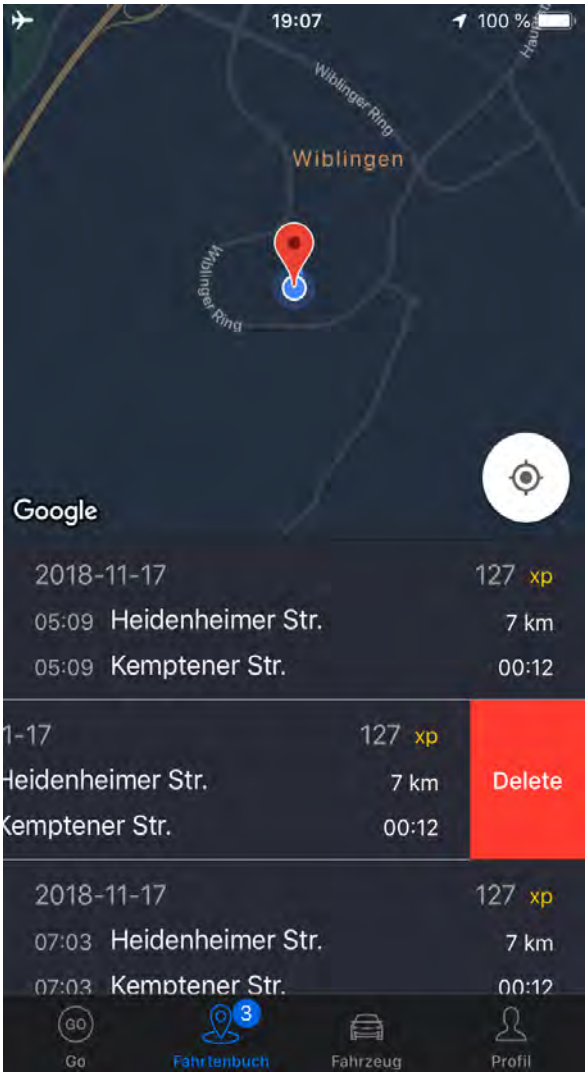

Abbildung 5.9: Fahrtenbuch.

5.3.6 Fahrstil

Die Klassen `SyaraDriverAwareness` und `SyaraLiveViewController` sind für die Fahrstil-Bewertung und ihre Visualisierung zuständig (siehe Abbildung 5.10). `SyaraDriverAwareness` übernimmt alle erfassten und synchronisierten Daten von `SyaraDataManager`, um das Fahrverhalten zu überwachen und anschließend zu bewerten. Es sind zwei Arten von Indikatoren zu unterscheiden: In-Echtzeit-Indikatoren (Geschwindigkeit und Verbrauch) und akkumulierte Indikatoren (Motorleistung, Beschleunigung, Verzögerung und vorausschauendes Fahren).

In Abbildung 5.11 ist die Bewertung der aktuellen Geschwindigkeit mit zwei Kreisen dargestellt: ein äußerer Kreis, der die maximal zulässige Geschwindigkeit repräsentiert sowie ein innerer Kreis, der die aktuelle Geschwindigkeit darstellt und seine Größe und Farbe dynamisch ändert. Je schneller man fährt und je näher man an die maximale Geschwindigkeit kommt, desto größer wird der Kreis und ändert seine Farbe von Grün über Orange zu Rot.

Die Animation der Verbrauchsbewertung ist analog zur Geschwindigkeitsbewertung. Der Soll-Wert ist in diesem Fall der Verbrauch, der auf der Eingabe im Fahrzeugprofil und auf der aktuellen Position (in einer Stadt, auf der Landstrasse oder Autobahn) basiert und daraus abgeleitet wird. Akkumulierte Indikatoren sind mit goldenen Kästen dargestellt. Die Bewertung wird vom Fahrtbeginn bis zum aktuellen Zeitpunkt immer aktualisiert, solange eine neue Fahrsituation erkannt wird. Alle akkumulierten Indikatoren sind prozentuell, wobei jeder Kasten einen Wert von 33 % darstellt. Der Leistungsindikator ist nicht anders als der Verbrauchsindikator über die Zeit. Sekündlich wird der Prozentwert aus dem Verhältnis Soll-/Ist-Verbrauch berechnet. Wird der Soll-Verbrauch überschritten, werden die Punkte abhängig vom Differenzwert und der Dauer der Überschreitung von 100 Punkten abgezogen. Die Klassen erkennen jede Geschwindigkeitsänderung. Jeder bewertete Beschleunigungsvorgang wird zur Beschleunigungshistorie addiert und daraus wird der Durchschnitt berechnet. Dies ist als aktuelle Bewertung durch die drei goldenen Kästen dargestellt. Analog zur Beschleunigung ist die Verzögerungsbewertung.

```
class SyaraDriverAwarenessViewController {
    var dataManager = SyaraDataManager.sharedInstance
    var syaraDriverAwareness = SyaraDriverAwareness()
    let speedLayer = CAShapeLayer()
    let maxSpeedLayer = CAShapeLayer()
    var speedRadius = 0
    var maxSpeedRadius = 0
    var maxSpeed = 60
    func evaluationSpeedAnimation(speed: Int, speedEvaluation: Double){
        let radius = map(speed: speed)
        let center = speedEvaluationView.center
        var speedRadius = CGFloat(radius) - CGFloat(map(speed: 0))
        if speedRadius > CGFloat(maxSpeedRadius) {
            speedRadius = CGFloat(maxSpeedRadius)
        }
        let circularPath = UIBezierPath(arcCenter: center,
                                        radius: speedRadius,
                                        startAngle: -CGFloat.pi / 2,
                                        endAngle: 2 * CGFloat.pi,
                                        clockwise: true)
        speedLayer.path = circularPath.cgPath
        switch speedEvaluation {
            case 0...1: speedLayer.fillColor = UIColor.green
            case 1.01...1.10: speedLayer.fillColor = UIColor.orange
            case 1.11...: speedLayer.fillColor = UIColor.red
            default: break
        }
    }
}
```

Listing 7: Geschwindigkeit-Animation.

```
class SyaraDriverAwareness {
    func accEvaluation(snapData: SnapData) -> Int {
        var total = 0.0

        for throttle in snapData.throttle {
           total = total + throttle.value
        }

        let average = Int(total) / snapData.throttle.count
        var throttleEvaluation = 0

        switch average {
            case 0...30: throttleEvaluation = 30
            case 31...60: valuation = 60
            case 61...: throttleEvaluation = 90
            default:break
        }
        accEvaluationHistory.append(throttleEvaluation)
        var sumThrottleEvaluation = 0
        for value in accEvaluationHistory {
            sumThrottleEvaluation = sumThrottleEvaluation + value
        }
        return sumThrottleEvaluation / accEvaluationHistory.count
    }

    func updateSpeedChange() {
      if let vehicleSpeed = SyaraDataManager.filtredVehicleSpeed,
         let fuelRate = SyaraDataManager.filtredFuelRate,
         let throttle = SyaraDataManager.filtredThrottlePosition,
         let rpm = SyaraDataManager.filtredEngineSpeed {
         let accEvaluation = syaraDriverAwareness.accTracking(::::)
         switch accEvaluation {
            case 1...10: acceleration(stars: 0)
            case 10...33: acceleration(stars: 1)
            case 34...66: acceleration(stars: 2)
            case 67...100: acceleration(stars: 3)
            default: break
         }
      }
    }
}
```

Listing 8: Beschleunigung-Bewertung.

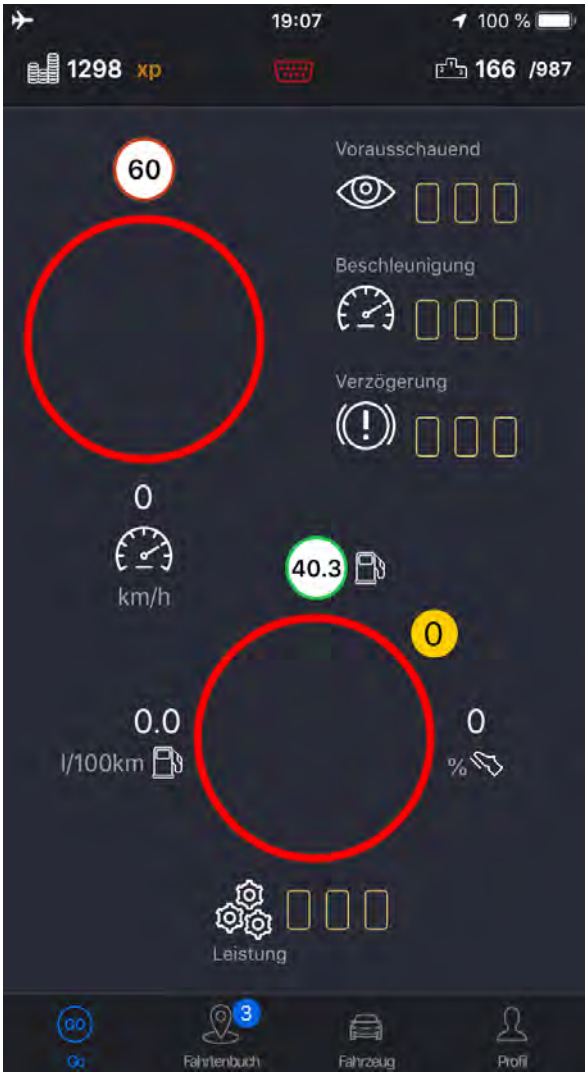

Abbildung 5.10: Feedback zum Fahrstil.

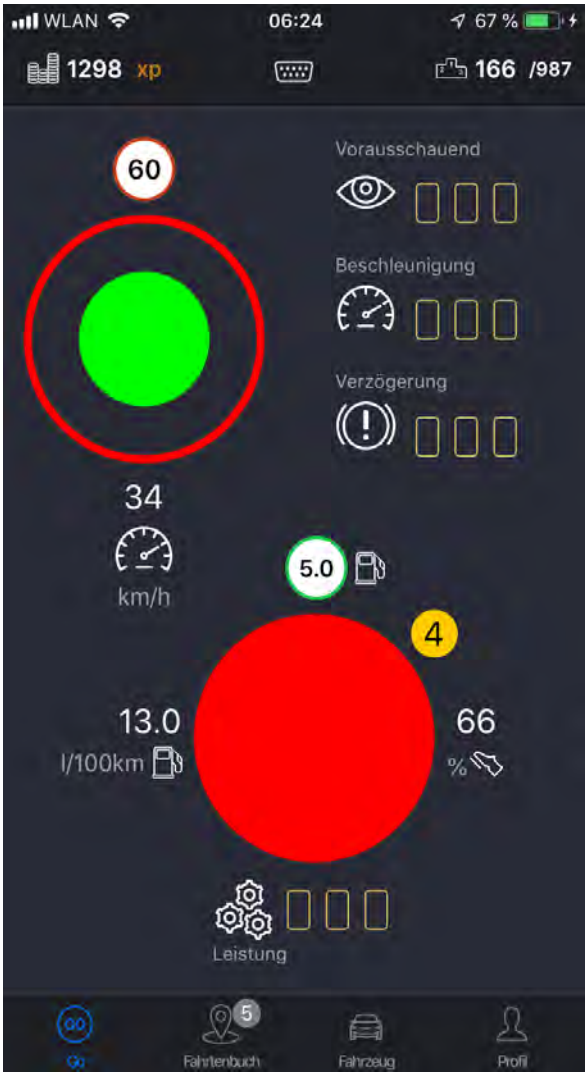

Abbildung 5.11: Fahrstil-Animation.

5.4 Simulation und Test

Die App wurde während der Implementierungsphase mithilfe eines Prüfstands getestet, der eine OBD-Schnittstelle hat. Ein Trip wurde während einer richtigen Fahrt als HEX-Datei aufgenommen und auf dem Prüfstand in einer Endlos-Schleife aufgespielt. Dies reduzierte die Kosten und den Zeitaufwand beim Testen.

Abbildung 5.12: OBD2-Simulator [18].

6 Zusammenfassung

Die Aufgabe der Masterarbeit bestand darin, eine mobile Anwendung für das Fahrstil-Feedback zu entwickeln und dessen Verbesserung zu erreichen. Nach einer Analyse der Fahrdynamik und ähnlicher Feedback-Systeme wurden die Anforderungen definiert, anhand derer die Architekturkonzepte und Benutzeroberflächen entworfen wurden. Schließlich wurden die Entwürfe in einer Implementierungsphase umgesetzt. Im folgenden Abschnitt erfolgt ein Ausblick auf die möglichen Weiterentwicklungen und Verbesserungen, die ergänzend in der App implementiert werden könnten.

6.1 Ausblick

Während der Entwicklung der App stellten sich mehrere Möglichkeiten zur Verbesserung bzw. Erweiterung verschiedener Aspekte der App heraus. Mögliche Verbesserungen und Erweiterungen, die zukünftig bei der App vorgenommen werden könnten, sind:

6.1.1 Individuelles andauerndes Fahrtraining

Das Bewusstsein für eine korrekte Fahrweise nimmt nach einer bestimmten Zeit ab. Dies zeigt sich vor allem bei neuen Fahrern, die kurz nach einer Schulung ein gutes Verhalten erreichen. Es nimmt jedoch kontinuierlich ab, sodass sich die Fahrer nach einigen Monaten im selben Zustand wie vor dem Training befinden. Viele Unternehmen bieten deshalb Fahrtrainingskurse an, für die viel Zeit, Energie und Kosten aufgewendet werden. Die App könnte erweitert werden, um intelligenter zu sein: Anstatt nur in Echtzeit oder akkumuliert ein Feedback zu geben, könnte sie das Verhaltensmuster des Fahrers archivieren und daraus seine Schwäche erkennen und entsprechend darauf reagieren, um beispielsweise den Fahrer zu trainieren oder individuelle Empfehlungen zu geben. Neuere Methoden und Ansätze könnten hier angewendet werden. Ein System, das dem Fahrer online und andauernd Hinweise zum richtigen Verhalten gibt, beeinflusst das Bewusstsein positiv und nachhaltig. Dies reduziert den Bedarf an externem

Fahrtraining bzw. an Schulungskosten. Des Weiteren motiviert es zu einem andauernden, selbstständigen und praktischen Lernen. Dabei nimmt die Benut-zerfreundlichkeit eine Schlüsselrolle ein. Die sinnvolle Darstellung der Daten soll erreicht werden, ohne den Fahrer abzulenken. Der Fahrstilassistent ergänzt ein Training und motiviert den Fahrer, seine erlernte Theorie in die Praxis umzusetzen.

6.1.2 Autonomer Fahrstil

Der Computer als Autofahrer wird im Vergleich zum Menschen nicht müde, sondern bleibt immer konzentriert und ruhig. Er hat keine Sehschwächen, trinkt keinen Alkohol und hält sich immer an die Verkehrsregeln. Das bedeutet, dass es zukünftig weniger Unfälle geben wird, denn der Computer hat eine bessere Achtsamkeit und schnellere Reaktionszeit als ein Mensch. Diese aber ist technisch bedingt und gilt nicht für alle Fahrsituationen. Zurzeit ist der Computer bei hohem Tempo nicht in der Lage, so vorausschauend und effizient zu fahren wie der Mensch. Die Technik bremst später und ruppiger, weil sie erst reagieren kann, wenn sie ein Hindernis in Reichweite erkennt. In anderen Situationen wie im Stau bietet die Technik ein besseres Verhalten, um den Verkehr flüssig zu halten.

Die Bewertung des Fahrerverhaltens, ob Fahrzeuge statt Benzin Strom verbrauchen und ob anstelle des Fahrers ein Roboter das Steuer übernimmt, wird immer gefragt sein, unabhängig davon, wer das Fahrzeug steuert und welche Leistungsquelle besteht. Es geht darum, zu bewerten, ob die Energie intelligent genutzt wird und Unfälle vermieden werden können, sowohl für die individuelle Fahrsituation als auch für die gemeinsame Nutzung der Fahrstrecke.

6.1.3 Fahrerliga

Ein spezifisches soziales Netzwerk für die Fahrer, das es ihnen ermöglicht, eigene Erfahrungen untereinander auszutauschen, führt zu einer Verbesserung des Fahrverhaltens. Mit Einführung des Wettbewerbs wird die menschliche Natur des Konkurrenzdenkens aktiviert, was auch zur Nutzung solcher Systeme motiviert. Um eine objektive Vergleichbarkeit der einzelnen Fahrer und eine maximale Flexibilität zu erreichen, müssen folgende Punkte berücksichtigt werden:

- Individualisierung nach Fahrertypen
- individuell einstellbare Fahrzeugprofile

- Einführung von Wettbewerben auf verschiedenen Ebenen, z. B. in der Stadt, in ländlichen Regionen oder weltweit.

Literaturverzeichnis

[1] BUNDESAMT, Statistisches: *Verkehrsunfälle - Fachserie 8 Reihe 7 - 2016*. 12.2017

[2] BUNDESAM, Statistisches: *Bericht*. Version: 12.2017. https://www.destatis.de/DE/PresseService/Presse/Pressemitteilungen/2016/12/PD16_451_85.html

[3] KUMAR, Manu ; KIM, Taemie: *Dynamic Speedometer: Dashboard Redesign to Discourage Drivers from Speeding*. 2005

[4] FABIUS STEINBERGER, Ronald S. April Moeller M. April Moeller: *Journal of Safety Research. The Antecedents, Experience, and Coping Strategies of Driver Boredom in Young Adult Males*. 10.2016

[5] JAMSON, Samantha L. ; HIBBERD, Daryl L. ; JAMSON, A. H.: *Drivers' ability to learn eco-driving skills; effects on fuel efficient and safe driving behaviour*. 02.2015

[6] BEUSEN, B ; BROEKX, S. ; DENYS, T. ; BECKX, C. ; DEGRAEUWE, B. ; GIJSBERS, M. ; SCHEEPERS, K. ; GOVAERTS, L. ; TORFS, R. ; PANIS, L.: *Using on-board logging devices to study the longer-term impact of an eco-driving course*. 2009

[7] VOORT van d.: *Design and evaluation of a new fuel efficiency support tool. Ph.D. dissertation, University of Twente*. 2001

[8] BARTH, M. ; BORIBOONSOMSIN, K.: *Energy and emissions impacts of a freeway-based dynamic eco-driving system*. 2009

[9] SCHICKLER, Marc ; REICHERT, Manfred ; PRYSS, Rüdiger ; SCHOBEL, Johannes ; SCHLEE, Winfried ; LANGGUTH: *Entwicklung mobiler Apps: Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health*. 2016

[10] Version: 11.2018. https://www.ecodrive.ch/de/

[11] *OBD-2 Allgemeines, technische Informationen*. Version: 11.2018. https://www.obd-2.de/obd-2-allgemeine-infos.html

[12] *Programmierer Tips*. Version: 11.2018. https://www.obd-2.de/programmierer-tips.html

[13] STEINBERGER., Fabius ; PROPPE., Patrick ; SCHROETER., Ronald ; ALT., Florian: *CoastMaster: An Ambient Speedometer to Gamify Safe Driving*. 10.2016

[14] MESCHTSCHERJAKOV, Alexander ; WILFINGER, David ; SCHERNDL, Thomas ; TSCHELIGI, Manfred: *Acceptance of Future Persuasive In-Car Interfaces Towards a More Economic Driving Behaviour*. 09.2009

[15] Version: 11.2018. https://media.daimler.com/marsMediaSite/de/instance/ko/EcoScore-car2go-laesst-Baeume-wachsen.xhtml?oid=9917754

[16] APPLE: *Apple Developer Documentation*. Version: 11.2018. https://developer.apple.com/documentation/

[17] APPLE: *The App Life Cycle*. Version: 11.2018. https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

[18] Version: 11.2018. https://www.diamex.de/dxshop/Diamex-OBD2-Profi-Simulator-alle-Protokolle

[19] BUNDESAM, Statistisches: *Bericht*. Version: 12.2017. https://www.destatis.de/DE/ZahlenFakten/GesamtwirtschaftUmwelt/Umwelt/UmweltoekonomischeGesamtrechnungen/MaterialEnergiefluesse/Tabellen/FahrleistungenHaushalte.html

[20] TAYLOR, M. ; LYNAM, D. ; BARUYA, A.: *The effects of drivers speed on the frequency of road accidents. Transport Research Laboratory TRL Report 421, Crowthorne*. 2000

[21] SILCOCK, D. ; SMITH, Knox K. ; D. ; BEURET, K.: *What limits speed? Factors that affect how fast we drive. AA Foundation for Road Safety Research, Basingstoke. Interim report*. 06.1999

[22] REPORT, Interim: *AA Foundation for Road Safety Research, Basingstoke*. 06.1999

[23] APPLE: *Swift Guide*. Version: 11.2018. https://docs.swift.org/swift-book/LanguageGuide/TheBasics.html

[24] APPLE: *App Programming Guide for iOS*. Version: 11.2018. https://developer.apple.com/library/archive/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Introduction/Introduction.html

[25] APPLE: *Auto Layout Guide*. Version: 11.2018. https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html#//apple_ref/doc/uid/TP40010853-CH7-SW1

[26] APPLE: *Core Data Programming Guide*. Version: 11.2018. https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CoreData/index.html

[27] DAHLINGER., Andre ; WORTMANN., Felix ; TIEFENBECK., Verena ; RYDER., Ben ; GAHR, Bernahrd: *Feldexperiment zur Wirksamkeit von konkretem vs. abstraktem Eco-Driving Feedback*. 02.2017

[28] PRACTICE, SURFACE VEHICLE R.: *OPERATIONAL DEFINITIONS OF DRIVING PERFORMANCE MEASURES AND STATISTICS*. 02.2013

[29] VOORT ., Mascha van d. ; ., Mark S. D. ; MAARSEVEENS, Martin van: *A prototype fuel-effciency support tool*. 06.2000

[30] SHORTRIDGE, Woodbury ; GABLE, Thomas M.: *AUDITORY AND HEAD-UP DISPLAYS FOR ECO-DRIVING INTERFACES*. 06.2017

[31] HAMMERSCHMIDT, Jan ; TUENNERMANN, Rene ; HERMANN, Thomas: *ECOSONIC: TOWARDS AN AUDITORY DISPLAY SUPPORTING A FUEL-EFFICIENT DRIVING STYLE*. 09.2014

[32] SCHAEZL, Johannes: *How Effective are Persuasive Technologies in Automotive Context?* 2015

[33] MARCUS, Aaron ; ABROMOWITZ, Scott: *The Driving Machine: Mobile UX Design That Combines Information Design with Persuasion Design*. 2013

[34] VOIGT, D.: *Schlau fahren – Sprit sparen. Expert Verlag, 3. aktualisierte Auflage*. 2012

[35] GUERROUDJ, R.: *Informationsgesellschaft und Globalisierung - sicheres, ökonomisches und umweltfreundliches Fahrverhalten, Universität Ulm*. 2016

[36] TSCHOEKE, H. ; HEINZE, H.: *Einige unkonventionelle Betrachtungen zum Kraftstoffverbrauch von PKW . Magdeburger Wissenschaftsjournal*. 2001

[37] SCHUTT, F.: *Der Grüne Fuß . Books on Demand*. 2002

[38] DE VLIEGER, I. ; DE KEUKELEERE, D. ; KRETZSCHMAR: *Environmental effects of driving behaviour and congestion related to passenger cars. Atmospheric Environment, Volume 34, Issue 27, p. 4649-4655*

[39] BEUSEN, B. ; BROEKX, S. ; DENYS, T. ; BECKS, C. ; DEGRAEUWE, B. ; GIJSBERS, Scheepers M. ; K., Gova.: *Using on- board logging devices to study the longer-term impact of an eco-driving course. Transportation Research Part D, vol. 14, no. 7, pp. 514-520.*

[40] FIREBASE: *Guides*. Version: 11.2018. https://firebase.google.com/docs/ios/setup

[41] FIREBASE: *Authentication*. Version: 11.2018. https://firebase.google.com/docs/auth/

[42] FIREBASE: *Cloud Firestore*. Version: 11.2018. https://firebase.google.com/docs/firestore/

[43] FIREBASE: *Cloud Functions for Firebase*. Version: 11.2018. https://firebase.google.com/docs/functions/

[44] BOARO, Lorenzo: *Design Patterns on iOS using Swift*. Version: 11.2018. https://www.raywenderlich.com/477-design-patterns-on-ios-using-swift-part-1-2

Name: Ryad Guerroudj Matrikelnummer: 584669

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Ryad Guerroudj