Hochschule Konstanz
Fakultät Informatik

Implementierung einer GUI für ein Infotainmentsystem eines Fahrzeugs

Projektausarbeitung im Rahmen der Vorlesung

Mensch-Maschine-Interaktion

Vorgelegt von:

Marten Kreis
[redacted]

Niklas Kugler
[redacted]

Fach: MMI

Fachsemester: 6

Studiengang: Automobilinformationstechnik

Lehrkraft: Prof. Dr. Umlauf

Ort und Datum: Konstanz, 02.Juli 2021

1 Inhalt

2 Kurze Beschreibung bzgl. Ziel des Projekts

Ziel des Projekts war es im Rahmen der Vorlesung *Mensch-Maschine-Interface* ein Infotainmentsystem eines modernen Fahrzeugs zu implementieren. Modernes Infotainmentsystem deshalb, weil es ein vollständiges Touchscreen-basiertes Infotainmentsystem ist ohne mechanische Tasten. In der Mittelkonsole des Fahrzeugs ist ein haptisches Display verbaut, das die Tastenbetätigungen nachahmen soll.

Das implementierte Infotainmentsystem soll in die Kategorie eines modernen Infotainmentsystem eingeordnet werden können und wurde in seiner Grundstruktur und dem Design von dem Infotainmentsystem des *AUDI A6 C8* inspiriert. Das Infotainment stellt kein Anspruch auf Vollständigkeit und Funktionalität, sondern wurde auf ausgewählte Teilfunktionen beschränkt. Darunter fallen ein Navigationsystem, ein Media-Player, ein Radio und ein Fahrzeugmenü. Außerdem wurde das Infotainmentsystem implementiert mit dem Ziel eine GUI mit einer Mensch-Maschine-Interaktion zu programmieren, worauf entsprechend der Fokus liegt.

Abbildung 1: Infotainmentsystem AUDI A6 C8, www.audi-mediacenter.com

3 Allgemeine Beschreibung des Infotainmentsystems

Im Rahmen dieses Projekts wurde ein Infotainmentsystem, wie es in realen PKWs üblich ist, entworfen und implementiert. Das Hauptmenü des Infotainmentsystems ermöglicht dem Benutzer eine der fünf entworfenen Applikationen auszuwählen, welche jeweils in eigenen Widgets implementiert sind. Eine zentrale Funktion des Systems ist eine Navigationsapplikation, welche eine Strecke mit Start- und Zielpunkt einlesen kann und eine Route berechnet. Die berechnete Trajektorie wird anschließend auf einer Karte dargestellt. Die Eingabe des Start- & Zielpunktes ist sowohl mittels Breiten- & Längengrade möglich als auch mit Ortsbezeichnungen und Postleitzahlen. Ein weiteres wichtiges Element ist das Radio, welches in jedem realen Fahrzeug ebenfalls Teil des Infotainmentsystems ist. Das Radio ermöglich die standardmäßige Auswahl von 5 verschiedenen Radiosendern, welche als MP3 Stream empfangen und abgespielt werden können. Zusätzlich können bis zu fünf weitere beliebige Sender als Favoriten hinzugefügt werden, indem der MP3 Link dem System zur Verfügung gestellt wird. Bei Bedarf kann die Favoritenliste wieder gelöscht werden. Zudem ist es möglich die Lautstärke des Radios anzupassen oder ganz stumm zu schalten. Neben dem Radio gibt es den Mediaplayer, welcher es ermöglicht im Voraus abgespeicherte Musikstücke abzuspielen. Bei einem realen Infotainmentsystem würde hierrunter z.B. auch die Möglichkeit des

Abspielens von Musik fallen, die von einem externen Speichermedium, wie z.B. einem Smartphone oder einem USB-Stick, fallen. Im Rahmen dieses Projekts wurde lediglich die Benutzeroberfläche des Mediaplayers mit 3 Beispieltracks implementiert. Die Möglichkeit Musik von einem anderen Medium zu spielen, besteht nicht. Der Mediaplayer ermöglicht die Auswahl zwischen verschiedenen Tracks, manuelles Starten und Stoppen des Players durch den Benutzer, sowie auch hier die Einstellung der Lautstärke. Radio und Mediaplayer können nicht gleichzeitig Musik wiedergeben. Wird einer der beiden Funktionen gestartet, sorgt das System dafür, dass die andere Komponente entsprechend das Abspielen einstellt. Das vierte Untermenü ist das Fahrzeug Menü. Dieses ermöglicht dem Benutzer Einstellungen bezüglich dem Fahrmodus, dem Klima oder den Fahrassistenzsystemen zu machen. Das Fahrmodus Menü zeigt ein 3D Modell des Fahrzeugs und ermöglicht dem Benutzer die Auswahl aus vier verschiedenen Fahrmodi. Das Klima-Widget ermöglicht die Einstellung der Lufttemperatur für Fahrer und Beifahrer, sowie die Möglichkeit die Klimaanlage aus und anzuschalten. Das Fahrassistenz-Widget enthält einige Schalter, die das Ein- und Ausschalten verschiedener Fahrassistenzsysteme, wie dem Parkassistenten oder dem Spurhalteassistenten. Zudem kann die Lautstärke der akustischen Warnung durch den Parkassistenten angepasst werden. Zuletzt gibt es die Einstellungsfunktion, welche es dem Benutzer ermöglicht die Hintergrundfarbe aller Fenster anzupassen. Dabei kann aus sechs verschiedenen Optionen gewählt werden.

Das Infotainmentsystem soll keine Desktopanwendung darstellen, sondern für den Einbau in die Mittelkonsole eines Fahrzeugs konzipiert sein. Deshalb ist die Fenstergröße bewusst nicht auf Vollbild ausgelegt. Außerdem ist die Bedienung auf ein Touchscreen-System ausgelegt, dass hauptsächlich durch Finger bedient wird, und nicht für eine Bedienung mit Tastatur und Maus. Dies wurde zum Großteil umgesetzt bis auf Texteingaben z.B. beim Routing. Diese Texteingaben werden normalerweise ermöglicht durch das Einblenden einer Tastatur im Bildschirm, worauf in diesem Projekt verzichtet wurde.

4 Anforderungen an das Infotainmentsystem

In folgendem Kapitel werden die Anforderungen an das im Rahmen dieses Projekts implementierte Infotainmentsystem gestellt.

Eines der zentralen Ziele des implementierte Infotainmentsystems ist eine gute Benutzerfreundlichkeit zu garantieren, damit das System möglichst von allen Personengruppen bedient werden kann und dabei kein Vorwissen benötigt wird. Diese Anforderung hat auch in realen Infotainmentsystemen von PKWs hohe Relevanz, da die Fahrzeuge von zahlreichen Kunden in unterschiedlichen Ländern und mit stark verschiedenem technischem Vorwissen gekauft werden. Trotzdem soll es jedem möglich sein das Infotainmentsystem ohne vorherige Einweisung bedienen zu können und angestrebte Ziele umzusetzen.

Deshalb gilt es die Menüs logisch zu strukturieren, sodass die gewünschte Funktion ohne Suchen gefunden werden kann. Die Menüs sollten dementsprechend nicht zu tief sein und mit prägnantem klar beschreibendem Namen versehen werden. Auch Benutzer die in diesem Bereich noch nicht über ausgeprägtes deklaratives Wissen, z.B. der Strukturierung des Menüs sollen in der Lage sein, das Infotainmentsystem zu bedienen. Außerdem können Icons / Abbildungen genutzt werden, um die Abstraktionsebene der GUI zu verringern und den Wiedererkennungswert zu erhöhen. Zudem

können Abbildungen dabei helfen, dass der Benutzer richtige Schlussfolgerungen (deklaratives Wissen) trifft. Somit können Benutzer, welche zwar das Infotainmentsystem noch nicht benutzt haben, allerdings über Vorwissen der verschiedenen Komponenten, wie z.B. ein Radio oder ein Navigationssystem verfügen, gezielt durch die Menüs Navigieren um zu der gewollten Funktion kommen.

Eine weitere Anforderung an ein Infotainmentsystem ist die ansprechende Gestaltung der Benutzeroberfläche, die vom Design zu dem Fahrzeug passen sollte. Das in diesem Projekt entworfene Infotainmentsystem ist für einen modernen Sportwagen konzipiert. Die grafische Benutzeroberfläche sollte somit neuzeitlich und elegant gestaltet sein.

Die letzte zentrale Anforderung an das Programm in Bezug auf die Darstellung ist eine Konsistenz der grafischen Benutzeroberfläche zu garantieren, welche sich durch alle Menüs zieht. So sollen Buttons, die identische Aktionen durchführen beispielsweise gleich aussehen. Dies hilft dem Benutzer sich zurechtzufinden und den Umgang mit dem System schneller zu lernen.

Neben den Anforderungen an die Gestaltung der Benutzeroberfläche, gibt es auch Anforderungen an die Strukturierung des Codes. Zentraler Aspekt hierbei ist eine strikte Trennung zwischen logischen Komponenten des Codes und der Benutzeroberfläche. Die Logikkomponenten interagieren mit den GUI Klassen, indem diese über Methoden den Logiklassen mitteilen, dass der Benutzer eine Interaktion getätigt hat. Die Kapselung des Codes wird mit Hilfe von Klassen umgesetzt. Bei allen Applikationen des Infotainmentsystems, die über einen Logikanteil verfügen sollen jeweils zwei Klassen implementiert werden. Eine der beiden Klassen beschreibt die zugrundeliegende Logik der Applikation, währen die zweite Klasse die Auswertung der Benutzerinteraktion übernimmt.

5 Tools / Verwendete Programmiersprache

Das Projekt wird mit den Programmiersprachen C++ und QML (Qt Modeling Language) ungesetzt. Zur Entwicklung der grafischen Benutzeroberfläche wird mit Qt 5.15.2 gearbeitet. Die genaue Konfiguration zum Kompilieren des Projekts ist in Abbildung 2 dargestellt. Als integrierte Entwicklungsumgebung (IDE) wird der Qt Creator verwendet, welcher zum einen die Bearbeitung der Quellcodes, als auch das Design der Benutzeroberfläche ermöglicht. Die meisten Funktionen des Infotainmentsystems sind mit C++ umgesetzt. Lediglich bei der Navigationsapplikation und einigen grafischen Elementen, wie der Temperaturanzeige des Klimamenüs oder dem 3D Modell des Fahrzeugs kommt QML zum Einsatz. C++ ermöglicht Objektorientiertes Programmieren, welches insbesondere bei der Kapselung des Codes eine wichtige Rolle spielt. Zudem ist die Programmiersprache in der Automobilbranche häufig vertreten und wird auch dort zum Teil zur Implementierung von Infotainmentsystemen genutzt. Die grafischen Oberflächen bestehen vor allem aus QWidgets, welche mit Hilfe des Qt Creators designed wurden. QML ist eine von der *The Qt Company* entwickelte Programmiersprache, welche auf JSON basiert. QML Code kann mit C++ mittels des Signals / Slot Mechanismus einer QObject Klasse interagieren und eignet sich besonders gut zum Entwurf von benutzerdefinierten Designs. Außerdem lassen sich Animationen deutlich besser mit QML umsetzen. Für die Map wird zudem eine Software-Bibliothek benötigt, damit SSL und TLS verwendet werden können. Dazu wurde noch die externe Bibliothek OpenSSL eingebunden (OpenSSL, http://slproweb.com/products/Win32OpenSSL.html).

Abbildung 2: Qt Konfiguration

6 Umsetzung / Implementierung

In diesem Kapitel wird die Implementierung der Bereiche Navigation, Radio, Mediaplayer, Fahrzeug (Fahrmodus, Klimaanlage, Fahrerassistenzsysteme) und Einstellungen genauer betrachtet.

6.1 Hauptmenü

Das Hauptmenü des Infotainmentsystems besteht zum einen aus einer fest angeordneten Struktur, bestehend aus einem Home Buttons, mit dem man immer zurück in das Hauptmenü kommt, sowie einer Menüauswahl. Der Benutzer kann mit Hilfe dieser Leiste schnell zwischen den wichtigsten Funktionen wechseln und erhält zudem einen guten Überblick, wo er sich gerade befindet. Die aktuell ausgewählte Funktion ist orange unterlegt. Um in das Hauptmenü zurück zu gelange, kann der Benutzer entweder auf den Homebutton in der linken oberen Ecke gehen, oder in dem Seitlichen Reiter auf Menü gehen. Diese Redundanz hilft besonders unerfahrenen Benutzern bei der Interaktion mit dem System. Zudem ist für den Homebutton statt einem Text ein Icon hinterlegt, was den Abstraktionsgrad verringert und die Steuerung erleichtert. Diese Struktur kommt im Hauptmenü und in allen anderen Untermenüs vor und stellt somit einen festen Seitenleiste dar, mit der zu jederzeit zu den wichtigsten Funktionen gewechselt werden kann.

Zentral sind die acht Kacheln angeordnet, über welche der Benutzer jeweils zu den unterschiedlichen Bereichen gelangen kann. Funktionsfähig sind dabei Radio, Navigation, Media Player, Fahrzeug und Einstellungen. Die Funktionalität für die letzten drei Bereiche Telefon, Nachrichten und Phone Apps wurde nicht implementiert und somit dienen die dazugehörigen Kacheln nur dazu, das visuelle Erscheinungsbild eines Infotainmentsystems zu bewahren.

6.2 Navigation

Das Navigationssystem besteht aus einer NavigationWidget-Klasse, die das Gerüst der Navigation Applikation bildet. Dazu gehört die Randstruktur, mit den PushButton’s für den Menü-Wechsel. Die

eigentliche Implementierung des Navigationsystems ist auf QML basiert und ist mit einem QuickWidget eingebettet.

Für die Map wird ein mapPlugin verwendet, welches den Map-Provider *OpenStreetMap* verwendet. In Abbildung 3 ist das Navigationssystem mit seiner Karte abgebildet und zeigt in seiner initialen Konfiguration einen vordefinierten Standort an (hier: HTWG Konstanz). Dies soll die Positionierung der Karte basierend auf der aktuellen Position nachahmen. Das Rein- und Rauszoomen ist möglich, indem das Mausrad bewegt wird und befindet sich auf einem vordefinierten Zoom-Level. Die Navigations-Anwendung hat zwei Hauptfunktionen. Zum einen ist ein Geocoding und zum anderen ist ein Routing möglich. Beim Geocoding ist es möglich. Sowohl beim Geocoding als auch beim Routing ist es möglich entweder Koordinaten einzugeben oder eine Suchadresse für das Geocoding bzw. eine Start- und eine Zieladresse für das gewünschte Routing. Das Fenster zur Eingabe von Koordinaten oder Adressen ist in Abbildung 5 beispielhaft für die Eingabe einer Start- und Zieladresse für das Routing dargestellt. Bei diesem Anwendungsbeispiel gibt es jeweils für die Start- und die Zieladresse drei Eingabefelder, bei denen der geforderte Input jeweils mit vorangestellten Labels eindeutig definiert ist. Bei dem Adress-Routing ist beispielsweise für die Start- und Zieladresse jeweils die Straße, der Ort und das Land einzugeben. Die Implementierung dazu ist in *RouteAdress.qml* zu finden. Entsprechend dafür gibt es für die anderen drei Funktionalitäten (Routing mit Koordinaten, Geocoding mit Koordinaten, Geocoding mit Adresse) jeweils ein entsprechendes QML-file. In den einzelnen Eingabemenüs sind die Eingabefelder schon im Voraus ausgefüllt und soll eine direkte Übernahme des akutellen Standorts darstellen bzw. ein Beispiel für mögliche Eingaben darstellen. Unten rechts in den jeweiligen Eingabemenüs gibt es zusätzlich noch drei Buttons, die zur intuitiven Bedienung mit ihrer Funktion gekennzeichnet sind. Der Button mit der Beschriftung „Proceed“ übernimmt die Eingabe und führt die Nachfrage der Funktion aus. Mit dem Button „Clear“ ist es möglich die vordefinierten Eingaben mit einem Knopfdruck zu löschen und mit dem Button mit der Beschriftung „Close“ kann das Eingabefenster beendet werden. Beim Aufrufen des Navigation-Untermenü erscheint die Ansicht, die in Abbildung 3 zu sehen ist. Neben der Karte ist oben links im QtWidget ein Button zu sehen. Wenn dieser Button gedrückt wird, erscheinen die Buttons der zuvor beschriebenen Untermenüs, mit den man in die jeweiligen Eingabefelder der verschiedenen Map-Funktionalitäten gelangt. In Abbildung 4 sind die eingeblendeten Buttons zu sehen (von links beginnend: Geocoding mit Adresse, Geocoding mit Koordinaten, Routing mit Adresse und Routing mit Koordinaten). Der Button links daneben, der für das Ein- und Ausblenden zuständig ist, ist im eingeblendeten Zustand dunkel gefärbt. Beim erneuten Betätigen des Buttons werden die Funktionalitäten wieder ausgeblendet und der Button färbt sich wieder wie ursprünglich. In Abbildung 4 ist ein beispielhaftes Routing von Konstanz nach München dargestellt. In Abbildung 5 ist dazu noch jeweils zu Start- und Endposition, sowie zum Eingabefeld eine Detailansicht zu sehen.

Die Implementierung der Map basiert auf den von Qt bereitgestellten Online-Dokumentation. Zum einen war dort ein einfaches Implementierungsbeispiel für eine Map zu finden (Minimal Map, QML, https://doc.qt.io/qt-5/qtlocation-minimal-map-example.html). Zum anderen gibt es eine allgemeine Dokumentation zu Maps und Navigation in QML (Maps und Navigation, https://doc.qt.io/qt-5/location-maps-qml.html).

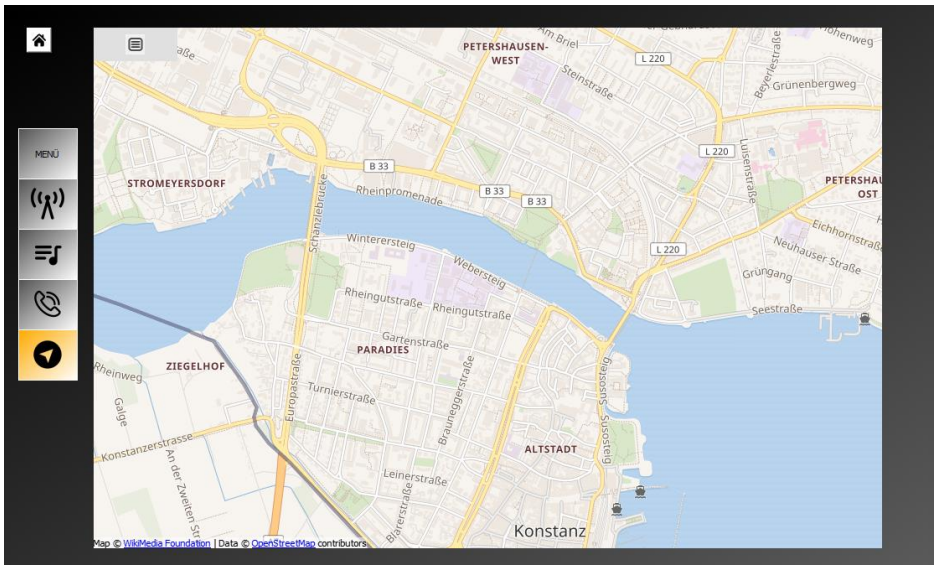

Abbildung 3: Navigationssystem

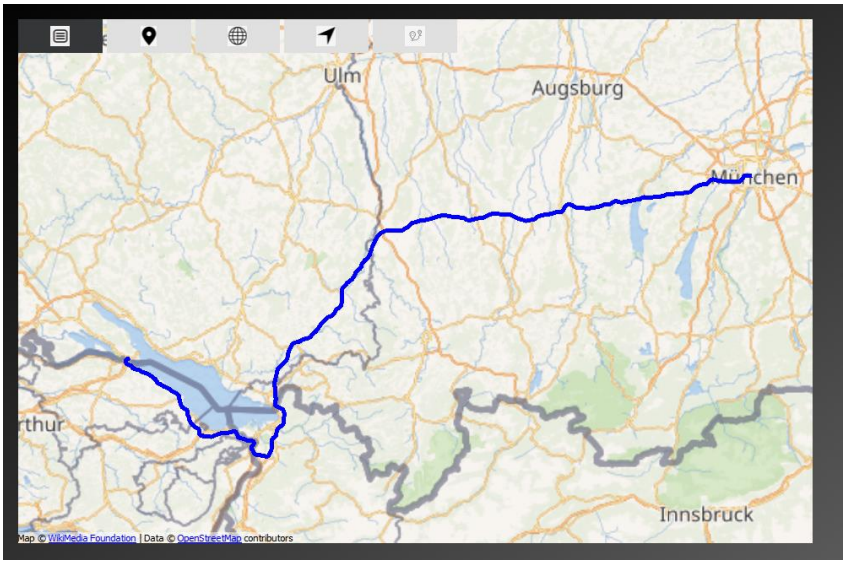

Abbildung 4: beispielhaftes Routing von Konstanz nach München

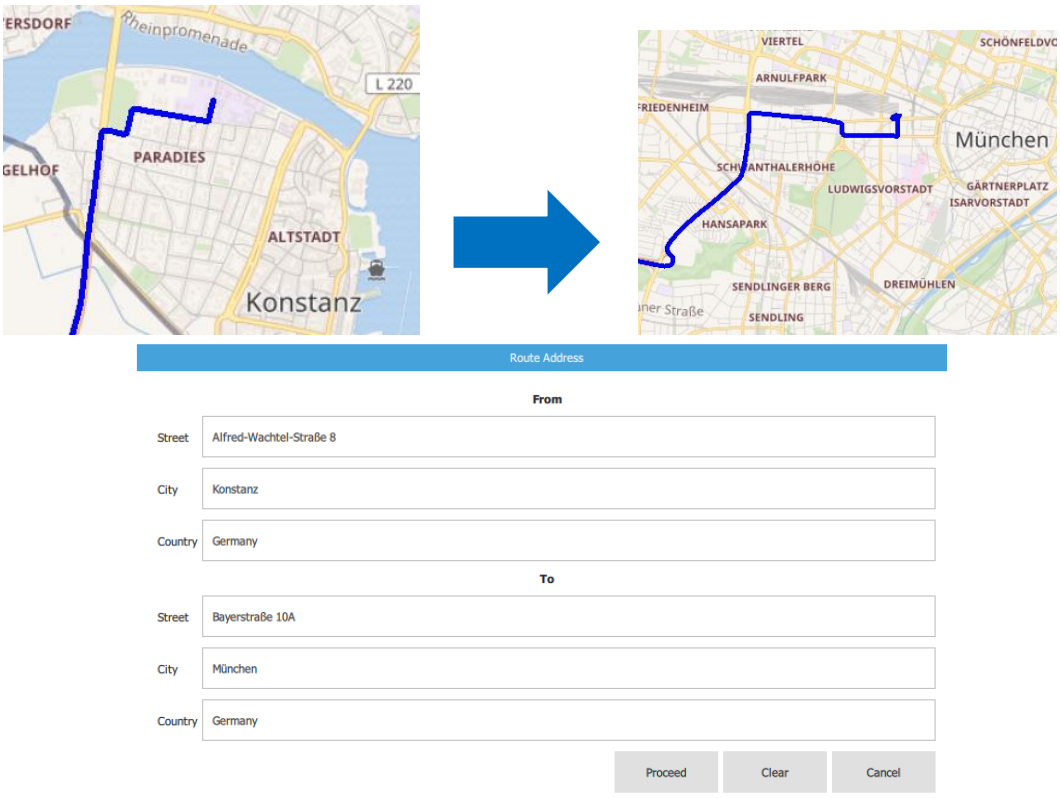

Abbildung 5: Beispielhaftes Routing (Detailansicht)

6.3 Radio

Die Radioimplementierung des Infotainmentsystems ist die Klassen *MultimediaPlayer* und *MultimediaPlayerWidget* aufgeteilt. Die *MultimediaPlayer* Klasse enthält den Logikanteil, in welcher z.B. der gewählte Sender abgespielt wird, die Laustärke angepasst werden kann, oder die Favoritenliste verwaltet wird. Die *MultimediaPlayerWiget* Klasse definiert die grafische Benutzeroberfläche der Radioapplikation und reagiert auf den Input des Benutzers, indem eine Methode der Logikklasse aufgerufen wird.

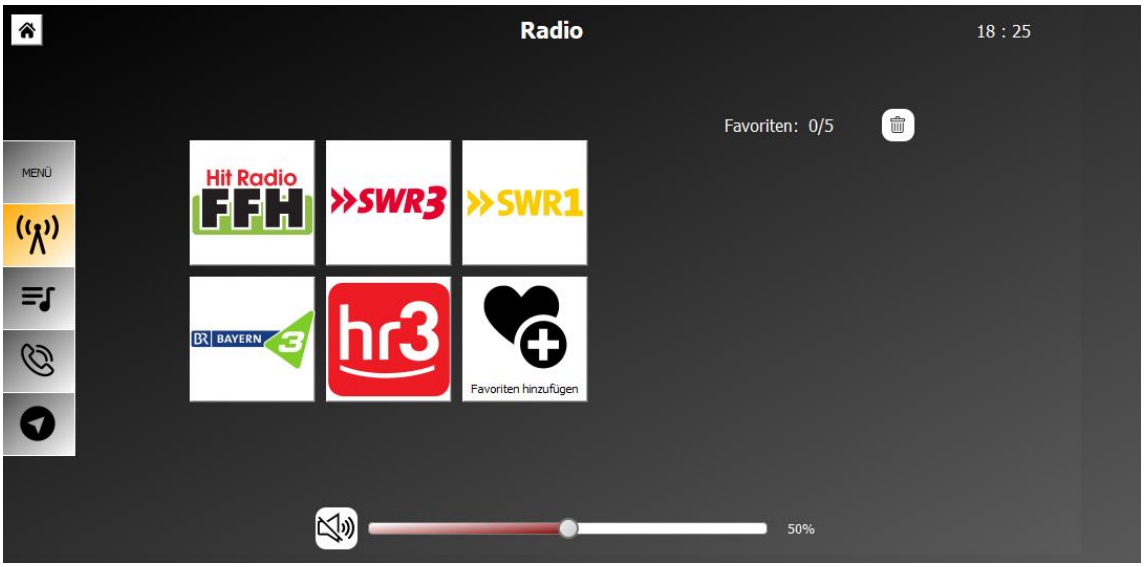

Abbildung 6: Benutzeroberfläche des Radios

In Abbildung 6 ist die grafische Benutzeroberfläche des Radios dargestellt. Zentral sind die Fünf standardmäßig vorhandenen Radiosender als Pushbuttons angeordnet, sowie die Option weitere Favoriten manuell hinzuzufügen. Wird einer der Pushbuttons gedrückt wird dieser orange hinterlegt und der entsprechende Sender wird mit Hilfe der Logikklasse abgespielt. Um das Abspielen zu stoppen, kann der Pushbutton erneut gedrückt werden. Die Lautstärke lässt sich mit Hilfe des Sliders unten einstellen. Neben der Position des Sliders als Referenz für die Lautstärke ist gibt es zudem ein Label, welches die Lautstärke in Prozent angibt. Um das Radio stummzuschalten, gibt es einen Pushbutton neben dem Slider, welcher mit einem Icon eines durchgestrichenen Lautsprechers versehen ist. Beim Betätigen dieses Buttons wird dieser ebenfalls orange hinterlegt und die Logikklasse setzt die Lautstärke des QMediaPlayaers auf 0. Um neue Favoriten hinzuzufügen kann der Pushbutton mit dem *Plus* Symbol verwendet werden. Neben dem Icon gibt es hier zusätzlich einen Text, um die Funktion dieses Buttons für den Benutzer zweifelsfrei zu erklären. Wird dieser Button gedrückt, erscheint ein Popup Fenster, welches den Benutzer auffordert, den Namen des Senders und den MP3 Link zu abspielen einzugeben. Alle Favoriten werden anschließend auf der rechten Seite angezeigt. Die Maximale Anzahl der Favoriten beschränkt sich auf 5 Kanäle, welches dem Benutzer durch die Zahl neben der Favoritenliste verdeutlicht wird. Sind alle fünf Favoritenslots belegt und der Benutzer nutzt dennoch die Funktion einen weiteren Favoriten hinzuzufügen, wird im durch das Popup Fenster erklärt, dass alle Favoritenplätze belegt sind. Die Eingabe von Sendernamen und MP3 Link ist nicht möglich. Darüber hinaus ist der Benutzer in der Lage die Favoritenliste zu löschen. Hierfür gibt es einen Pushbutton, welcher mit einem Mülleimer-Icon ausgestattet ist. Wird dieser Button gedrückt wird der Benutzer nochmals gefragt, oder die Favoritenliste wirklich gelöscht werden soll. Dies dient dazu, das System vor fehlerhaften Eingaben zu Schützen und dem Benutzer die Möglichkeit zu geben, die zuvor getroffene Entscheidung zurückzunehmen. Bestätigt der Benutzer dies jedoch, wird die Favoritenliste gelöscht.

Das Abspielen von Liveradiosendern wird mit Hilfe von MP3 Streams erreicht, die direkt von den Sendern zur Verfügung gestellt werden. Die QMediaPlayer Klasse, eine von der *The Qt Company* bereitgestellt Funktion, ermöglicht die Eingabe einer MP3 URL. Auf diese Klasse wird im Rahmen dieses Projektes zurückgegriffen. Der Klasse wird abhängig von der Benutzereingabe eine URL zu einem MP3 Stream bereitgestellt und anschließend die Wiedergabe des Mediums aktiviert. Dies geschieht in der Logikklasse des Radios. Außerdem erlaubt es der QMediaPlayer die Lautstärke, sowie den Player Status (Abspielen/Stoppen) mit Hilfe von Methoden zu wählen.

6.4 Media Player

Die Implementierung des Mediaplayers ist ebenfalls in eine Logiklasse *mediaPlayer* und eine GUI Klassen *mediaPlayerWidget* aufgeteilt. Auch hier enthält die Logikklasse, wie bereits bei dem Radio, den QMediaPlayer, mit welchem eine vorgegebene MP3 Datei abgespielt werden kann. Die Klasse mediaPlayerWidget registriert den Benutzerinput und ruft entsprechend Methoden der Logikklasse auf.

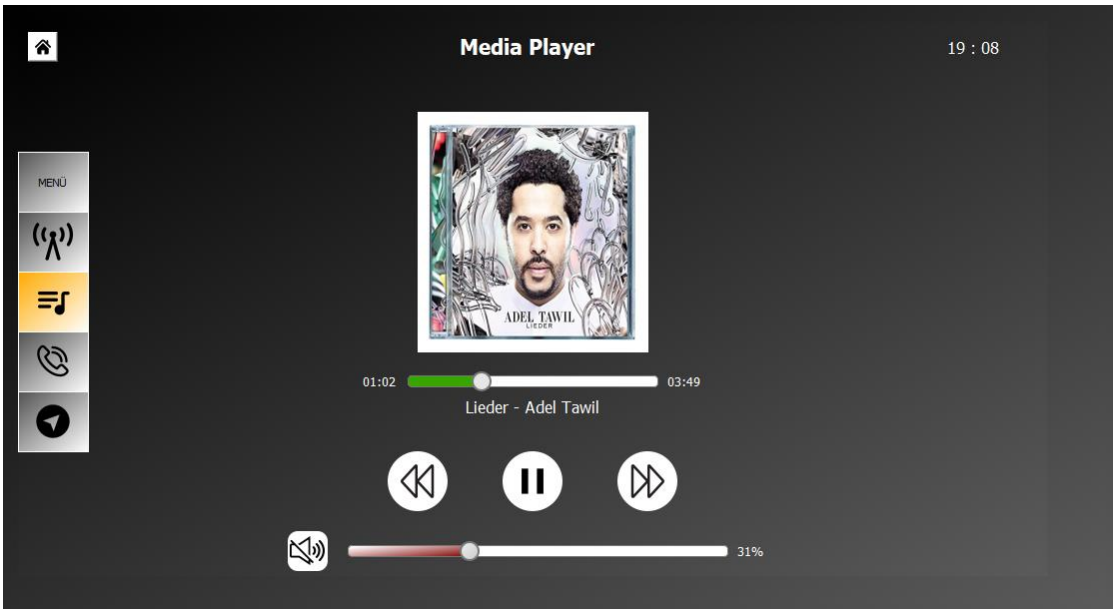

Abbildung 7: Benutzeroberfläche des Mediaplayers

Abbildung 7 zeigt die grafische Benutzeroberfläche des Mediaplayers. Auf der linken Seite ist wie zuvor die Menüleiste, sowie der Homebutton.

Im Zentrum des Fensters ist das Cover des Songs dargestellt. Mithilfe des Covers kann sich der Benutzer orientieren, welches Lied gerade abgespielt wird. Zudem wird oberhalb des Play Buttons der Titel des Lieds, sowie der Interpret dargestellt. Der aktuelle Wiedergabezeitpunkt kann mit Hilfe des grün hinterlegten Sliders abgeschätzt werden, oder anhand der Zeitangabe auf der linken Seite abgelesen werden. Auf der rechten Seite des Sliders ist die maximale Wiedergabe Zeit des Lieds angegeben. Der Benutzer ist zudem in der Lage durch Bewegung des Sliders den Wiedergabezeitpunkt anzupassen. Die Wiedergabe kann mit dem Play Button gestartet / gestoppt werden, welcher abhängig von seinem Zustand das Icon wechselt, um Missverständnissen, in welchem Modus sich der Player gerade befindet, vorzubeugen. Mit den Buttons rechts und links von dem Playbutton kann der Track danach, bzw. davor abgespielt werden. Auch hier ist es möglich die Lautstärke mittels einem Slider einzustellen, oder den Player ganz stumm zu schalten. Der Slider für die Lautstärke und der Stummschaltknopf kommen bereits in dem Radio vor und sind in ihrem Layout identisch, um Konsistenz zu bewahren.

Wird der Mediaplayer durch den Benutzer aktiviert, sorgt das Programm dafür, dass die Wiedergabe des Radios angehalten wird, falls der Benutzer noch einen Sender ausgewählt hat. Dasselbe Verhalten gilt auch für den umgekehrten Fall, also dass der Benutzer zuerst den Mediaplayer gestartet hat und anschließend einen Radiosender auswählt. Dann wird automatisch die Wiedergabe des Mediaplayers gestoppt.

Es stehen 3 Beispieltracks zur Verfügung, die dazu dienen die Funktionalität des Mediaplayers zu demonstrieren. In einem realen Infotainmentsystem würden die Lieder in der Regel über ein externes Speichermedium, wie ein Smartphone oder USB-Stick bereitgestellt.

6.5 Fahrzeug

Das Fahrzeugmenü besteht lediglich aus der Klasse *vehicleWidget*, welches dem Benutzer ermöglicht eines der drei Untermenüpunkte „Fahrmodus“, „Klima“ und „Fahrerassistenz“ auszuwählen, wie in Abbildung 8 dargestellt.

Abbildung 8: Menüauswahl Fahrzeug

Die Buttons sind sowohl mit Icons als auch mit Beschriftung versehen, um dem Benutzer möglichst genau zu beschreiben was in den jeweiligen Untermenüs befindet. Verfügt der Nutzer bereits über deklaratives Wissen bezüglich der einzelnen Komponenten, wird er so angeregt dies zu nutzen, um Schlussfolgerungen der jeweiligen Inhalte zu ziehen.

Das erste Untermenü ermöglicht dem Benutzer den Fahrmodus einzustellen. Hierbei kann er zwischen der 4 Modi „Effizienz“, „Komfort“, „Dynamisch“ und „Auto“ wählen, wie in Abbildung 9 ersichtlich ist. ZU dem Fahrzeugmenü kann jederzeit über den Rückkehrknopf (linke Ecke) gelangt werden.

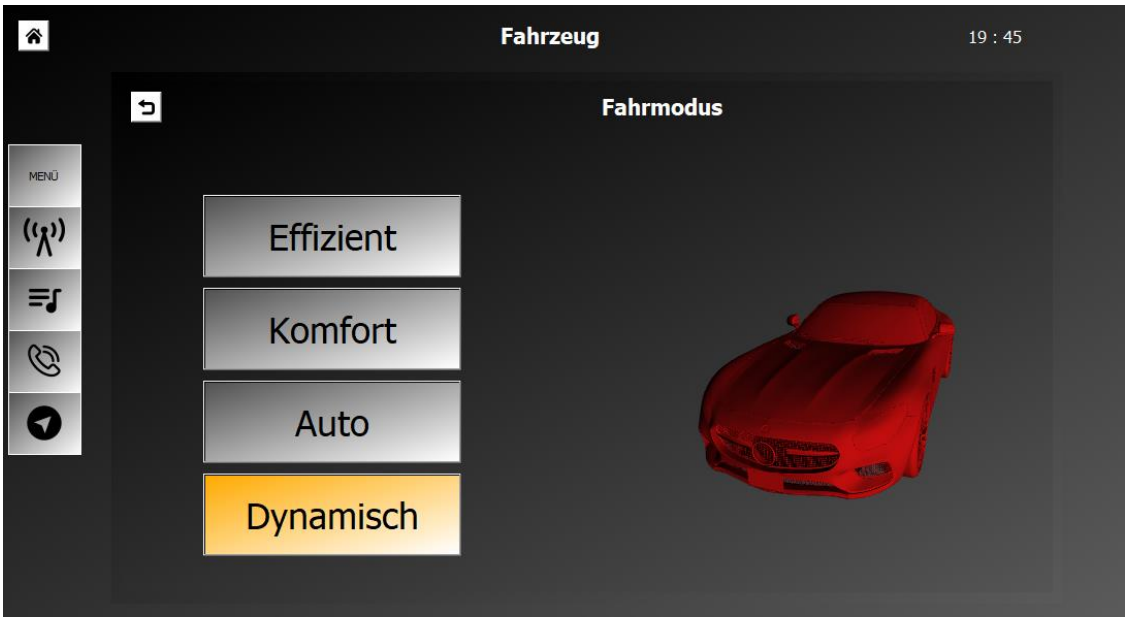

Abbildung 9: Benutzeroberfläche Fahrmodus

Wird einer der Pushbuttons aktiviert, wird der Hintergrund, wie auch schon zuvor z.B. beim Radio, orange hinterlegt. Auf der rechten Seite ist zudem ein rotierendes 3D Modell des Fahrzeugs dargestellt.

Das Modell steht unter folgendem Link zum Download zur Verfügung und wurde nicht selbst mit einer Modellierungssoftware erstellt: https://www.cgtrader.com/free-3d-models/car/sport/mercedes-amg-gt-s-2016-3dmodel. Als Dateiformat wird OBJ verwendet. Um das Modell in das Programm zu integrieren, wird es mit Hilfe von QML geladen. In dem QML File *car_model.qml* findet ebenfalls die Eulerrotation um die Z-Achse des Fahrzeugs statt. Damit ein 3D Modell mittels QML gerendert werden kann, muss mit Hilfe des Maintenance Toolkits von Qt als zusätzliches Paket *Qt Quick 3D* installiert werden.

Das *Klima* Untermenü setzt sich aus der Logikklasse climateControl und der GUI Klasse climateControlWidget zusammen. Die Displays, welche die eingestellte Temperatur von Fahrer und Beifahrer anzeigen sind in QML entworfen und interagieren mit der GUI über einem Signal / Slot Mechanismus.

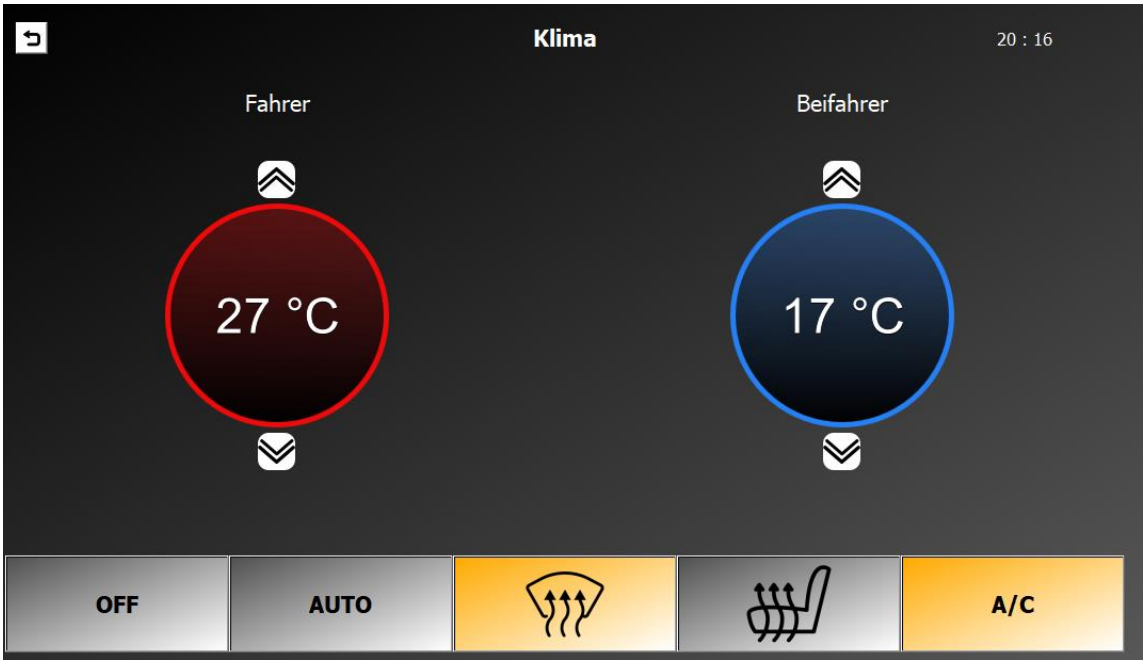

Abbildung 10: Benutzeroberfläche Klima

Die Temperatur für Fahrer und Beifahrer kann individuell über die Pfeilknöpfe eingestellt werden. Zulässig ist ein Bereich von 12°C-36°C. Ist dieser Grenzwert erreicht wird der entsprechende Button deaktiviert. Die aktuelle Temperatur ist innerhalb des Temperaturdisplay ablesbar. Zusätzlich ändert sich die Farbe, abhängig von der Eingestellten Temperatur zwischen Rot, Grau und Blau. Zusätzlich gibt es weitere Pushbuttons, die es dem Benutzer ermöglichen die Klimaanlage ganz auszustellen, die Temperaturregelung automatisch zu übernehmen, die Frontscheibe zu beheizen und die Sitzheizung anzustellen. Ist einer der Pushbuttons aktiviert, wird dieser orange hinterlegt. Die Logikklasse enthält dabei die Zustände der Buttons und der Temperatur und meldet der GUI Klasse, ob einer der Buttons deaktiviert werden soll. Die Änderung der Farbe des Klima Displays findet in dem QML File *klima_dips_driver / klima_disp_codriver* statt und wird durch ein Signal der GUI Klasse ausgelöst.

Das letzte Untermenü ermöglicht die Einstellungen der Fahrerassistenzsysteme, wie in Abbildung 11 dargestellt. Die getätigten Einstellungen sind rein visuell, somit gibt es hierfür keine Logikklasse. Die Switches sind in QML implementiert und über ein QQuickWidget in das ui Formular eingebunden. Die Pushbuttons, die die Lautstärke der Einparkhilfe kontrollieren, werden bei Aktivierung orange hinterlegt. Es kann immer nur einer der drei Buttons aktiviert werden.

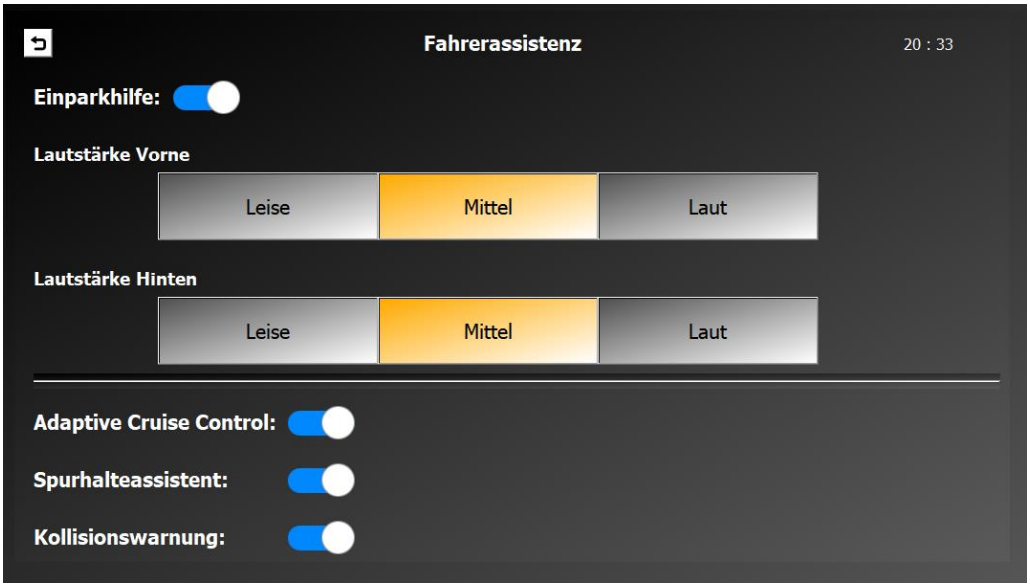

Abbildung 11: Benutzeroberfläche Fahrerassistenz

6.6 Einstellungen

Das Einstellungsmenü teilt sich in die Logikklasse *settings* und die GUI Klasse *settingsWidget* auf. Die Logikklasse enthält die vordefinierten Hintergründe und passt die globale Variable an, die den aktuell ausgewählten Hintergrund enthält.

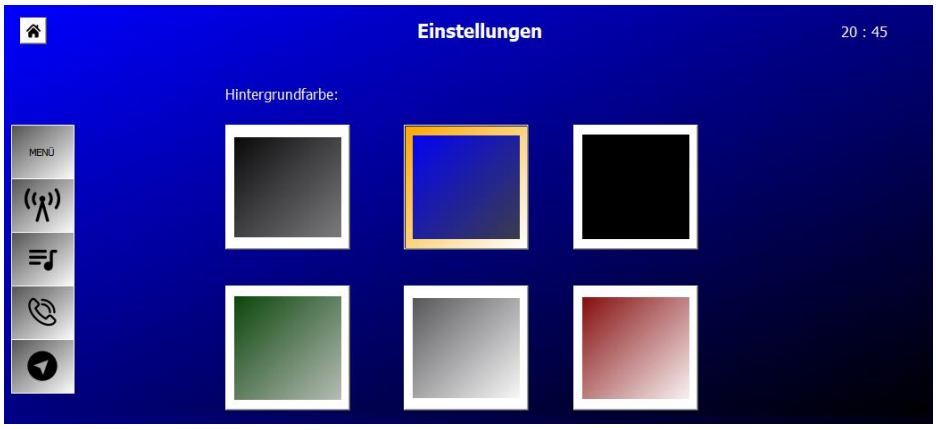

Abbildung 12: Benutzeroberfläche Einstellungen

Wie in Abbildung 12 zu sehen ist, ist es dem Benutzer möglich aus 6 vordefinierten Hintergrundfarben eine zu wählen. Die gewählte Hintergrundfarbe wird anschließend auf alle anderen Widgets übertragen.

7 Fazit

Zusammenfassend lässt sich sagen, dass die Grundfunktionen eines Infotainmentsystems implementiert wurden und dabei die im Kapitel 3 beschriebenen Anforderungen eingehalten wurden. Die Menüs wurden möglichst breit aufgebaut, um das Kurzzeitgedächtnis der Benutzer nicht zu sehr zu beanspruchen. Hierbei wurde ebenfalls auf die 7±2 Regel geachtet, welche besagt, dass ein Mensch in der Regel in der Lage ist bis zu 9 Dinge zeitgleich im Kurzzeitgedächtnis zu behalten. Zudem sind viele Buttons mit Icons und zum Teil auch Beschriftung ausgestattet, um die Orientierung zu erleichtern. Der Benutzer kann so sein deklaratives Wissen nutzen, um Schlussfolgerungen über die Funktionen, die sich hinter einem Button befinden, zu ziehen. Im Allgemeinen bieten Menüdesigns den Vorteil, dass der Benutzer schnell mit dem System vertraut ist, was bei einem Infotainmentsystem in einem Fahrzeug hohe Priorität hat. Zudem werden wenige Tastatureingaben zur Navigation innerhalb des Programms benötigt. Auch dies ist ein wichtiger Faktor, da der Fahrer sich vor allem auf das Fahren konzentrieren soll und somit seine Aufmerksamkeitsspanne für die Bedienung des Infotainmentsystems stark begrenzt ist.

Darüber hinaus werden die acht goldenen Regeln des GUI Designs bei der Implementierung angewendet. Konsistenz wird erzielt, indem z.B. Pushbuttons mit gleichen oder ähnlichen Funktionen identisches Design, aktivierte Buttons sind immer orange hinterlegt und der Benutzer kann über das seitliche Menü oder den Homebutton aus jedem Untermenü direkt wieder zum Hauptfenster zurück gelangen. Das Layout des Systems bleibt außerdem in allen Untermenüs erhalten. Des Weiteren wird darauf geachtet, dass das System von Personen mit unterschiedlichen Erfahrungsstufen bedient werden kann und sollte auch für Personen, die mit dem System zuvor noch nicht in Kontakt getreten sind, gut zu kontrollieren sein. Darüber hinaus erhält der Benutzer zu jeder vorgenommenen Eingabe ein visuelles Feedback, bei dem Klima Display, z.B. auch neben der Gradzahl zusätzlich durch die Farbe des Ringes. Die Benutzerdialoge, beispielsweise bei der Erstellung eines neuen Favoritensenders des Radios sind klar aufgebaut und erklären dem Benutzer genau, was benötigt wird. Ist die maximale Anzahl an Favoriten erreicht, wird dies auch nochmals in dem Dialogfenster verdeutlicht. Nicht reversible Fehler bei der Eingabe, z.B. dem Löschen der Favoritenliste des Radios werden vermieden, indem der Benutzer vorher gefragt wird, ob er diese Aktion wirklich vornehmen wollte. In allen anderen Fällen ist sogar ein sofortiges Rückgängigmachen der ungewollten Aktion durch den Benutzer möglich.