DAIS Dokumentation

Dashboard and Infotainment System

Gruppe 7

Noll, Marco, 870529
Frenzel, Julian, 870716
Schug, Jens Cedric, 870886
Zimmermann, Philipp, 870872

Infotainment

Die Infotainmentanwendung besteht aus einer großen Applikation, welche aus fünf unterschiedlichen, kleineren Modulgruppen zusammengesetzt ist. Diese Modulgruppen beinhalten: Einen Musikspieler, eine Simulation eines Telefons, die Anzeige von autospezifischen Informationen, eine Kartenanwendung und Navigationsmöglichkeit und das Impressum.

Software-Design

Die Datei „main.qml“ beinhaltet alle Daten, die für die Interaktion und das Design der Anwendung wichtig sind. Die Dateien Pageform1.ui.qml bis Pageform5.ui.qml beinhalten nur wenig Code, da sie als Instanz in der „main.qml“ verwendet werden und dort alle wichtigen Elemente (Buttons, Images usw.) hinzugefügt werden. Es ist nicht möglich eine Interaktion von Buttons in „ui.qml“ Dateien zu implementieren, deshalb musste die gesamte Implementierung in die „main.qml“ ausgelagert werden. „ui.qml“-Dateien dienen primär der Darstellung des UI-Inhalts und -Design. In „AddressModel.qml“ und „MP3Model.qml“ stehen Informationen, die für den Musikspieler und bei der Nutzung des Kartenmaterials relevant sind. Die restlichen Dateien implementieren die MQTT-Schnittstelle und initialisieren das Anwendungsfenster. Die empfangenen Daten vom MQTT-Broker werden durch die Klasse ValueSource repräsentiert. In dem Ordner „background“ befinden sich alle verwendeten Icons und Bilder. Alle Musikstücke des Musikspielers sind im Ordnerverzeichnis „music“.

Aufbau der Applikation

Verbindung zum MQTT-Broker

Wenn die „main.qml“ geladen wurde, wird automatisch eine Verbindung zum Host des MQTT-Brokers hergestellt. Dabei müssen folgende Parameter beachtet werden: der Hostname , der Port, der Benutzername und das Passwort. Die Implementierung der MQTT-Schnittstelle und MQTT-Subscription wurde aus dem Beispiel des Tesla-Dashboards (https://github.com/Garzuuhl/DAIS/tree/master/Seafile/dashboard-examples) übernommen , welches zur Verfügung gestellt wurde.

Musikspieler

Der Benutzer kann mit mehreren Buttons interagieren. Musikstücke können durch Drücken dieser Schaltflächen gestartet und gestoppt werden. Es wird die Funktion playMusic bzw. stopMusic aufgerufen. Sie aktualisiert die Bilder der Buttons , damit sie zum jeweiligen Zustand passen und spielt das momentane Musikstück ab bzw. stoppt es. Dabei wird durch einen Timer der aktuelle Zeitpunkt des Musikstücks bestimmt, konvertiert (die Zeitangabe ist in Millisekunden) und danach angezeigt. Entsprechend dieser Zeit wird die Größe eines Rechtecks verändert, d.h. ist das Musikstück bei der Hälfte der Gesamtlänge, wird auch das Rechteck halb so groß. Dadurch sieht der Benutzer immer genau, wie lange der Titel schon gespielt wurde. Zusätzlich besteht die Möglichkeit den letzten und den nächsten Titel abzuspielen. Der Titel des Musikstücks und die Anzeige der Songlänge werden entsprechend aktualisiert.

Beim Musikspieler wurden Inhalte von BenSound (https://www.bensound.com/) verwendet. Momentan können nur die drei Lieder, die im Projektverzeichnis enthalten sind verwendet werden. Es wurde in Erwägung gezogen, dass der Benutzer auch Lieder vom USB-Gerät , von Spotify® oder anderen Diensten laden kann. Aus zeitlichen Gründen wurde dieser Gedanke jedoch revidiert.

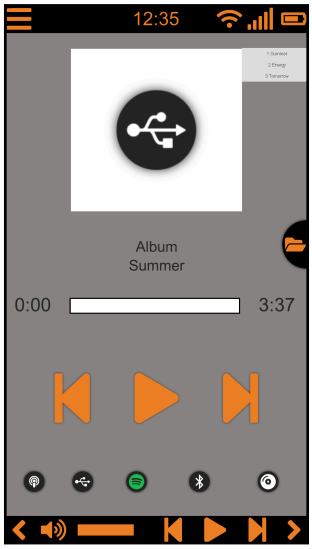

Abb. 1 zeigt den Musikspieler

Telefon

Wie bei einem realen Telefon kann der Benutzer eine Nummer eingeben. Drückt man auf den orangenen Hörer wird ein Telefonanruf simuliert. Will man die Nummer neu eingeben, kann man mit Hilfe der „Zurück“-Taste die Nummer löschen. Momentan ist nur das Löschen der gesamten Nummer möglich. Als Zusatzfeature war ein Kontaktbuch und Favoriteneinträge gedacht.

Da, der Fahrsimulator kein Telefonnetz benutzen kann bzw. das System dieses nicht benötigt, wurde ein Telefonanruf simuliert. Nachdem man die Nummer eingeben hat erscheint ein neuer Bereich in dem drei Kreise, ein Label mit der aktuellen Gesprächszeit und ein Telefonhörer dargestellt werden. Die drei Kreise simulieren den Verbindungsaufbau zwischen den Gesprächspartnern. Durch eine Animation färben sich die Kreise orange. Nach einer festen Zeit wird das Gespräch beendet und man kehrt zur Eingabe der Telefonnummer zurück.

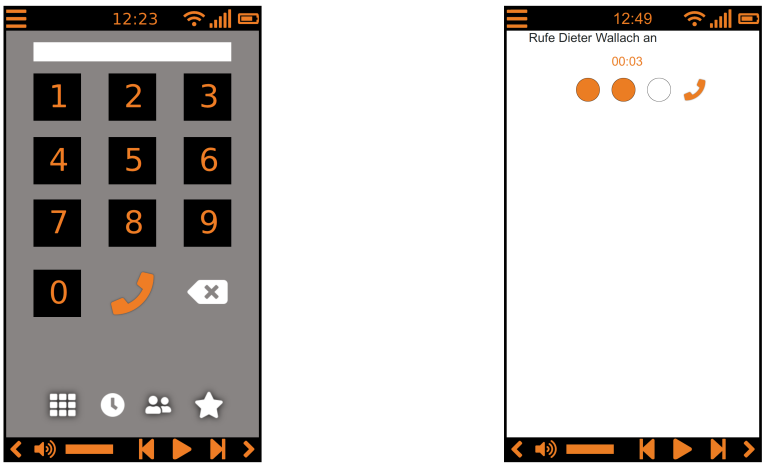

Abb. 2 zeigt das Telefonmenü und die Simulation des Anrufs

Fahrzeuginformationen

Vom MQTT-Broker werden verschiedenste Daten wie Geschwindigkeit, Radumdrehungen, Luftdruck der Reifen usw. versendet. Diese Daten werden bei der Anzeige der Autoinformation verwendet. Der Reifendruck wird an einem Bild von einem Auto angezeigt. Der momentane Kilometerstand wird abgerufen und auch als Label gerendert. Es ist zu beachten, dass der Simulator nicht alle Werte eines realen Autos widerspiegelt, da es sich um ein spezielles Setting handelt. Die MQTT-Anbindung ist funktionstüchtig und die vorhandenen Werte werden auch übernommen.

Abb. 3 Fahrzeuginformationen

Karteninformationen und Navigation

Karteninhalte

Das vierte Modul ist eine Anwendung der Klasse „Map“. Karteninhalte werden anhand eines Plugins namens „osm“, welches Kartenmaterial von OpenStreetMap (https://www.openstreetmap.de/) verwendet, geladen. Damit das Plugin funktioniert muss man einen „TileServer“ angeben. Dieser Server liefert Karteninformationen für verschiedene Anwendungsgebiete. Es gibt z.B. Karten für Radfahrer, für Wanderer und den „normalen“ Straßenverkehr. Die aktuelle Position in einer Fahranwendung (gegeben in Längen und Breitengrad) wird vom MQTT-Broker übermittelt. Diese Standortinformation wird verwendet um die Inhalte der Karte zu aktualisieren bzw. zu positionieren. Der Mittelpunkt der Karte ist immer der aktuelle Standort. Um die Position besser darzustellen wurde ein orangener „MapCircle“ benutzt.

Navigation

Ursprünglich war gedacht, dass der Benutzer einen Begriff in eine Suchleiste eingibt und die Nominatim-API (http://nominatim.openstreetmap.org) den Namen in eine Geoposition (enthält Längen und Breitengrad des Suchorts) umwandelt. Diese Information kann durch das QT-Element „Routequery“ genutzt werden. „Routequery“ benutzt die Openstreetmap-API um eine Route zu generieren. Danach kann die Route graphisch dargestellt werden.

Abb. 4. Navigation und Routenplanung

Kopf, Fußzeile und Customcontrolls

Die Kopfzeile enthält die aktuelle (lokale) Systemuhrzeit und einen Button mit dem man das Hauptmenü aufrufen kann.

Die Fußzeile enthält die im Musikspieler implementierten Buttons im Kleinformat. Außerdem ist es möglich den Song zu muten und die Lautstärke mit Hilfe eines Lautstärkeregler festzulegen. Drückt der Benutzer auf einen der Pfeile links oder rechts gelangt man zur Lüfter und Heizungssteuerung.

Abb. 5: Header und Folter

Lüfter und Heizungssteuerung

Die Stärke des Lüfters kann in drei Stufen durch „Plus“- und „Minus“-Tasten gewählt werden. Außerdem wird die Temperatur des Fahrzeuginnenraums (in Grad Celsius) dargestellt. Die Fahrer und Beifahrersitzheizung lässt sich an und ausschalten. Drückt man auf den Pfeil, der nach oben zeigt, wird ein erweitertes Kontextmenü geöffnet. Dort findet man Schaltflächen für die Front bzw. Heckheizung und Raumzirkulation

Abb. 6: Lüftersteuerung, Heizung und Temperatur

Hauptmenü / Auswahl der Szenen

Das Hauptmenü dient als Navigationselement. Die Applikation ist durch Hinzufügen weiterer Contentseiten erweiterbar. Man kann durch Wischgesten nach links bzw. rechts durch die Inhalte der Applikation „swipen“. Bis man jedoch an die richtige Stelle gelangt kann es unter Umständen recht lange dauern. Durch betätigen der Buttons gelangt der Nutzer schnell zum jeweiligen Inhalt.

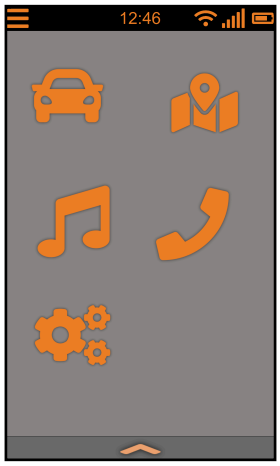

Abb. 7: Hauptmenü, Navigation durch die Applikation

Impressum

Da der Platz im Dashboard begrenzt ist, wurde im Infotainment das Impressum angelegt. Hier wird auf die verwendete Musik, Plugins und rechtlichen Informationen eingegangen.

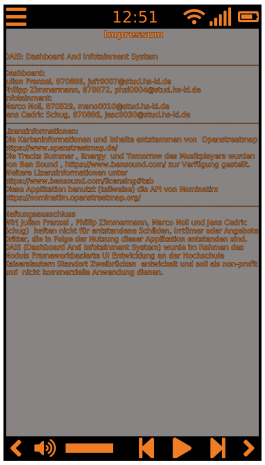

Abb. 8: Impressum