NFC Tagger

1.0

Erzeugt von Doxygen 1.8.11

Inhaltsverzeichnis

1 Ziel

Bis zur Abgabe des Projektes zum Ende des WS 2015/16 soll eine App programmiert werden die Kontakte aus Android oder von einer Eingabemaske einlesen und auf eine NFC Karte speichern, sowie von der Karte lesen und in den Android Kontakten speichern kann. Zusätzlich sollen die Kontaktinformationen direkt zwischen zwei NFC fähigen Geräten kontaktlos ausgetauscht werden können. Dafür soll eine übersichtliche Oberfläche mit maximal vier Buttons erstellt werden und eine Hilfe für den Nutzer zur Verfügung stehen [MANUAL].

2 Einleitung

Im Rahmen unseres Studiums der Wirtschaftsinformatik haben wir uns im 5. Semester für die Profilierung Mobile Applikationen entschieden. Hauptbestandteil und auch prüfungsrelevant für das Modul ist das eigenständige Planen und Implementieren einer Android App. Nach einer kurzen Einführung in die Android Programmierung durch Herrn Professor Schemmert lag es an den Studierenden sich für eines der vorgeschlagenen Projekte oder den Entwurf eines eigenen Projektes zu entscheiden. Wir haben uns dazu entschieden, eine von uns vorgeschlagene NFC App zu programmieren. Im Laufe des Semsters konnten wir Projektplanungs- und Software Engineering Kenntnisse aus den vergangenen Semstern praktisch anwenden und damit den planungsgemäßen Ablauf des Projektes sicherstellen. Trotz unterschiedlicher Vorkenntnisse in der Android Programmierung konnten wir uns alle gut in die Programmierung einbringen und jeder einzelne einen Teil beitragen. Im Folgenden soll unsere Herangehensweise, aufgetretene Schwierigkeiten und weitere Informationen aufgezeigt werden.

Motivation

Auf der Suche nach einer App mit praktischem Mehrwert und interessantem Inhalt haben wir uns für eine N↩ FC App entschieden. Nachdem die anfägliche Idee eines Klonens der HftL Card (Handy als Zugangskarte) sich bald als unrealistisch herausstellte, haben wir uns entschieden eine VisitenkartenApp für NFC Cards zu schreiben. Aufgrund der Preisentwicklung und umweltschonenden Eigenschaften von elektronischen Visitenkarten scheint ein "baldiges" Ersetzen der klassischen Visitenkarte aus Papier realistisch. Zudem werden Kontakte heute in der Regel digital gespeichert und verwaltet was ein analoges Austauschen von Daten redundant macht.

3 Anforderungen

Ausgehend von unserer Zielformulierung und der Motivation ergeben sich einige funktionale und nicht funktionale Anforderungen. Zu den Funktionalen gehören das Lesen und Schreiben von Kontakten in Android, sowie der NFC Karte. Auch ein kontaktloses Übertragungsprotokoll für Kontakte wird benötigt. Um die Akzeptanz der App und damit auch deren Verbreitung zu unterstützen ergeben sich auch Anfoderungen an die optische Gestaltung der App. Sie sollte möglichst übersichtlich (nicht überladen), intuitiv und adaptiv an die Gerätegröße gestaltet sein. Zusätzlich ist es uns wichtig, eine Dokumentation für den Nutzer zu erstellen die den nötigen Support auf ein Minimum reduziert und eine frustfreie Verwendung der App ermöglicht. [Hier brauchen wir vermutlich noch mehr...]

4 UML-Diagramme

Zur besseren Übersicht wurden zu Beginn des Projektes mehrere Diagramme nach UML-Standard 2.0 erstellt.

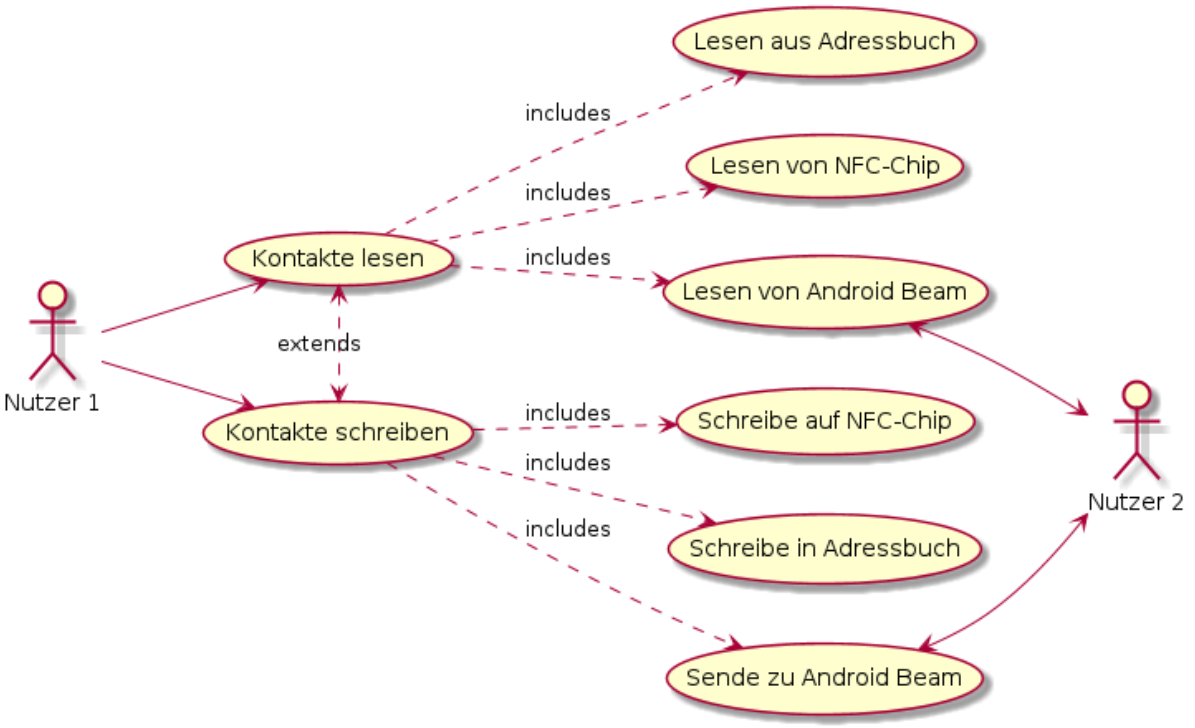

Abbildung 1 Usecasediagramm

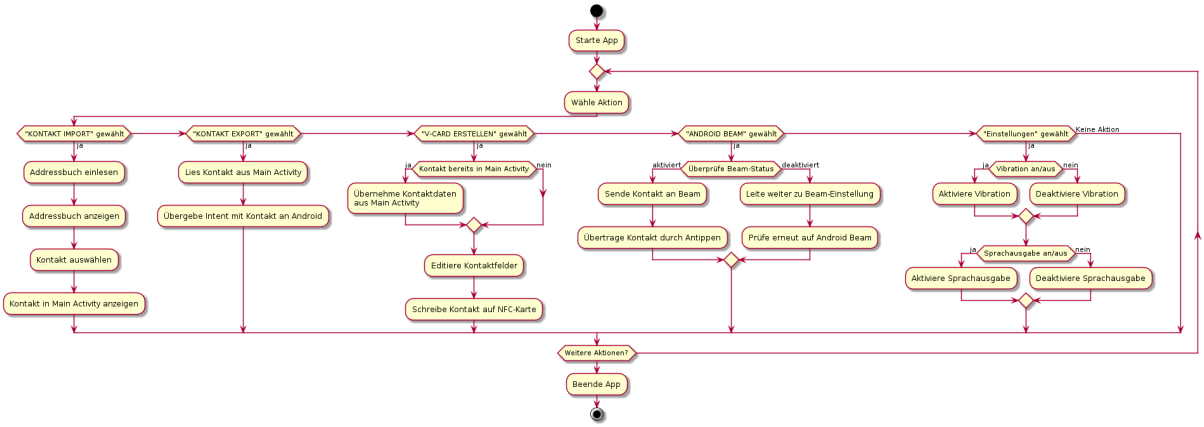

Abbildung 2 Aktivitätsdiagramm

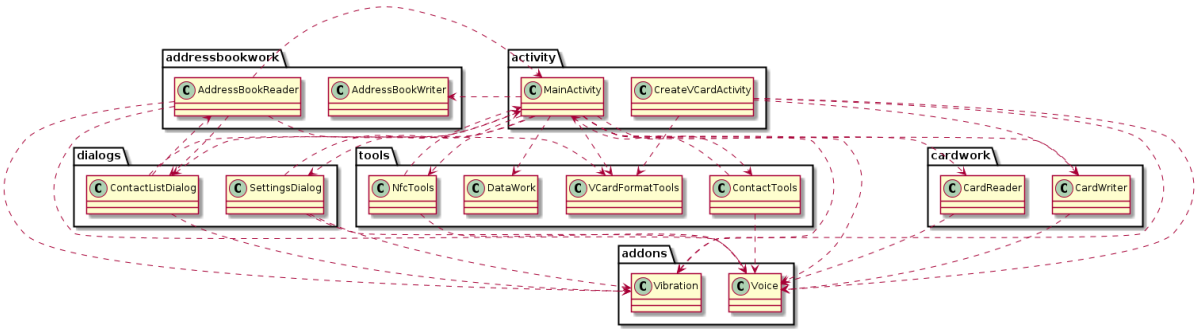

Abbildung 3 Klassendiagramm

5 Hierarchie-Verzeichnis

5.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

6 Klassen-Verzeichnis

6.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

7 Klassen-Dokumentation

7.1 com.ag.mk.nfccardreadwrite.addressbookwork.AddressBookReader Klassenreferenz

Öffentliche Methoden

- **AddressBookReader** (MainActivity mainActivity)
- void getAdressbookData (int position)
- void readAllContacts (ContentResolver contentResolver)
- ArrayList< String > **getListItems** ()

7.1.1 Ausführliche Beschreibung

Diese Klasse beinhaltet alle Methoden zum Auslesen eines Kontaktes aus dem Adressbuch.

Autor

Klaus Steinhauer, Marko Klepatz

7.1.2 Dokumentation der Elementfunktionen

7.1.2.1 void com.ag.mk.nfccardreadwrite.addressbookwork.AddressBookReader.getAdressbookData (int *position*)

Diese Methode sammelt beim Aufruf alle benötigten Daten zum ausgewählten Kontakt.

Parameter

position	übergibt die Position aus der Namen-ListView zum lokalisieren des Kontaktes im Adressbuch

7.1.2.2 void com.ag.mk.nfccardreadwrite.addressbookwork.AddressBookReader.readAllContacts (ContentResolver *contentResolver*)

Diese Methode liest alle Kontakte im Adressbuch aus und schreibt die Namen in die statische **listItems** Liste aus der **ContactListDialog** Klasse.

Parameter

contentResolver	

Siehe auch

ContactListDialog

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- addressbookwork/AddressBookReader.java

7.2 com.ag.mk.nfccardreadwrite.addressbookwork.AddressBookWriter Klassenreferenz

Öffentliche, statische Methoden

- static void writeContact (final Context context, final ArrayList< String > cardContent)

7.2.1 Ausführliche Beschreibung

Diese Klasse beinhaltet die Methode zum Schreiben von Address-Daten in das Adressbuch.

Autor

Klaus Steinhauer, Marko Klepatz

7.2.2 Dokumentation der Elementfunktionen

7.2.2.1 static void com.ag.mk.nfccardreadwrite.addressbookwork.AddressBookWriter.writeContact (final Context *context,* final ArrayList< String > *cardContent*) [static]

Diese Methode generiert einen Intent welcher alle zu schreibenden Address-Daten enthält und ruft im Anschluss das Adressbuch über diesen Intent auf und übergibt die Daten.

Parameter

context	übergibt den Context der rufenden Activity
cardContent	übergibt die zu schreibenden Daten in Form einer Array List

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- addressbookwork/AddressBookWriter.java

7.3 com.ag.mk.nfccardreadwrite.cardwork.CardReader Klassenreferenz

Öffentliche, statische Methoden

- static String readTag (Intent intent)

7.3.1 Ausführliche Beschreibung

Diese Klasse beinhaltet die Methode zum Auslesen der Kartendaten.

Autor

Oliver Friedrich, Marko Klepatz

7.3.2 Dokumentation der Elementfunktionen

7.3.2.1 static String com.ag.mk.nfccardreadwrite.cardwork.CardReader.readTag (Intent *intent*) [static]

Diese Methode liest aus dem übergebenem Intent die Daten aus, die sich auf der Karte befinden und wandelt sie in einen String um.

Parameter

intent	enthält die Daten die auf der Karte sind

Rückgabe

gibt den String mit den Daten auf der Karte zurück

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- cardwork/CardReader.java

7.4 com.ag.mk.nfccardreadwrite.cardwork.CardWriter Klassenreferenz

Öffentliche Methoden

- **CardWriter** (Context context)
- void writeNdefMessage (Tag tag, NdefMessage ndefMessage)
- NdefMessage createNdefMessage (String content)

7.4.1 Ausführliche Beschreibung

Diese Klasse beinhaltet die Methoden die für das Beschreiben eines NFC Chips benötigt werden.

Autor

Oliver Friedrich, Marko Klepatz

7.4.2 Dokumentation der Elementfunktionen

7.4.2.1 NdefMessage com.ag.mk.nfccardreadwrite.cardwork.CardWriter.createNdefMessage (String *content*)

Diese Methode generiert die NDEF Message mit dem speziellen Mime Type für diese App.

Parameter

content	übergibt den Inhalt der auf die Karte geschrieben werden soll

Rückgabe

gibt die generierte NDEF Message zurück

7.4.2.2 void com.ag.mk.nfccardreadwrite.cardwork.CardWriter.writeNdefMessage (Tag *tag,* NdefMessage *ndefMessage*)

Diese Methode schreibt die NDEF Nachricht auf den NFC Chip.

Parameter

tag	übergibt das Tag-Format
ndefMessage	übergibt die NDEF Nachricht zum beschreiben auf den NFC Chip

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- cardwork/CardWriter.java

7.5 com.ag.mk.nfccardreadwrite.dialogs.ContactListDialog Klassenreferenz

Öffentliche Methoden

- ContactListDialog (MainActivity mainActivity)
- void showDialog ()

7.5.1 Ausführliche Beschreibung

Diese Klasse beinhaltet die Methoden zum Initialisieren und Anzeigen des Kontakt-Listen-Dialogs.

Autor

Marko Klepatz

7.5.2 Beschreibung der Konstruktoren und Destruktoren

7.5.2.1 com.ag.mk.nfccardreadwrite.dialogs.ContactListDialog.ContactListDialog (MainActivity *mainActivity*)

Dieser Konstruktor leitet alle Initialisierungen für den ContactListDialog ein.

Parameter

mainActivity	übergibt die Klasse MainActivity für den Context der GUI Elemente

Siehe auch

MainActivity
AddressBookReader

7.5.3 Dokumentation der Elementfunktionen

7.5.3.1 void com.ag.mk.nfccardreadwrite.dialogs.ContactListDialog.showDialog ( )

Diese Methode leitet das Auslesen der Kontakte ein und ruft die Anzeigemethode für den ContactListDialog.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- dialogs/ContactListDialog.java

7.6 com.ag.mk.nfccardreadwrite.tools.ContactTools Klassenreferenz

Öffentliche Methoden

- ContactTools (MainActivity mainActivity)
- void mailContact (final String email)
- void **easterEgg** (String name)
- void callContact (String name, final String number)

7.6.1 Ausführliche Beschreibung

Diese Klasse beinhaltet verschiedene Methoden um die Kontaktdaten direkt an andere Anwendungen im Gerät zu übergeben und zu starten.

Autor

Marko Klepatz

7.6.2 Beschreibung der Konstruktoren und Destruktoren

7.6.2.1 com.ag.mk.nfccardreadwrite.tools.ContactTools.ContactTools (MainActivity *mainActivity*)

Parameter

mainActivity	übergibt die MainActivity für den Context zum Ausführen von Code der nur auf dieser ausgeführt werden kann

Siehe auch

MainActivity

7.6.3 Dokumentation der Elementfunktionen

7.6.3.1 void com.ag.mk.nfccardreadwrite.tools.ContactTools.callContact (String *name,* final String *number*)

Diese Methode generiert einen Intent welcher die Nummer des Kontaktes übergibt und ruft mit diesem direkt das Programm zum Anrufen von Kontakten auf und ruft diesen an.

Parameter

name	übergibt den Namen des ausgewählten Kontakts für die Sprachausgabe
number	übergibt die Nummer des ausgewählten Kontakts zum Anrufen

7.6.3.2 void com.ag.mk.nfccardreadwrite.tools.ContactTools.mailContact (final String *email*)

Diese Methode generiert einen Intent der die E-Mail Adresse des aktuell ausgewählten Kontakts übergibt und startet eine Auswahl mit allen potenziell einsetzbaren E-Mail Programmen auf dem Gerät.

Parameter

email	übergibt die E-Mail Adresse des aktuell ausgewählten Kontakts

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- tools/ContactTools.java

7.7 com.ag.mk.nfccardreadwrite.activity.CreateVCardActivity Klassenreferenz

Abgeleitet von AppCompatActivity.

Geschützte Methoden

- void onCreate (Bundle savedInstanceState)
- void onNewIntent (Intent intent)
- void onResume ()
- void onPause ()

7.7.1 Ausführliche Beschreibung

Diese Activity beinhaltet die Logik zu den GUI Elementen, die benötigt werden um NFC Medien zu beschreiben.

Zusätzlich beinhaltet sie eine Methode zum Einlesen gültiger NFC Medien.

Autor

Marko Klepatz, Oliver Friedrich

7.7.2 Dokumentation der Elementfunktionen

7.7.2.1 void com.ag.mk.nfccardreadwrite.activity.CreateVCardActivity.onCreate (Bundle *savedInstanceState*) [protected]

Diese Methode setzt alle relevanten Eigenschaften für die GUI und leitet die Initialisierungen aller Objekte ein.

Zusätzlich nimmt sie den aufrufenden Intent von der MainActivty entgegen und prüft ob dieser Daten zum Beschreiben auf ein NFC Medium enthält und leitet diese weiter, an die GUI Objekte.

Parameter

savedInstanceState	

Siehe auch

NfcAdapter
CardWriter

7.7.2.2 void com.ag.mk.nfccardreadwrite.activity.CreateVCardActivity.onNewIntent (Intent *intent*) [protected]

Diese Methode empfängt einen von dem Manifest gefilterten Intent, welcher nur dann in dieser ankommt wenn das NFC Medium, was an das Gerät gehalten wird, mit den gültigen Technologien ausgestattet ist.

Die aktuell gültigen Technologien sind in folgenden Dateien einsehbar:
AndroidManifest.xml
tech.xml

Parameter

intent	

7.7.2.3 void com.ag.mk.nfccardreadwrite.activity.CreateVCardActivity.onPause ( ) [protected]

Diese Methode deaktiviert den NFC Adapter.

7.7.2.4 void com.ag.mk.nfccardreadwrite.activity.CreateVCardActivity.onResume ( ) [protected]

Diese Methode konfiguriert und aktiviert bei Aufruf den NFC Adapter.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- activity/CreateVCardActivity.java

7.8 com.ag.mk.nfccardreadwrite.tools.DataWork Klassenreferenz

Öffentliche Methoden

- **DataWork** (Context context)

Öffentliche, statische Methoden

- static String readSingleLineFile (String dataName)
- static List< String > readMultiLineFile (String dataName)
- static void writeSingleLineFile (String dataName, String information)
- static void writeMultiLineFile (String dataName, List< String > informationList)

7.8.1 Ausführliche Beschreibung

Diese Klasse beinhaltet die Methoden zum Lesen und Schreiben von Dateien für die Anwendung.

Autor

Marko Klepatz

7.8.2 Dokumentation der Elementfunktionen

7.8.2.1 static List<String> com.ag.mk.nfccardreadwrite.tools.DataWork.readMultiLineFile (String *dataName*) [static]

Diese Methode liest eine Datei mit mehreren Zeilen aus und gibt den Inhalt on Form einer Liste vom Typ String zurück.

Parameter

dataName	übergibt den Namen der zu lesenden Datei

Rückgabe

values gibt die Liste mit dem Inhalt jeder Zeile zurück

7.8.2.2 static String com.ag.mk.nfccardreadwrite.tools.DataWork.readSingleLineFile (String *dataName*) [static]

Diese Methode liest eine Datei mit einer Zeile aus und gibt den Inhalt on Form eines Strings zurück.

Parameter

dataName	übergibt den Namen der zu lesenden Datei

Rückgabe

gibt den ausgelesenen String zurück

7.8.2.3 static void com.ag.mk.nfccardreadwrite.tools.DataWork.writeMultiLineFile (String *dataName,* List< String > *informationList*) [static]

Diese Methode schreibt eine mehrzeilige Information in eine Datei mit Umbrüchen

Parameter

dataName	übergibt den Namen der zu schreibenden Datei
informationList	übergibt die zu schreibenden Informationen

7.8.2.4 static void com.ag.mk.nfccardreadwrite.tools.DataWork.writeSingleLineFile (String *dataName,* String *information*) [static]

Diese Methode schreibt eine einzeilige Information in eine Datei ohne Umbrüche

Parameter

dataName	übergibt den Namen der zu schreibenden Datei
information	übergibt die zu schreibende Information

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- tools/DataWork.java

7.9 com.ag.mk.nfccardreadwrite.activity.MainActivity Klassenreferenz

Abgeleitet von AppCompatActivity, CreateNdefMessageCallback und OnInitListener.

Öffentliche Methoden

- void setVCardInformationOnMainScreen (String vCardInformation)
- NdefMessage createNdefMessage (NfcEvent event)
- void onInit (int status)
- boolean **onCreateOptionsMenu** (Menu menu)
- boolean **onOptionsItemSelected** (MenuItem item)

Statische öffentliche Attribute

- static final String **TAG** = "Nfc Card App"

Geschützte Methoden

- void onCreate (Bundle savedInstanceState)
- void onNewIntent (Intent intent)
- void onResume ()
- void onPause ()
- void onDestroy ()

7.9.1 Ausführliche Beschreibung

Diese Activity beinhaltet die Logik zu den GUI Elementen die benötigt werden um NFC Medien zu lesen und anzuzeigen.
Zusätzlich werden hier das Anzeigen von Kontakten aus dem Adressbuch und das Importieren von neuen Kontakten, sowie die Android Beam-Funktion, eingeleitet. Ebenso wird hier auch die CreateVCardActivity gerufen.

Weiterführend wird immer diese Activity gerufen, wenn eine für diese App gültige NFC Technologie oder der spezielle Mime Type dieser App erkannt wird. (Der Mime Type ist in der Klasse **CardWriter** einsehbar)

Siehe auch

CardWriter

Autor

Marko Klepatz, Oliver Friedrich

7.9.2 Dokumentation der Elementfunktionen

7.9.2.1 NdefMessage com.ag.mk.nfccardreadwrite.activity.MainActivity.createNdefMessage (NfcEvent *event*)

Diese Callback Methode gibt die zu sendende NDEF Message für den Android NFC Beamer zurück.

Parameter

event	

Rückgabe

gibt die zu sendende NDEF Message für den Android NFC Beamer zurück.

7.9.2.2 void com.ag.mk.nfccardreadwrite.activity.MainActivity.onCreate (Bundle *savedInstanceState*) [protected]

Diese Methode setzt alle relevanten Eigenschaften für die GUI und leitet die Initialisierungen aller Objekte ein.

Zusätzlich nimmt diese Methode einen von dem Manifest gefilterten Intent entgegen welcher nur dann in dieser ankommt wenn das NFC Medium, was an das Gerät gehalten wird, mit den gültigen Technologien ausgestattet ist

bzw. den Mime Type für diese App besitzt.

Die aktuell gültigen Technologien sind in folgenden Dateien einsehbar:
AndroidManifest.xml
tech.xml

Parameter

savedInstanceState	

7.9.2.3 void com.ag.mk.nfccardreadwrite.activity.MainActivity.onDestroy ( ) [protected]

Diese Methode beendet beim Aufruf die TextToSpeech Vorgang/Funktion.

7.9.2.4 void com.ag.mk.nfccardreadwrite.activity.MainActivity.onInit (int *status*)

Diese Methode weist der TextToSpeech Klasse die Sprache zu mit welcher dann die Sprachausgabe erfolgt. In diesem Fall wird immer die Standardsprache des Geräts verwendet.

Parameter

status	

7.9.2.5 void com.ag.mk.nfccardreadwrite.activity.MainActivity.onNewIntent (Intent *intent*) [protected]

Diese Methode empfängt einen von dem Manifest gefilterten Intent, welcher nur dann in dieser ankommt wenn das NFC Medium, was an das Gerät gehalten wird, mit den gültigen Technologien ausgestattet ist.

Die aktuell gültigen Technologien sind in folgenden Dateien einsehbar:
AndroidManifest.xml
tech.xml

Parameter

intent	

7.9.2.6 void com.ag.mk.nfccardreadwrite.activity.MainActivity.onPause ( ) [protected]

Diese Methode deaktiviert den NFC Adapter.

7.9.2.7 void com.ag.mk.nfccardreadwrite.activity.MainActivity.onResume ( ) [protected]

Diese Methode konfiguriert und aktiviert bei Aufruf den NFC Adapter.

7.9.2.8 void com.ag.mk.nfccardreadwrite.activity.MainActivity.setVCardInformationOnMainScreen (String *vCardInformation*)

Diese Methode lädt die Daten auf die **vCardListView**.

Parameter

vCardInformation	übergibt die Informationen die sich auf dem NFC Medium befinden.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- activity/MainActivity.java

7.10 com.ag.mk.nfccardreadwrite.tools.NfcTools Klassenreferenz

Öffentliche Methoden

- NfcTools (MainActivity mainActivity, NfcAdapter nfcAdapter)
- void checkNFC ()
- void checkNFCSupport ()

7.10.1 Ausführliche Beschreibung

Diese Klasse beinhaltet die Methoden zum Überprüfen ob die NFC Technologie vorhanden und aktiviert ist.

Autor

Marko Klepatz

7.10.2 Beschreibung der Konstruktoren und Destruktoren

7.10.2.1 com.ag.mk.nfccardreadwrite.tools.NfcTools.NfcTools (MainActivity *mainActivity,* NfcAdapter *nfcAdapter*)

Parameter

mainActivity	übergibt die MainActivity für den Context
nfcAdapter	übergibt den NfcAdapter zur Überprüfung

Siehe auch

MainActivity
NfcAdapter

7.10.3 Dokumentation der Elementfunktionen

7.10.3.1 void com.ag.mk.nfccardreadwrite.tools.NfcTools.checkNFC ( )

Diese Methode überprüft ob die NFC Technologie aktiviert ist und gibt eine Warnung in Form einer Toast Nachricht aus wenn sie deaktiviert ist.

7.10.3.2 void com.ag.mk.nfccardreadwrite.tools.NfcTools.checkNFCSupport ( )

Diese Methode überprüft ob die NFC Technologie auf dem Gerät vorhanden ist und gibt eine Toast Nachricht aus wenn sie es nicht ist und beendet die App.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- tools/NfcTools.java

7.11 com.ag.mk.nfccardreadwrite.dialogs.SettingsDialog Klassenreferenz

Öffentliche Methoden

- SettingsDialog (MainActivity mainActivity)
- void **showDialog** ()
- Switch **getVibrationSwitch** ()
- Switch **getVoiceSwitch** ()

7.11.1 Ausführliche Beschreibung

Diese Klasse beinhaltet alle Methoden zum Generieren und Anzeigen des Einstellungs-Dialogs. Weiterführend werden hier auch alle Einstellungen direkt in Dateien auf dem Handy geschrieben.

Siehe auch

DataWork

Autor

Marko Klepatz

7.11.2 Beschreibung der Konstruktoren und Destruktoren

7.11.2.1 com.ag.mk.nfccardreadwrite.dialogs.SettingsDialog.SettingsDialog (**MainActivity** *mainActivity*)

Dieser Konstruktor leitet alle Initialisierungen für den SettingsDialog ein.

Parameter

mainActivity	übergibt die Klasse MainActivity für den Context der GUI Elemente

Siehe auch

MainActivity

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- dialogs/SettingsDialog.java

7.12 com.ag.mk.nfccardreadwrite.tools.VCardFormatTools Klassenreferenz

Öffentliche, statische Methoden

- static ArrayList< String > extractCardInformation (String[ ] cardContent)
- static String getFormatedVCardString (String userName, String mobileNumber, String homeNumber, String eMail)

7.12.1 Ausführliche Beschreibung

Diese Klasse beinhaltet die Methoden, die für die V-Card Formatierung und das Extrahieren der Informationen aus dem V-Card Format benötigt werden.

Autor

Marko Klepatz, Klaus Steinhauer

7.12.2 Dokumentation der Elementfunktionen

7.12.2.1 static ArrayList<String> com.ag.mk.nfccardreadwrite.tools.VCardFormatTools.extractCardInformation (String[ ] *cardContent*) [static]

Diese Methode extrahiert die Informationen aus einem V-Card String und bringt sie in ein für diese App sinnvolles Ausgabeformat.

Parameter

cardContent	übergibt die rohen Karten-Daten

Rückgabe

vCardInformationList gibt eine Liste mit allen extrahierten Daten zurück

7.12.2.2 static String com.ag.mk.nfccardreadwrite.tools.VCardFormatTools.getFormatedVCardString (String *userName,* String *mobileNumber,* String *homeNumber,* String *eMail*) [static]

Diese Methode formatiert die Roh-Adress-Daten in einen V-Card-Format String.

Parameter

userName	übergibt den Namen des gewählten Kontaktes
mobileNumber	übergibt die Mobilnummer des gewählten Kontaktes
homeNumber	übergibt die Festnetznummer des gewählten Kontaktes
eMail	übergibt die E-Mail Adresse des gewählten Kontaktes

Rückgabe

gibt den V-Card-Format String zurück

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- tools/VCardFormatTools.java

7.13 com.ag.mk.nfccardreadwrite.addons.Vibration Klassenreferenz

Öffentliche Methoden

- Vibration (Vibrator vibrator)

Öffentliche, statische Methoden

- static void **setVibration** (boolean vibration)
- static boolean **isVibration** ()
- static void vibrate ()

7.13.1 Ausführliche Beschreibung

Diese Klasse ist für die Vibrationsfunktion, insofern aktiviert, für alle Klassen und Activities zuständig.

Sie muss einmal über ihren Konstruktor initialisiert werden und ist dann für alle Klassen, durch ihren statischen Charakter verfügbar.

Vibrator ist die zu initialisierende Klasse mit der die Vibrationsfunktion ausgeführt werden kann.

Autor

Marko Klepatz

7.13.2 Beschreibung der Konstruktoren und Destruktoren

7.13.2.1 com.ag.mk.nfccardreadwrite.addons.Vibration.Vibration (Vibrator *vibrator*)

Dieser Konstruktor nimmt die in der MainActivity initialisierte Vibrator Klasse entgegen, die hier benötigt wird um die Vibration auszuführen.

Parameter

vibrator	übergibt die benötigte Vibrator Klasse zum Ausführen der Vibration

7.13.3 Dokumentation der Elementfunktionen

7.13.3.1 static void com.ag.mk.nfccardreadwrite.addons.Vibration.vibrate ( ) [static]

Diese Methode ist beim Aufruf dafür zuständig, dass das Gerät 25 Millisekunden vibriert.

Zusätzlich wird geprüft ob die Variable **vibration** auf true oder false gestellt ist.

true: vibriert
false: vibriert nicht

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- addons/Vibration.java

7.14 com.ag.mk.nfccardreadwrite.addons.Voice Klassenreferenz

Öffentliche Methoden

- Voice (TextToSpeech textToSpeech)

Öffentliche, statische Methoden

- static void speakOut (String message)
- static boolean **isSound** ()
- static void **setSound** (boolean sound)

7.14.1 Ausführliche Beschreibung

Diese Klasse ist für die Sprachausgabe, insofern aktiviert, für alle Klassen und Activities zuständig.

Sie muss einmal über ihren Konstruktor initialisiert werden und ist dann für alle Klassen, durch ihren statischen Charakter verfügbar.

Vibrator ist die zu initialisierende Klasse mit der die Vibrationsfunktion ausgeführt werden kann.

Autor

Marko Klepatz

7.14.2 Beschreibung der Konstruktoren und Destruktoren

7.14.2.1 com.ag.mk.nfccardreadwrite.addons.Voice.Voice (TextToSpeech *textToSpeech*)

Dieser Konstruktor nimmt die in der MainActivity initialisierte TextToSpeech Klasse entgegen, die hier benötigt wird um die Sprachausgabe auszuführen.

Parameter

textToSpeech	übergibt die benötigte TextToSpeech Klasse zum Ausführen der Sprachausgabe

7.14.3 Dokumentation der Elementfunktionen

7.14.3.1 static void com.ag.mk.nfccardreadwrite.addons.Voice.speakOut (String *message*) [static]

Diese Methode ist beim Aufruf für die Sprachausgabe zuständig.

Zusätzlich wird geprüft ob die Variable **sound** auf true oder false gestellt ist.

true: Sprachausgabe wird ausgeführt
false: Sprachausgabe wird nicht ausgeführt

Parameter

message	übergibt die Nachricht, welche über die Sprachausgabe ausgegeben wird

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- addons/Voice.java

Index