Dokumentation

Seminararbeit zu der Vorlesung “Entwicklung mobiler Applikationen“

Tabea Kiupel, Marco Meyer
13.11.2014

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

EINLEITUNG

Folgendes Dokument stellt die Dokumentation zu der Entwicklung einer mobilen Applikation für ein Android-System dar.

Zu Anfang werden die Anforderungen an die zu erstellende Applikation aufgelistet. Daraufhin wird ein kurzer Überblick über die Klassen gegeben, welche im Rahmen der Implementierungen entstanden sind. Nähere Erläuterungen zu den einzelnen Klassen folgen im nachfolgenden Abschnitt. In diesem wird der Quellcode der einzelnen Klassen erläutert. Neben den Klassen werden zusätzlich kurz die Änderungen an der Manifest-Datei sowie an den Style-Themen dargestellt.

ANFORDERUNGEN AN DIE APPLIKATION

Die erstellte Applikation soll über ein Menü Bilder aufnehmen können. Diese Bilder sollen in einem eigenen Verzeichnis mit Informationen über den Aufnahmezeitpunkt und die Position zum Zeitpunkt der Aufnahme gespeichert werden. Die aufgenommenen Bilder sollen in einer Liste mit Bildnamen und Datum er Aufnahme angezeigt werden. Wählt man eines der aufgelisteten Elemente aus, soll eine Activity gestartet werden, welche das aufgenommene Bild anzeigt.
Dort soll es möglich sein, eine Karte anzeigen, welche anzeigt, an welcher Stelle das gespeicherte Bild erstellt wurde. Ebenso soll das Versenden des Fotos per E-Mail und das Speichern des Fotos in die Fotogallerie ermöglicht werden.

KLASSEN-ÜBERSICHT

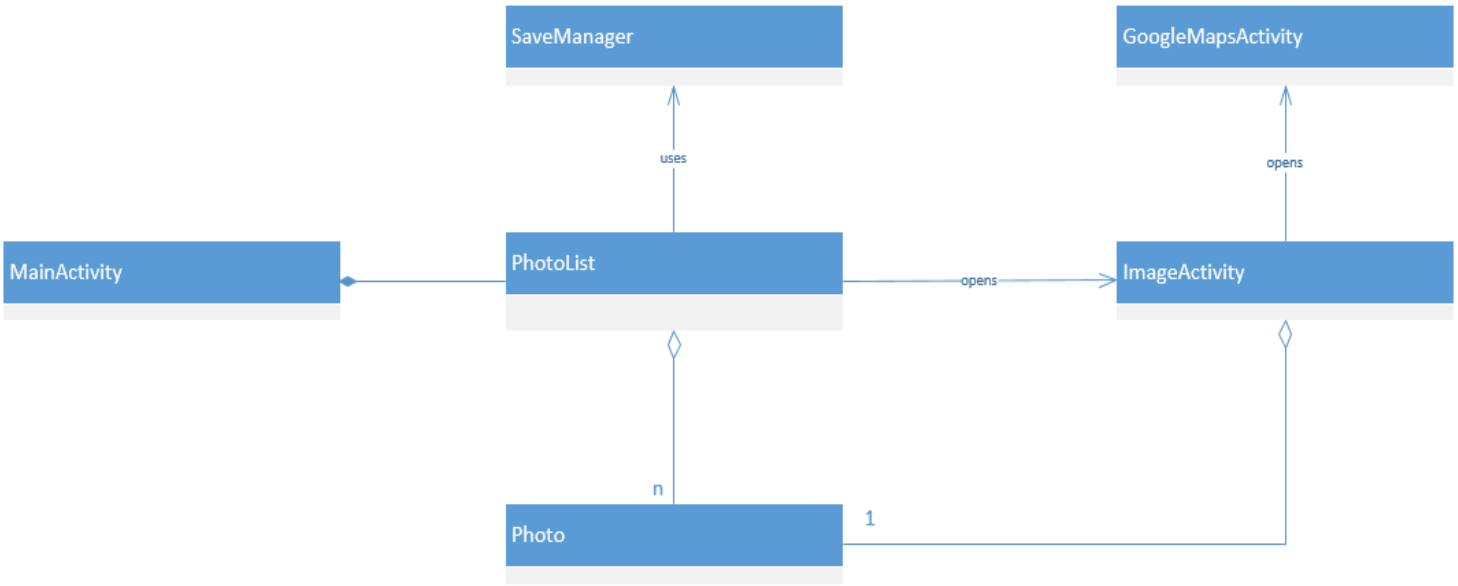

Abbildung 1: UML-Klassendiagramm

ERLÄUTERUNGEN DES QUELLCODES

Es folgen die Erläuterungen zu den Implementierungen der einzelnen Kassen. Dabei werden neben den einzelnen Attributen die modifizierten Funktionen betrachtet. Zusätzlich werden bei den Aktivitäten das Layout und Menü näher erläutert. Ebenso werden dort die Ergebnisse der Implementierung visuell dargestellt.

DIE KLASSE MAINACTIVITY

Die Hauptaktivität der Applikation, die beim Start ausgeführt wird. Sie lässt den Nutzer bestimmte Optionen bezüglich der Bilder, welche von der Applikation verwaltet werden, ausführen. Ebenso kommt der Nutzer über mehrere Rückschritte zu dieser Activity zurück.

LAYOUT: ACTIVITY_MAIN.XML

Das Erscheinungsbild dieser Activity besteht aus einem LinearLayout. Das LinearLayout schließt die ganze Benutzeroberfläche ein. In diesem Layout ist ein ListView-Objekt *photo_list* eingebettet. Während die *photo_list* sich einerseits über die komplette Breite der Oberfläche erstreckt passt sie anderseits ihre Höhe, je nach ihrem Inhalt an.

MENÜ: MAIN.XML

Das Menü der MainActivity spezifiziert zwei Elemente: Beim ersten Element handelt es sich um den Menüpunkt “*takePhoto*”, welcher mit einem Icon in der ActionBar angezeigt. Allerdings wird das Icon nur angezeigt,wenn genug Platz in der ActionBar vorhanden ist. Ist nicht genug Platz vorhanden, wird der Menüpunkt in den Überfluss geschoben. Das Gleiche gilt für das zweite Element “*deletePhotos*”, welches wiederum mit einem eigenem Icon visualisiert wird.

ÜBERSICHT

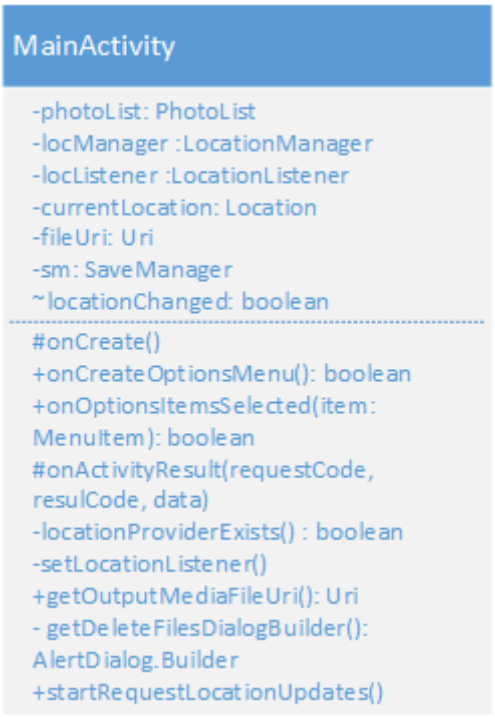

Abbildung 2: Klasse MainActivity

ATTRIBUTE

PhotoList photoList

Eine Liste aus Objekten des Typs Photos. Siehe Klasse PhotoList.

LocationManager locManager

Der Location-Manager, der für die Beschaffung der notwendigen GPS-Daten zuständig ist.

LocationListener locListener

Der Location-Listener, der auf die Änderung der GPS-Koordinaten achtet und diese an die Klasse weitergibt.

Location currentLocation

Die aktuell gemessenen GPS-Koordinaten.

SaveManager sm

Instanz der SaveManager Klasse. Siehe Klasse SaveManager.

METHODEN

onCreate(Bundle savedInstanceState)

Bei der Erstellung der Hauptaktivität werden die notwendigen Klassenattribute instanziiert, die Bildinformationen werden aus den XML-Dokumenten geladen und die GPS-Erkennung wird eingerichtet.

onOptionsItemSelected(Menu item)

Diese Methode startet die Kamera-Aktivität, sofern auf den Menüeintrag „*takePhoto*“ geklickt wird. Startet die Kamera-Aktivität, wird ebenso die Erkennung der GPS-Daten gestartet.
Falls auf „*deletePhotos“* geklickt wird, erscheint eine Abfrage ob wirklich alle Fotos gelöscht werden sollen.

onActivityResult(int requestCode, int resultCode, Intent data)

Diese Methode wird nachdem der Nutzer die Kamera-Aktivität genutzt hat aufgerufen. Zunächst wird die Erkennung der GPS-Daten beendet und es wird geprüft, ob der Nutzer ein Foto geschossen hat. Ist dies der Fall erstellt es ein neues Objekt vom Typ Photo, siehe Klasse Photo, mit den aktuellen Metadaten. Dieses Foto wird der Liste der Fotos hinzugefügt.

locationProviderExists()

Diese Funktion prüft, ob der GPS-Provider existiert und gibt bei seiner Existenz entsprechend wahr zurück.

setLocationListener()

Diese Methode instanziiert den LocationListener. Sobald das erste Mal eine Position erkannt wird, wird dies dem Anwender zurückgemeldet.

getOutputMediaFileUri()

Gibt die Uri zurück, die den Pfad für ein mögliches neues Foto enthält.

getDeleteFilesDialogBuilder()

Ein AlertDialog.Builder wird für den Menüeintrag „*deletePhotos*“ erzeugt. Der Builder erhält einen Titel, eine Nachricht, einen PositivButton und einen NegativButton. Der PositivButton setzt den Löschvorgang fort und alle Bilder werden mitsamt ihren Metadaten auf dem Gerät

gelöscht. Benutzt der Anwender den NegativButttton, so wird der Löschvorgang abgebrochen. Der Anwender kehrt zur Übersicht der MainActivity zurück.

startRequestLocationUpdates()

Prüft hier zunächst auf den GPS-Provider und startet die Bestimmung von Positionsdaten. Er nutzt bei dessen Abwesenheit jedoch das Netzwerk um die Positionsdaten zu bestimmen.

ERGEBNIS

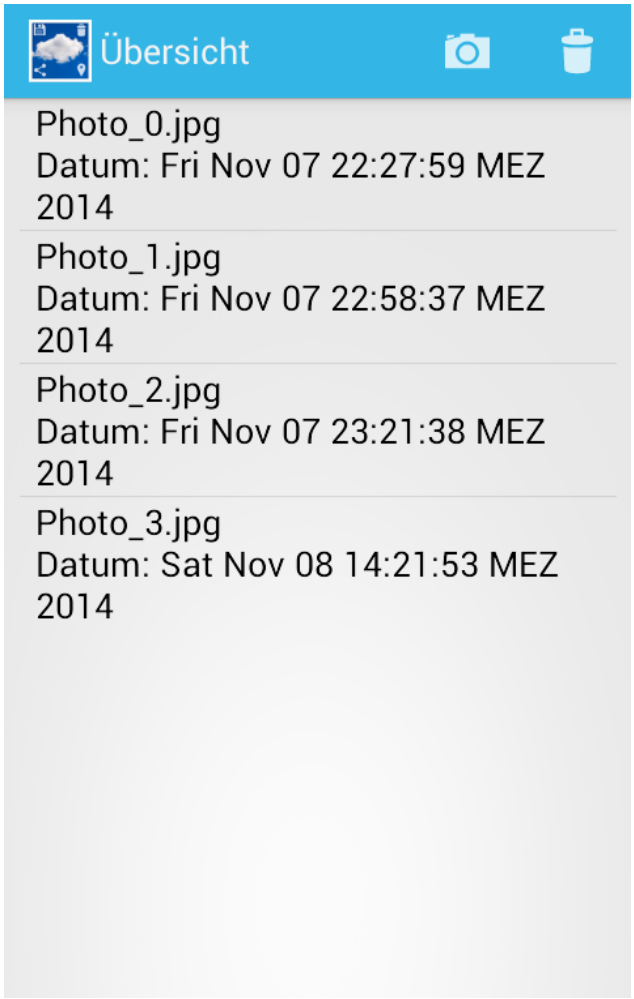

Abbildung 3: Auflistung von Elementen in der MainActivity

Abbildung 4: Dialog für das Löschen von allen Bidern

DIE KLASSE IMAGEACTIVITY

Die ImageActivity ist die Aktivität der Applikation, welche von der MainActivity aus geöffnet werden kann. Dabei erhält sie die zugehörigen Bildinformationen und zeigt ein gemachtes Foto an. Mit diesem Bild können bestimmte Operationen ausgeführt werden.

LAYOUT: ACTIVITY-IMAGE.XML

Für das Erscheinungsbilder der Aktivität wird nur ein ImageView-Objekt verwendet. Dieses füllt die gesamt Benutzeroberfläche aus.

MENÜ: IMAGE.XML

In dem Menü dieser Aktivität befinden sich die Elemente, welche es dem Nutzer ermöglichen, Operationen auf dem Bild auszuführen.
Das erste Element „*showOnMap*" wird genau wie die zwei weiteren Elemente „*saveInLibrary*" und „*action_share*", nur dann in der ActionBar mit ihrem Icon angezeigt, wenn genug Platz in dieser vorhanden ist. Dabei muss dem Element „*action_share*" eine besondere Beachtung geschenkt werden. Dieses Element wird zusätzlich mit der actionProviderClass „android.widget.ShareActionProvider" in Beziehung gesetzt.

ÜBERSICHT

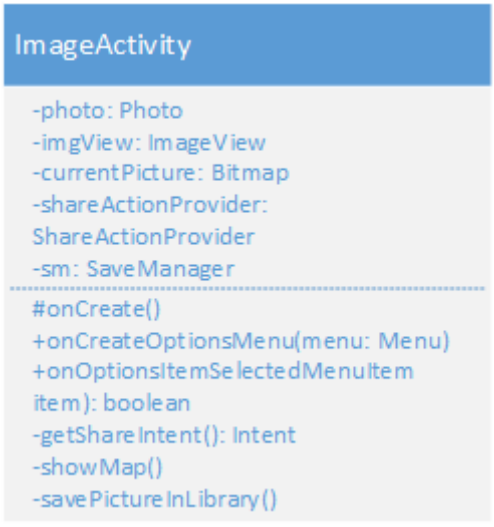

Abbildung 5: Klasse ImageActivity

ATTRIBUTE

Photo photo
Das Photo-Objekt, welches der Aktivität übergeben wird.

ImageView imgView
Das ImageView-Objekt, welches in der activity-image.xml definiert wurde.

Bitmap currentPicture
Das Foto, welches vom Anwender in der Kamera-Aktivität erstellt wurde.

ShareActionProvider shareActionProvider
Das Objekt vom Typ ShareActionProvider, welches zum Teilen des Bildes verwendet wird.

SaveManager sm
Instanz der Klasse SaveManager

METHODEN

onCreate(Bundle savedInstanceState)
Zum Beginn der Lebenszeit der Aktivität ImageActivity wird neben einem neuen SaveManager, das Klassenattribut *photo* instanziiert. Das Photo-Objekt bekommt die von der PhotoList übergebenen Bildinformationen und das aktuelle Bild *currentPicture* wird instanziiert. Dieses Bild wird an die ImageView *imgView* übergeben.

onCreateOptionsMenu(Menu menu)
Das zuvor spezifizierte Menu der ImageActivity wird geladen.
Weiterhin wird dem Element „*action_share*“ der *shareActionProvider* zugewiesen. Weiterhin wird dem *shareActionProvider* sein Inhalt, welcher geteilt werden soll, übergeben.

onOptionsItemSelected(MenuItem item)
Den Menüelementen werden die Funktionalitäten zugewiesen, welche für das Ausführen der gewünschten Aktion zuständig sind.

getShareIntent()
Es wird ein Intent zurückgegeben, welcher das aktuelle Bild beinhaltet.

showMap()

Die Aktivität GoogleMapsActivity wird gestartet, wenn GPS-Daten vorhanden sind. Dieser Aktivität werden zusätzlich Informationen über das Bild weitergegeben.
Sind keine Informationen über den Standort verfügbar, wird dies dem Anwender mitgeteilt.

savePictureInLibrary()

Das *currentPicture* wird in der Bildergalerie gespeichert.

ERGEBNIS

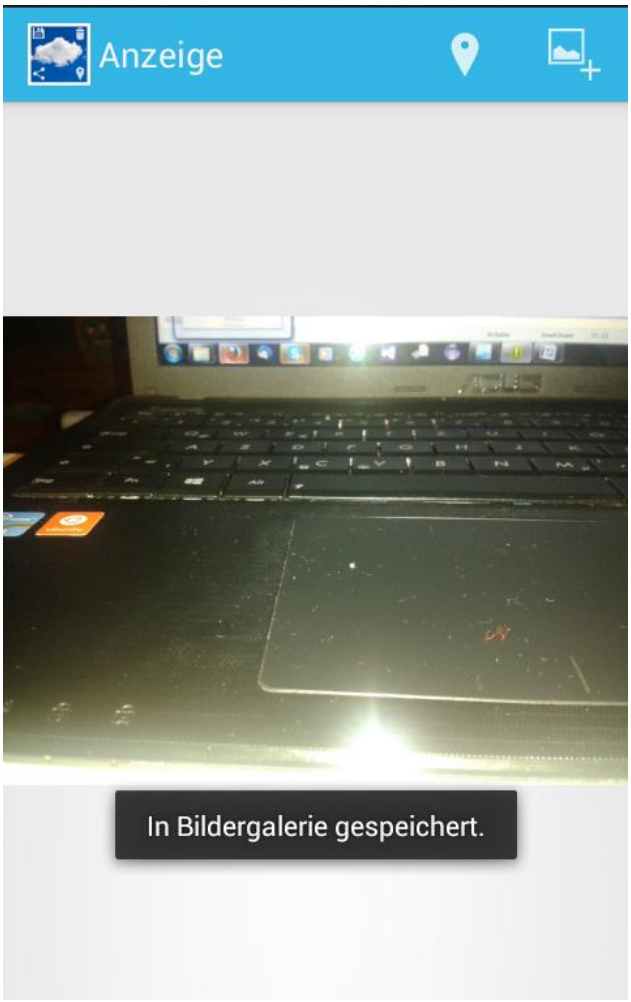

Abbildung 7: Anzeige eines Bildes in der ImageActivty mit Meldung

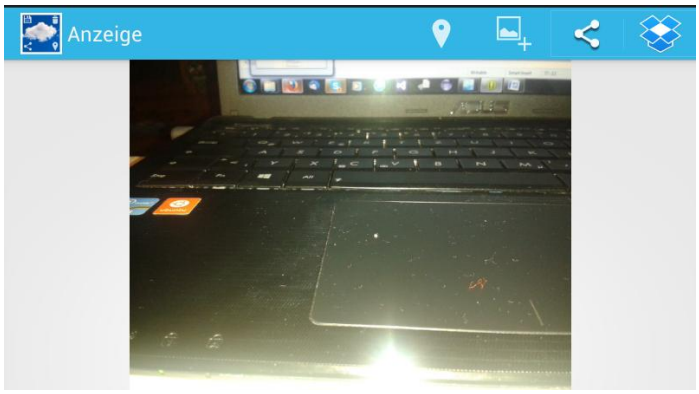

Abbildung 6: Anzeige aller Menüelemente in der ActionBar

DIE KLASSE GOOGLEMAPSACTIVITY

Die GoogleMapsActivity wird gestartet, wenn der Anwender in der ImageActivity den Menupünkt „*showOnMap*“ auswählt. Von der ImageActivity erhält die GoogleMapsActivity Informationen über den Standpunkt eines Bildes zum Zeitpunkt der Aufnahme. Diese werden weiterverarbeitet und graphisch dargestellt.

LAYOUT: ACTIVITY_GOOGLE_MAPS.XML

Das Layout der Activity besteht aus einem MapFragment . Dieses wird durch die Klasse SupportMapFragment aus dem Package com.google.android.gms.map der eingebunden Google-Play-Services-Library klassifiziert.
Weiterhin wurden dem MapFragment folgende Attribute zu einer individuellen Darstellung beigefügt:

```
map:mapType="hybrid"
map:uiRotateGestures="true"
map:uiScrollGestures="true"
map:uiZoomControls="true"
map:uiZoomGestures="true"
```

MENÜ: GOOGLE_MAPS.XML

Das Menü besteht aus zwei Elementen, welche in einer Gruppe zusammengefasst werden. Dabei handelt es sich um eine Auflistung, der möglichen Anzeigemöglichkeiten der Karte. Dem Anwender ist es nur möglich ein Element aus dieser Gruppe auszuwählen. Standardmäßig ist der Kartentyp ausgewählt, welcher im Layout definiert wurde.

ÜBERSICHT

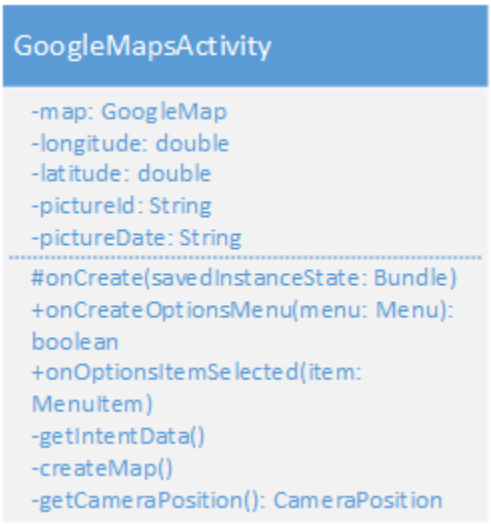

Abbildung 8: Klasse GoogleMapsActivity

ATTRIBUTE

GoogleMap map

Die GoogleMap, welche dem Anwender angezeigt wird und durch diesen ebenso modifiziert werden kann

Double longitude

Der Längengrad für ein Objekt des Typs Photo

Double latitude

Der Breitengrad für ein Objekt des Typs Photo

String pictureId

Die Identifikationsnummer für ein Objekt des Typs Photo

String pictureDate

Das Aufnahmedatum für ein Objekt des Typs Photo

METHODEN

onCreate(Bundle savedInstanceState)

In dieser Methode werden die Funktionen getIntentData() und createMap() aufgerufen. Diese instanziieren die Klassenattribute.
Anschließend wird die Kartenansicht auf die cameraPosition gerichtet, welche in der Funktion getCameraPosition() ermittelt wird.

onCreateOptionsMenu(Menu menu)

Das zuvor spezifizierte Menu der GoogleMapsActivity wird geladen.

onOptionsItemSelected(MenuItem item)

Diese Methode verändert je nach Auswahl des Anwenders die Darstellung der Karte.
Betätigt der Anwender den Menüpunkt „*satellite*“ wird der Menüpunkt markiert und die Karte ändert ihre Darstellungsform zum Typ MAP_TYPE_HYBRID. Analog dazu ändert die Karte ihre Darstellung zum Typ MAP_TYPE_NORMAL, wenn der Nutzer den Menüpunkt „*map*“ auswählt.

getIntentData()

Die Klassenattribute *longitude*, *latitude*, *pictureId* und *pictureDate* werden mit den Werten des übergebenen Intents instanziiert.

createMap()

Das Klassenattribut *map* wird instanziiert. Dabei wird dem Attribut *map* das zugehörige Fragment aus dem definierten Layout mit Hilfe eines FragmentManagers zugewiesen.
Zusätzlich dazu wird aktiviert, dass der Anwender seinen Standort auf der Karte einsehen kann.
Anschließend wird ein Marker an der Position gesetzt, an welcher das Bild aufgenommen wurde. Der Marker wird mit den zuvor instanziierten Attributen modifiziert. Der Anwender bekommt einen Marker angezeigt, welcher sich an der Position der Bildaufnahme befindet. Der Marker zeigt ebenso den Namen und das Aufnahmedatum an.

getCameraPosition()

Diese Funktion gibt die CameraPosition *cameraPosition* zurück. Die Position der Kartenkamera wird auf die Position des Standorts des aufgenommenen Bildes gesetzt. Auf diesen Punkt wird dann mit einem festen Wert gezoomt.

ERGEBNIS

Abbildung 9: Map in Satellitenansicht mit Marker

Abbildung 10: Map in Kartenansicht mit Marker

DIE KLASSE PHOTO

Die Klasse Photo steht sowohl in einer Beziehung mit der Klasse PhotoList als auch mit der Klasse ImageActivity. Beide Male stellt sie Teile der anderen Klassen dar.

ÜBERSICHT

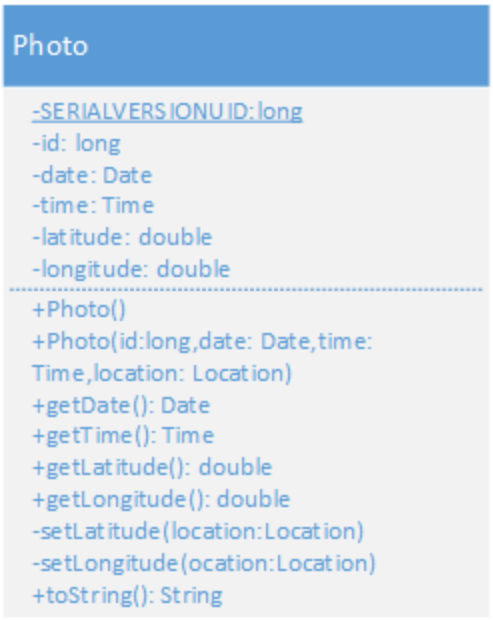

Abbildung 11: Klasse Photo

ATTRIBUTE

Long id

Die Identifikationsnummer für ein Objekt des Typs Photo

Date date

Das Datum, wann das Objekt erzeugt wurde

Time time

Die Zeit, wann das Objekt erzeugt wurde

Double latitude

Der geografische Breitengrad des Objekts

Double longitude
Der geografische Längengrad des Objekts

METHODEN

Photo()
Standard-Konstruktor der Klasse Photo. In ihm werden die Klassenattribute mit festdefinierten Standardwerten initialisiert.

Photo(String id, Date date, Time time, Location location)
Ein weiterer Konstruktor der Klasse Photo. Er initialisiert die Klassenattribute mit den übergebenen Werten. Um die Klassenattribute *longitude* und *latitude* zu initialisieren werden die Funktionen *setLatitude(location)* und *setLongitude(location)* aufgerufen.

getId()
Getter-Funktion für das Klassenattribut „*id*“

getDate()
Getter-Funktion für das Klassenattribut „*date*”

getTime()
Getter-Funktion für das Klassenattribut „*time*“

getLatitude()
Getter-Funktion für das Klassenattribut „*latitude*“

getLongitude()
Getter-Funktion für das Klassenattribut „*longitude*“

setLatitude(Location location)
Setter-Funktion für das Klassenattribut „*latitude*“.
Wenn eine Location existiert, wird das Klassenattribut mit dem der Location zugeordneten Latitude instanziiert.

setLongitude(Location location)
Setter-Funktion für das Klassenattribut „*longitude*“.
Wenn eine Location existiert, wird das Klassenattribut mit dem der Location zugehörigen Longitude instanziiert.

toString()

Gibt einen String zurück, welcher für die Darstellung eines Photo-Objektes in der PhotoList verantwortlich ist.

DIE KLASSE PHOTOLIST

Die Klasse PhotoList bildet den Mittelpunkt aller Klassen. Einerseits öffnet sie die Klasse ImageActivity und benutzt für die Ausführung ihrer Funktionen die Klasse SaveManager. Andererseits ist sie eine Ansammlung von Photo-Objekten und kann nicht ohne die MainActivity existieren.

ÜBERSICHT

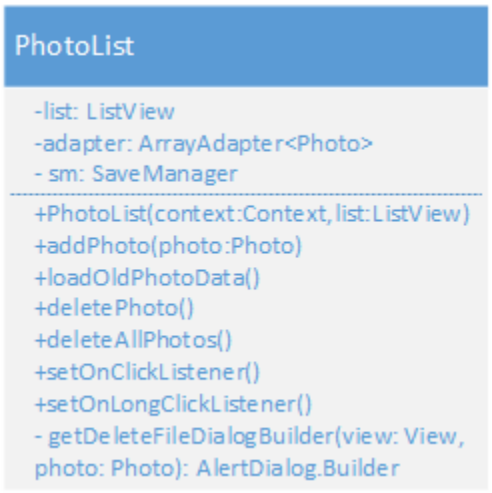

Abbildung 12: Klasse PhotoList

ATTRIBUTE

ListView list
Ein Objekt des Typs ListView

ArrayAdapter<Photo> adapter
Ein ArrayAdapter gefüllt mit Objekten des Typs Photo

SaveManager sm
Instanz der SaveManager-Klasse

METHODEN

PhotoList(Context context, ListView list)
Konstruktor der Klasse PhotoList.
Die Klassenattribute werden initialisiert: Dem Attribut „*list*“ wird eine übergeben ListView zugeordent. Der ArrayAdapter „*adapter*“ wird einem neuen ArrayAdapter mit dem übergebenen Context und dem Layout “simple_list_istem_1” zugeordnet. Der *list* wird der *adapter* beigefügt. Ebenso wird ein neuer SaveManager instanziiert.

addPhoto(Photo photo, Bitmap picture)
Dem Adapter wird ein neues Objekt vom Typ Photo angefügt. Ebenso wird die Methode saveEntirePictureData(Photo photo) des SaveManagers ausgeführt.

loadOldPhotoData()
Dem Adapter werden alle bereits mit zugehöriger XML-Datei gespeicherten Bilder angefügt.

deletePhoto(Photo photo)
Das Photo-Objekt wird aus dem Adapter entfernt, nachdem die Methode deleteEntirePictureData(Photo photo) ausgeführt wurde.

deleteAllPhotos()
Der Adapter wird geleert und die Methode deleteAllPictureData() (siehe SaveManager) wird aufgerufen.

setOnClickListener()

Für die Liste „*list*“ wird ein neuer OnItemClickListener gesetzt. Dieser erkennt in der onItemClick-Methode, welches Objekt vom Typ Photo aus der ListView angeklickt wurde und startet die ImageViewActivity. An diese gibt er das Photo-Objekt weiter.

setOnLongClickListener()

Für die Liste „*list*“ wird ein neuer OnItemLongClickListener gesetzt. Dieser erkennt in der onItemLongClick-Methode, welches Objekt vom Typ Photo in der Liste angeklickt wurde und zeigt einen erzeugten AlertDialog an.

getDeleteFileDialogBuilder(fianl View view, final Photo photo)

Ein AlertDialog.Builder wird erzeugt. Der Builder erhält einen Titel, eine Nachricht, einen PositivButton und einen NegativButton. Der PositivButton ruft, wenn er angeklickt wurde, die Methode deletePhoto(Photo photo) auf und das Bild wird mitsamt seinen Metadaten auf dem Gerät gelöscht. Benutzt der Anwender den NegativButttton, so wird der Löschvorgang abgebrochen. Der Anwender kehrt zur Übersicht der Bilder zurück.

.

DIE KLASSE SAVEMANAGER

Die Klasse SaveManager beinhaltet die ganze Logik bezüglich des Erzeugen, Speichern, Löschen, und Laden von Bild- und XML-Dateien. Hauptsächlich wird sie von der Klasse PhotoList verwendet.

ÜBERSICHT

Abbildung 13: Klasse SaveManager

ATTRIBUTE

PICTURE_PREFIX
Konstante, in der das Präfix für Bild- und XML-Dateien gespeichert ist.

photoDir
Das Verzeichnis in dem die Bilder gespeichert werden.

xmlDir
Das Verzeichnis in dem die XML-Dateien gespeichert werden.

xstream
Objekt zur Verarbeitung von XML-Dateien.

METHODEN

SaveManager()
Der SaveManager wird instanziiert. Dabei werden die zu nutzenden Dateipfade auf die Ordner "Photos" und "XMLs" bestimmt und diese bei Bedarf erstellt.

saveEntirePictureData(Photo photo)
Dient zur Speicherung jeglicher mit dem Foto verbundenen Daten.

List<Photo> loadEntirePictureData()
Es werden sämtliche in den XMLs gespeicherten Fotodaten geladen. Diese werden Fotoliste zurückgegeben.

Bitmap loadPictureWithId(String pictureId)
Gibt das Bild mit der gegebenen ID als Bitmap zurück.

long getUniqueId()
Überprüft die existierenden XML Dokumente im XML-Ordner der Applikation und wählt die nächste einzigartige ID als Inkrement der bislang höchsten ID im Ordner.

String getNewPictureFileName()
Gibt den Dateinamen für das nächste zu erstellende Bild zurück.

String getPictureFileName(long id)
Gibt den Dateinamen des Fotos mit der gegebenen ID zurück.

deleteEntirePictureData(Photo photo)
Löscht sowohl das gespeicherte Bild des übergeben Fotos, als auch die dazugehörige XML-Datei.

deleteAllPictureData()
Löscht das gespeicherte Bild und die dazugehörige XML-Datei für alle Fotos.

loadPictureMetaDataFromXml(File file)
Läd die Fotodaten aus der gegebenen XML-Datei und gibt das Foto zurück.

savePictureMetaDataInXml(Photo photo)
Speichert die Fotodaten des gegebenen Fotos als XML-Datei.

writeXmlToFile(String id, String xml)
Speichert den Xml-String des Fotos mit der gegebenen ID in eine Datei.

getXmlFileName(String id)
Gibt den Dateinamen der XML-Datei mit der gegebenen Foto-ID zurück.

deletePictureInDir(Photo photo)
Löscht das Bild zu dem übergebenen Photo-Objekt aus dem Verzeichnis.

deletePictureMetaDataXml(Photo photo)
Löscht die XML-Datei zu dem übergebenen Photo-Objekt.

deleteAllPictureInDir()
Löscht alle Bilddateien der Applikation.

deleteAllPictureMetaDataXml()
Löscht alle XML-Dateien der Applikation.

setDirectories()
Setzt die im SaveManager vermerkten Pfade auf Dateipfade, die auch außerhalb der Applikation verwendet werden können, damit der Inhalt mit anderen Applikationen geteilt werden kann.

createDirIfNotExist(File path)
Erzeugt die Ordnerstruktur für einen gegeben Pfad, sofern diese noch nicht existiert.

ANDROIDMANIFEST

Im Folgenden werden einige interessante Änderungen in der AndroidManifest-Datei erläutert.

SDK-VERSIONEN

En Endgerät benötigt mindestens ein API-Level von 14, um die Applikation ausführen zu können. Entwickelt wurde die Applikation für ein API-Level von 21.

BENUTZERRECHTE

Damit die Applikation fehlerfrei alle Operationen ausführen kann, muss sie über bestimmte Berechtigungen auf dem Endgerät verfügen:

- Standort:
  Netzwerkbasierter Standort, Genauer Standort (GPS)

- Speicher:
  Inhalt des USB-Speichers ändern/löschen

- Netzwerkommunikation:
  Vollständiger Internetzugriff, Netzwerkstatus anzeigen

- Hardwaresteuerung:
  Fotos und Videos aufnehmen

GOOGLE-PLAY-DIENSTE

Zur Einbindung der Google-Play-Dienste bedarf es weiterer Modifikationen in der Manifest-Datei der Applikation. Damit die Applikation die eingebundene “google-play-services_lib”-Library erkennt und mit einer Google-Maps arbeiten kann, musste dieses Package aus der Library in das Projekt eingebunden werden.

Ebenso wurde der Applikation die Berechtigung gegeben, dass sie Daten von der Karte empfangen kann.

THEMES UND STYLES

Damit nicht das Standard-Thema, sondern ein nicht vordefiniertes Thema das Erscheinungsbild der Applikation bestimmt, wurde ein neues Thema *CustomActionBarTheme* von dem Standard-Thema Theme.AppCompat.Light abgeleitet. In diesem abgeleiteten Thema, welches in der themes.xml zu finden ist, wurde die ActionBar modifiziert. In diesem Fall wurde nur die Hintergrundfarbe der ActionBar umgesetzt. Damit diese Änderungen zum Tragen kommen, wurde die AndroidManifest-Datei angepasst.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- the theme applied to the application or activity -->
    <style name="CustomActionBarTheme"
           parent="@style/Theme.AppCompat.Light">
        <item name="android:actionBarStyle">@style/MyActionBar</item>
    </style>

    <!-- ActionBar styles -->
    <style name="MyActionBar"
           parent="@style/Widget.AppCompat.Light.ActionBar.Solid.Inverse">
        <item name="android:background">#33B5E5</item>
    </style>
</resources>
```

Abbildung 14: themes.xml mit dem CustomActionBarTheme