Entwicklung einer Foto-App für Android

Studienarbeit

im Studiengang Angewandte Informatik

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Tobias Dorra und Philipp Pütz

November 2015

Bearbeitungszeitraum	4 Wochen
Kurs	TINF14-AIBC

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abbildungsverzeichnis

1 Einführung

Bei gemeinsamen Freizeitaktivitäten mit Freunden entstehen oft viele Fotos. Diese werden dann entweder gar nicht oder erst Wochen später miteinander geteilt.

Unser Ziel ist es, die Benutzererfahrung beim Teilen der Bilder zu verbessern. Das soll mit einer eigenen Foto-App passieren. Anstatt die gesammelten Bilder erst nach der Veranstaltung auszuwählen und dann an die entsprechenden Kontakte zu schicken, sollen die geschossenen Bilder bereits während der Veranstaltung automatisch mit den anderen anwesenden Freunden geteilt werden.

1.1 Funktionen

1.1.1 Organisation der Fotos

Die Fotos werden in sogenannten „Streams“ gruppiert. Jeder Stream repräsentiert dabei ein gemeinsames Erlebnis mit Freunden. Streams können umbenannt oder gelöscht werden. Beim Löschen eines Streams werden alle enthaltenen Bilder ebenfalls gelöscht.

Um einen Stream mit Fotos zu befüllen, kann entweder die Gerätekamera genutzt werden, oder es werden Fotos aus der Android-Galerie importiert. Die Fotos werden mit Name und Vorschaubild im Stream angezeigt. Außerdem existiert eine Detailansicht, in der man das Foto vergrößert betrachten kann. Fotos können umbenannt oder gelöscht werden. Ausgewählte Fotos können wieder zurück in die Android-Galerie exportiert, oder über die Teilen-Funktion anderen Apps zugänglich gemacht werden.

1.1.2 Teilen von Fotos

Zu einem Stream können mehrere Benutzer hinzugefügt werden. Die Fotos in einem Stream werden live zwischen allen Teilnehmern synchronisiert.

Alle Teilnehmer in einem Stream sind gleichberechtigt. Es gibt also keinen „Streamadministrator“ mit besonderen Rechten. Jeder Teilnehmer eines Streams kann Bilder hinzufügen, Bilder löschen und die

Teilnehmerliste verwalten. Insbesondere ist jeder Teilnehmer auch dazu berechtigt, weitere Teilnehmer hinzuzufügen.

Entfernt ein Benutzer einen Stream, so bleibt der Stream bei den anderen Teilnehmern noch vorhanden. Bei diesen wird lediglich der Benutzer aus der Teilnehmerliste entfernt.

Wird ein Benutzer nachträglich zu einem Stream hinzugefügt, so erhält er nicht nur die neuen Bilder, sondern bekommt auch automatisch Zugriff auf alle Bilder, die bereits vor seinem Eintritt im Stream vorhanden waren.

1.1.3 Nutzung alternativer Methoden zur Datenübertragung

Wenn man gezwungen ist, das Mobilfunknetz zu benutzen, ist die Internetverbindung oft schlecht. Außerdem haben viele Smartphone-Nutzer ein begrenztes Datenvolumen. Daher sollen die Daten nicht über das Internet, sondern über alternative Verbindungen übertragen werden. Diese beinhalten:

1. Bluetooth
2. Bluetooth Low Energy (eventuell, falls technisch machbar)
3. Wi-Fi Direct

1.2 Projektumfang

1.2.1 Umgesetzte Funktionalitäten

Die Anwendung, die im Rahmen dieses Projektes entwickelt wurde, stellt lediglich einen Prototyp dar. Es werden also nicht alle beschriebenen Funktionen umgesetzt. Bei der Architektur wurde jedoch darauf geachtet, dass die Anwendung um die beschriebenen Funktionalitäten erweiterbar ist. Die folgenden Funktionalitäten wurden im Prototyp umgesetzt:

ID	Beschreibung
F-10	Bilder können zur App hinzufügt werden
F-10.1	Bilder können über die Kamera aufgenommen werden
F-10.2	Bilder aus der Galerie geladen werden
F-20	Bilder können verwaltet werden

ID	Beschreibung
F-20.1	Bilder können gelöscht und umbenannt werden
F-20.2	Bilder haben verschiedene Attribute
F-30	„Streams“ (Ordner) gruppieren Bilder
F-30.1	Streams können erstellt, gelöscht und umbenannt werden
F-40	Bilder werden in einer listen ähnlichen Darstellung mit Vorschaubild und Bildinformationen angezeigt
F-50	Bilder können in einer Vollbildansicht angezeigt werden
F-50.1	Aus der Vollbildansicht können Bilder in die Galerie exportiert und per E-Mail und Bluetooth versendet werden
F-60	Die Architektur ist erweiterbar
D-10	Alle Appdaten (Streams, Bilder, Datenbanken) werden in einen speziellen Verzeichnis gespeichert
D-10	Gespeicherte Bilder erhalten als Zusatz das aktuelle Datum
D-20	Alle Bild- und Streaminformationen werden in einer Datenbank gespeichert
UI-10	Es existiert eine Hauptansicht
UI-10.1	Die Hauptansicht zeigt Vorschaubilder von aufgenommene und aus der Galerie geladenen Bilder an
UI-10.2	Zu jedem Bild werden in der Hauptansicht der Bildtitel und Aufnahmeinformationen angezeigt
UI-10.3	Durch an tippen der Bilder startet eine neue Activity die das Bild in einer Vollbildansicht anzeigt
UI-10.4	Durch einen „long tap“ auf ein Bild wir ein Kontext Menü angezeigt
UI-10.4.1	Im Kontext Menü stehen folgende Optionen zur Verfügung: Bild umbenennen, Bild löschen, Bild in Galerie exportieren
UI-10.5	Ein Floating-Action Button Menü bietet weitere Optionen
UI-10.5.1	Über das Menü können Bilder mit der Kamera aufgenommen werden

ID	Beschreibung
UI-10.5.2	Über das Menü können Bilder aus der Galerie werden
UI-10.6	Ein „Navigation Drawer“ bietet die Möglichkeit neue „Streams“ anzulegen, die ähnlich wie Ordner, Bilder gruppieren
UI-10.6.1	Die Hauptansicht zeigt den Inhalt eines Streams an
UI-10.7	Über das Hauptmenü können Streams umbenannt werden
UI-10.8	Über das Hauptmenü können Streams und alle zugehörigen Bilder gelöscht werden
UI-10.9	Eine Standardansicht animiert den Nutzer zum anlegen eines Streams, wenn kein Stream angelegt ist
UI-20	Es existiert eine Vollbildansicht für Bilder
UI-20.1	Die Vollbildansicht wird durch an tippen eines Bildes in der Hauptansicht geöffnet
UI-20.2	Es ist möglich ein Bild aus der Vollbildansicht in die Galerie zu exportieren
UI-20.3	Es ist möglich ein Bild aus der Vollbildansicht per Bluetooth zu versenden

1.2.2 Offene Funktionalitäten

Die im folgenden aufgelisteten Funktionalitäten wurden bisher nicht implementiert und werden für weitere Versionen vorgemerkt.

ID	Beschreibung
F-70	Es können Benutzer zu Streams hinzugefügt werden
F-70.1	Neue Benutzer erhalten Zurgiff auf vorhandene und zukünftige Bilder im Stream
F-80	Bilder werden zwischen den verschiedenen Nutzern live synchronisiert

ID	Beschreibung
F-90	Es existiert ein Art Synchronisationslog der die unterschiedlichen Aktivitäten der Benutzer untereinander abgleicht
F-100	Benutzer können aus Streams austreten
F-100.1	Tritt ein Benutzer aus einem Stream aus, so werden lokal alle Bilder des Streams und der Stream selbst gelöscht
F-100.2	Ausgetretene Benutzer werden bei anderen Benutzern aus der Streamteilnehmerliste entfernt
F-110	Es stehen verschiedene Synchronisationsmöglichkeiten zur Verfügung:
F-110.1	Bluetooth
F-110.1	Bluetooth Low Energy (falls verfügbar)
F-110.1	Wi-Fi Direct
F-120	Alle Streamteilnehmer können über eine Teilnehmerliste eingesehen werden
F-130	In der Vollbildansicht kann an Bilder herangezoomt werden
F-130	Durch Gesten kann in der Vollbildansicht zum nächsten oder vorherigen Bild gewechselt werden
F-140	Das gesamte UI ist für verschiedene Endgeräte (Tablets, Android TVs) optimiert
D-10	Der Syncronisationslog und die Teilnehmerlisten werden in der Datenbank gespeichert
UI-30	In der Hauptansicht eines Streams ermöglicht ein Menübutton die Teilnehmerliste einzusehen und Teilnehmer hinzuzufügen
UI-40	Der Menüpunkt „Einstellungen“ im Navigation Drawer ermöglicht die Synchronisationsmethode und allgemeine Synchronisationseinstellungen festzulegen

2 Architektur

Um die Anwendung modular und damit auch erweiterbar zu halten, haben wir uns bei der Umsetzung des Projektes am MVC-Prinziep orientiert. Deshalb lässt sich die gesammte Anwendung in drei Komponenten aufteilen: Das Model ist für die Repräsentation und Speicherung der Daten zuständig. Diese Daten werden im View angezeigt. Er beinhaltet in der Hauptsache die Layouts der verschiedenen Activities sowie die zugehörigen Code-Behind-Klassen. Der Controller ist für die Verarbeitung der Daten zuständigt und stellt somit das Bindeglied zwischen Model und View dar.

2.1 Model

Während die Bilder selber auf der SD-Karte des Gerätes gespeichert werden, haben wir uns dazu entschieden, die Metadaten in einer SQLite-Datenbank zu speichern. So sind auch komplexere Datenabfragen möglich, was uns später bei der Synchronisation der Daten zwischen den verschiedenen Streamteilnehmern zugute kommen wird.

2.1.1 Datenbanklayout

Die Datenbank besteht aus zwei Tabellen: „Picture“ und „Stream“.

Jeder Datensatz in der Tabelle „Stream“ repräsentiert einen Stream in der App. Die Tabelle enthält die folgenden Spalten:

1. **streamid**: Ordnet jedem Stream eine eindeutige ID zu.
2. **streamname**: Speichert den Name eines Streams.
3. **created**: Enthält einen Unix-Timestamp, der den Zeitpunkt der Erstellung eines Streams wiederspiegelt.

Jeder Datensatz in der Tabelle „Picture“ repräsentiert ein Bild in der App. Die Tabelle enthält tie folgenden Spalten:

1. **pictureid**: Ordnet jedem Bild eine eindeutige ID zu.

2. **picturename**: Speichert den Titel eines Bildes.
3. **created**: Enthält einen Unix-Timestamp, der angibt, wann ein Bild in die App importiert wurde.
4. **filename**: Speichert den Dateinamen, an dem die eigentliche Bilddatei zu finden ist.
5. **streamid**: Enthält die ID des Streames, in dem sich das Bild befindet.

2.1.2 Datenbankzugriff

Um auf die Datenbank zuzugreifen, gibt es pro Tebelle eine Klasse („TblStream“ und „TblPicture“). Diese enthält für jede Datenbankabfrage eine statische Methode. Diese Methoden können vom Controller benutzt werden, um Daten abzufragen oder zu verändern.

2.2 Controller

Teil des Controllers sind die beiden Klassen „PicturesManager“ und „StreamManager“. Sie stellen dem View alle Funktionalitäten zur Verfügung, die dieser zum Anzeigen und Editieren von Bildern und Streams benötigt.

- StreamManager
  1. (refreshListOfStreams) Abfragen der Liste aller Streams.
  2. (insertStream) Erzeugen eines Streams.
  3. (updateStream) Umbenennen eines Streams.
  4. (deleteStream) Löschen eines Streams, inklusive aller enthaltener Bilder.
- PicturesManager
  Der PicturesManager arbeitet immer auf allen Bildern eines speziellen Streams. Der Stream, der vom PicturesManager verwendet wird, wird diesem im Konstruktor übergeben.
  1. (findAllPictures) Abfragen einer Liste aller Bilder des Streams.
  2. (insertPicture) Erzeugen eines Bildes mithilfe einer Bilddatei, die sich bereits im richtigen Ordner befindet. Diese Methode wird zum importieren von Kameraaufnahmen verwendet.
  3. (importInsertPicture) Importiert ein Bild von einer beliebigen Uri. Diese Methode wird zum Importieren von Bildern von der Galerie verwendet.
  4. (deletePicture) Löscht ein Bild aus der Datenbank sowie die zugehörige Bilddatei.

5. (updatePicture) Ändert den Titel eines Bildes.

2.2.1 Multitasking

Alle Aktionen, die vom „PicturesManager“ oder „StreamManager“ ausgeführt werden, greifen über das Model indirekt auf die Datenbank zu. Desweiteren muss der „PicturesManager“ Bilder kopieren und verkleinern. Dies sind alles Aktionen, die potenziell lange dauern. Daher dürfen diese nicht im UI-Thread einer App ausgeführt werden. Das würde nämlich zur Folge haben, dass sich das Benutzerinterface aufhängt. Um das zu vermeiden, führen wir alle Aktionen in einem seperaten Thread aus. Dafür kommt die Klasse „AsyncTask“ von Android zum Einsatz.

2.2.2 Kommunikation mit dem View

Nachdem Daten geändert wurden, muss dies dem View mitgeteilt werden, damit dieser die Darstellung aktualisieren kann.

Dafür existieren die Interfaces „IPicturesCallback“ und „IStreamsCallback“. Sie definieren Callback-funktionen, die die Code-Behind-Klassen aus dem View über Änderungen der Daten informieren. Dafür müssen diese das jeweils passende Interface implementieren und sich beim „StreamManager“ oder „PicturesManager“ als Callback-Objekt anmelden. Die beiden Controller-Klassen können dann in den jeweils passenden Situationen die passende Callback-Funktion des registrierten Callback-Objektes aufrufen. So wird der View über die Änderung der Daten informiert.

2.3 View