Dokumentation der Galerie-App „Piccer“

Seminararbeit „Entwicklung mobiler Applikationen - Thema Android“

des Studiengangs Angewandte Informatik International Business Competence

an der Dualen Hochschule Baden-Württemberg Mannheim

von

Arwed Mett, Dominic Steinhauser

29. November 2015

Bearbeitungszeitraum	8 Wochen
Matrikelnummer, Kurs	4278042, 1807718, TINF AIBC 2015
Ausbildungsfirma	SAP SE, Walldorf
Betreuer	Herr Sommer
Gutachter	Herr Sommer

Erklärung

Wir erklären hiermit:

1. dass wir unsere Seminararbeit „Entwicklung mobiler Applikationen - Thema Android“ mit dem Thema *Dokumentation der Galerie-App „Piccer“* ohne fremde Hilfe angefertigt haben;
2. dass wir die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet haben;
3. dass wir unsere Seminararbeit „Entwicklung mobiler Applikationen - Thema Android“ bei keiner anderen Prüfung vorgelegt haben;
4. dass die eingereichte elektronische Fassung exakt mit der eingereichten schriftlichen Fassung übereinstimmt.

Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Mannheim, 29. November 2015

Arwed Mett, Dominic Steinhauser

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einleitung

Im Rahmen der Vorlesung „Entwicklung mobiler Applikationen - Thema Android“ an der DHBW Mannheim, ist es vorgesehen ein Projekt durchzuführen, dessen Ziel es ist eine lauffähige App für das Betriebssystem Android zu erstellen.
Es ist den Studenten freigestellt, welche Android-Versionen die erstellte App unterstützen soll.„Piccer“ ist ab der Android-Version 4.4, API Level 19, lauffähig.
Als Testgeräte wurden ein „Motorola Moto X“ der zweiten Generation mit dem Betriebssystem Android 5.1 (Lollipop), sowie ein „Samsung Galaxy S3“ mit dem Betriebssystem CyanogenMod 11.0, welches Android 4.4 entspricht, genutzt.

1.1 Aufgabenstellung

1. Erstellen Sie eine App, die Fotos über ein Menü aufnimmt und aus der Fotogalerie lädt.
2. Speichern Sie das aufgenommene Foto in einem speziellen Verzeichnis mit aktuellem Datum und Uhrzeit.
3. Stellen Sie die aufgenommenen Fotos in einer Liste dar. In der Liste soll der Name des Fotos und das erstellte Datum angezeigt werden.
4. Wenn man auf ein Foto tippt, dann wird eine zweite Activity gestartet, indem das gemachte Foto in einer ImageView angezeigt wird.
5. Eine weitere Option soll das Versenden des Fotos per E-Mail ermöglichen.
6. Eine weitere Option soll das Speichern des Fotos in die Fotogalerie ermöglichen.

1.2 Funktionen von „Piccer“

„Piccer“ kann:

1. Fotos aufnehmen und aus der Galerie laden.

2. Thumbnails in einer Liste mit Datum, Uhrzeit und Titel anzeigen.
3. Mehrere Fotos, die in der Liste ausgewählt wurden, löschen, in der Galerie speichern oder teilen.
4. Die Anordnung der Liste kann verändert werden. Entweder ist das erste Foto, das angezeigt wird, das zuletzt hinzugefügte oder das zu erst hinzugefügte.
5. Ein ausgewähltes Foto in der ImageView, als zweite Activity, anzeigen.
6. Im ImageView kann das ausgewählte Foto versendet, gelöscht oder in der Galerie gespeichert werden.
7. Außerdem kann der Titel geändert werden.

2 Bedienung der Anwendung

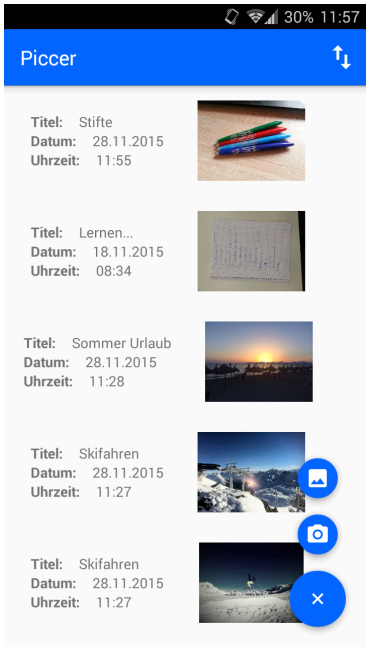

Abbildung 2.1: Übersicht

Abbildung 2 zeigt die Anwendung nach dem Start. Zu sehen ist eine Liste von Bildern, die der App entweder durch das Aufnehmen eines Fotos oder das Laden eines Bildes aus der Galerie, hinzugefügt wurde. Links neben dem Foto werden ein Titel, das Datum und die Uhrzeit, an dem das Bild aufgenommen wurde, angezeigt.
Wird ein Bild aus der Galerie geladen und es liegt ein Datum vor, wann das Bild erstellt wurde, so zeigt „Piccer“ dieses Datum an. Falls es nicht möglich ist, ein solches Datum aus der Datei auszulesen, wird das Datum gesetzt, wann das Bild der App hinzugefügt wurde.

Der Floating-Action-Button im unteren rechten Eck ermöglicht das Aufnehmen oder Laden von Fotos in die Anwendung.

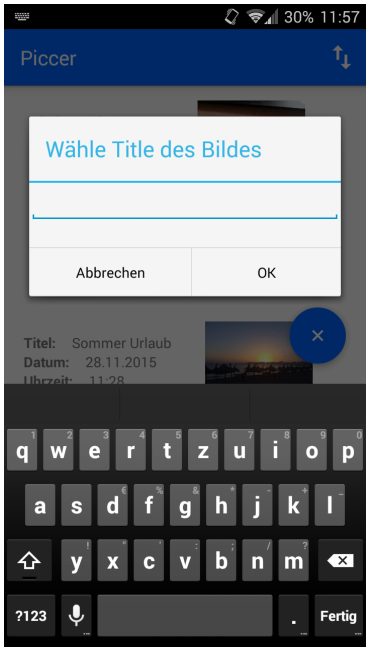

Abbildung 2.2: Titel vergeben

Im oberen rechten Rand kann die Auswahl zum Anzeigen der Liste umgekehrt werden. Es ist dem Nutzer überlassen, ob das zuletzt aufgenommen Foto zuerst (Sortierung der Bilder in der Liste: neu nach alt) oder zuletzt (Sortierung: alt nach neu) angezeigt werden soll.

Abbildung 2 zeigt den Bildschirm nachdem ein Foto aufgenommen wurde. Dabei kann der Nutzer einen Titel dem Bild hinzufügen. Falls auf „Abbrechen“ geklickt wird, dann wird das Bild ohne Titel abgespeichert.

Ist es der Fall, dass in der Liste, durch langes Drücken, ein oder mehrere Bilder markiert sind, so ist es möglich über ein Menü am rechten oberen Bildschirm, diese zu löschen, teilen oder in der Galerie zu speichern. Dies ist in Abbildung 2 dargestellt.

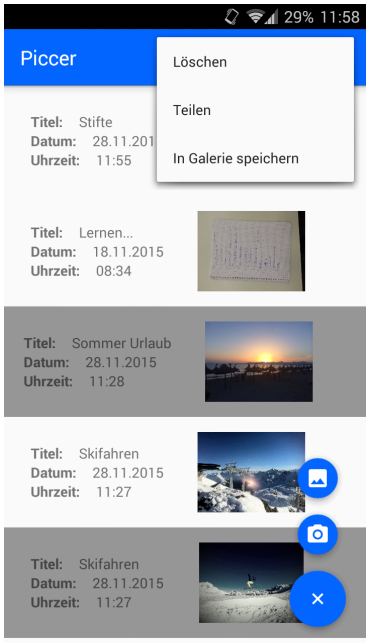

Abbildung 2.3: Mehrfachauswahl

Durch einmaliges, kurzes Drücken auf ein Bild der Liste, wird die zweite Activity gestartet. Der zweite Bildschirm 2 zeigt das ausgewählte Bild in voller Größe in einem „ImageView“. Hier ist ebenfalls ein Floating-Action-Button enthalten. Dieser ermöglicht es, das Bild nach links bzw. rechts zu drehen oder den Titel zu verändern. Außerdem lässt sich das Bild auch einzeln, über das Menü rechts oben, löschen, teilen oder in der Galerie speichern.

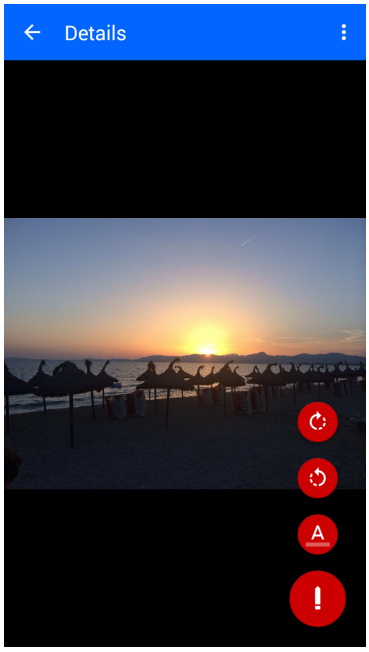

Abbildung 2.4: Detail Ansicht

3 Technische Dokumentation

3.1 Persistenz

Die Bilder der Anwendung werden in einem Applikations internen Ordner gespeichert. Um ein Bild laden zu können und Metainformationen über das Bild zu speichern, wird in einer SQLite Datenbank, Metainformationen von jedem Bild gespeichert. Teil dieser Metainformationen ist der Pfad zu dem jeweiligen Bild. Außerdem war die Anforderung an die Applikation, dass das Datum und die Uhrzeit an der das Bild erstellt wurde angezeigt werden soll. Diese Daten finden sich ebenfalls in der Datenbank.

3.2 Klassen

Die Applikation besteht aus zwei Aktivitäten. Die Piccer Aktivität zeigt eine Liste von Bildern (siehe Abbildung 2) und ist die Hauptaktivität, wohingegen die ImageDetailView Aktivität (siehe Abbildung 2) einen ImageView eines selektierten Bildes zeigt. Die restlichen Klassen sind Hilfsklassen. Im folgenden werden die wichtigsten Klassen beschrieben.

3.2.1 Piccer

Die Piccer Klasse ist eine Aktivität, die die Hauptseite verwaltet. Sie kümmert sich um das Laden von Views und initialisiert die Hauptliste. Außerdem dient sie zur Verwaltung von Menüs und startet eine neue Aktivität sobald auf ein Listenelement gedrückt wird.

3.2.2 ImageItem

Das ImageItem modelliert ein Listenelement. Es beinhaltet Referenzen auf die Datei in der das Bild gespeichert und Informationen über das Bild, wie z.B. Titel und Datum.

3.2.3 PiccerDatabaseHandler

Diese Klasse speichert Daten über die Bilder in einer Datenbank. Sie dient als Hilfsklasse zum Speichern und Laden von ImageItems.

3.2.4 ImageItemAdapter

Die Hauptliste ist durch einen ListView realisiert. Dieser ListView braucht einen Adapter zur Beschaffung von Daten, die in der Liste angezeigt werden sollen. Die Klasse ImageItemAdapter ist ein Adapter, der Daten aus einer Datenbanktabelle ließt und anschließend die jeweiligen Views der Hauptliste mit diesen befüllt.

3.2.5 ImageThumbnailLoader

Damit Bilder in einer Liste angezeigt werden können, ohne dass der Ladevorgang die Liste beim scrollen ins Stocken bringt, müssen die Bilder auf einem separaten Thread geladen werden. Ein Objekt der Klasse ImageThumbnailLoader lädt ein Bild und platziert es anschließend in einem ImageView. Dabei wird das Bild auf einem separaten Thread geladen.

3.2.6 ImageDetailView

Diese Klasse ist eine Aktivität, die ein Bild anzeigt. Außerdem bietet sie dem Benutzer die Möglichkeit das Bild zu drehen oder einen neuen Titel zu setzen.

3.2.7 AsyncRotator

Mit Hilfe dieser Klasse lassen sich Bilder drehen. Dabei wird die Rotierfunktion auf einem separaten Thread ausgeführt. Dadurch wird verhindert, dass die Benutzeroberfläche blockiert wird. Die Klasse legt zum Drehen der Daten ein temporäres File an. Sobald das Bild gedreht wurde und in dem temporären File gespeichert ist, wird das eigentliche File des zu drehenden Bildes, durch das temporäre ersetzt. Dies ist notwendig, da die Operation auf einem separaten Thread ausgeführt wird. Würde keine temporäre Datei verwendet werden und ein anderer Thread versucht zur gleichen Zeit, in der das Bild gedreht wird, Daten aus der Datei zu lesen, würde dies zu einer Beschädigung der Datei führen.

3.3 Implementierung am Beispiel von Funktionen

3.3.1 Versenden von Bildern

```
ArrayList<Uri> imageUris = new ArrayList<>();

for (long id : ids) {
    ImageItem imageItem = this.handler.getImage(this, id);
    imageUris.add(imageItem.getImageUri());
}

Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND_MULTIPLE);
sendIntent.putParcelableArrayListExtra(sendIntent.EXTRA_STREAM,
    imageUris);
sendIntent.putExtra(sendIntent.EXTRA_TEXT, R.string.sendMessage);
sendIntent.setType("image/*");
sendIntent.addFlags(sendIntent.FLAG_GRANT_READ_URI_PERMISSION);
startActivity(Intent.createChooser(sendIntent, getResources().getText(R
    .string.share)));
```

Um mehrere Bilder zu versenden, müssen erst alle dazugehörigen URIs in einer ArrayList gespeichert werden. Danach wird ein neuer Intent erstellt, die Funktion „ACTION_SEND_MULTIPLE“ gesetzt und anschließend die ArrayList übergeben, die die zu versendenden Bilder enthält. Es wir zusätzlich noch eine Nachricht erstellt und ein Menü, das alle installierten Apps auf dem Smartphone, die Inhalt versenden können, geöffnet. Im der Klasse „ImageDetailView“ verläuft dieser Vorgang anlog, nur reicht es aus ein einziges Image zu versenden.

3.3.2 Laden von Bildern aus der Galerie

Wird in der ersten Activity die Funktion ausgewählt, dass ein Bild aus der Galerie geladen werden soll, so startet die in Listing 3.1 beschriebene Methode „loadPicture“. Diese erstellt einen Intent und gibt die URI an den Konstruktor der Klasse ImageItem weiter. Der Konstruktor lädt dann eine Kopie des Bildes aus der Galerie in die App „Piccer“.

```
    public void loadPicture(View view) {
        Intent intent = new Intent(Intent.ACTION_PICK, android.provider
            .MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(intent, REQUEST_GALLARY);
    }
```

Listing 3.1: Klasse Piccer: Laden von Bildern aus der Galerie

```
this.context = context;
        generateName();
        BufferedInputStream in = null;
        BufferedOutputStream out = null;
        try {

            InputStream src = context.getContentResolver().
                openInputStream(uri);
            File dest = getFile();

            in = new BufferedInputStream(src);
            out = new BufferedOutputStream(new FileOutputStream(dest));

            byte[] buffer = new byte[1024];
            in.read(buffer);
            do {
                out.write(buffer);
            } while (in.read(buffer) != -1);

            notifyCache();
        } catch (FileNotFoundException e) {
            Toast.makeText(context, R.string.couldNotLoadImage, Toast.
                LENGTH_SHORT).show();
            Log.w("Piccer", "error", e);
        } catch (IOException e) {
            Toast.makeText(context, R.string.couldNotLoadImage, Toast.
                LENGTH_SHORT).show();
            Log.w("Piccer", "error", e);
        } finally {
            try {
                if (in != null) in.close();
                if (out != null) out.close();
            } catch (IOException e){}
        }
```

Listing 3.2: Klasse ImageItem: Laden von Bildern aus der Galerie

3.3.3 Vergeben eines Titels

Es wird ein Dialog geöffnet in dem ein Titel vergeben werden kann (siehe Listing 3.3). Anschließend wird der Titel durch die Methode `saveImage` in der Datenbank gespeichert.

```
public void addImage(final ImageItem imageItem){

    AlertDialog.Builder builder = new AlertDialog.Builder(this);
```

```
4       builder.setTitle(R.string.pleaseSelectName);
5
6       // Set up the input
7       final EditText input = new EditText(this);
8       // Specify the type of input expected;
9       input.setInputType(InputType.TYPE_CLASS_TEXT );
10      builder.setView(input);
11
12      // Set up the buttons
13      builder.setPositiveButton(R.string.ok, new DialogInterface.
            OnClickListener() {
14          @Override
15          public void onClick(DialogInterface dialog, int which) {
16              String title = input.getText().toString();
17              imageItem.setTitle(title);
18              saveImage(imageItem);
19          }
20      });
21      builder.setNegativeButton(R.string.abort, new DialogInterface.
            OnClickListener() {
22          @Override
23          public void onClick(DialogInterface dialog, int which) {
24              saveImage(imageItem);
25              dialog.cancel();
26          }
27      });
28
29      builder.show();
30  }
```

Listing 3.3: Titel vergeben

3.3.4 Löschen von Bildern

Um ein Bild zu löschen muss sowohl das File, in dem das Bild gespeichert ist gelöscht werden als auch der Tabelleneintrag in der Datenbank. Dafür gibt es in der Klasse PiccerDatabaseHandler die Methode `deleteImages` (siehe Listing 3.4), mit der mehrere Bilder durch eine Menge von IDs gelöscht werden können.

```
1   public void deleteImages(Set<Long> imageIds) {
2       SQLiteDatabase db = this.getWritableDatabase();
3       String query = "SELECT " + PATH + " FROM " + TABLE_IMAGES
4               + " WHERE _id in (";
5       String subquery = "";
6       for(long id : imageIds) {
```

```
7           subquery += id + ", ";
8       }
9       subquery = subquery.substring(0, subquery.length() - 2);
10      query += subquery;
11      query += ")";
12      Cursor c = db.rawQuery(query, null);
13      c.moveToFirst();
14      do {
15
16          File file = new File(this.context.getExternalFilesDir("img"), c
                .getString(c.getColumnIndex(PATH)));
17          file.delete();
18      } while (c.moveToNext());
19      query = "DELETE FROM " + TABLE_IMAGES + " WHERE _id in (" +
            subquery + ")";
20      db.execSQL(query);
21      db.close();
22  }
```

Listing 3.4: Löschen mehrerer Bilder

3.3.5 Laden von Thumbnails

Ein großes Problem ist es Bilder in einer Liste anzuzeigen, da diese erst von der Festplatte geladen werden müssen und eine große Menge von Daten darstellen. Dadurch kann eine lange Ladezeit entstehen, wodurch das Scrollen der Liste anfängt zu ruckeln. Um dieses Problem zu lösen gibt es mehrere Möglichkeiten. Einerseits kann das Bild redundant abgespeichert werden. Dabei wird eine deutlich kleinere Kopie des Bildes redundant gespeichert. Dies verkürzt die Ladezeit des Bildes, löst aber nicht immer das Problem. Auf einem Smartphone mit einer sehr langsamen Festplatte kann es immer noch zu langen Ladezeiten kommen. Eine zweite Möglichkeit besteht darin, alle Bilder zu Beginn der Anwendung in den Hauptspeicher zu laden. Allerdings haben viele Smartphone nur einen begrenzten Arbeitsspeicher, wodurch bei einer zu großen Liste die Anwendung abbricht.

Deshalb werden in der „Piccer“ Anwendungen die Thumbnails asynchron auf einem separaten Thread geladen. Dabei wird auch nicht das komplette Bild geladen sondern nur ein eine komprimierte Version. Um nicht bereits geladene Bilder erneut laden zu müssen, hat die Klasse ImageItem ein privates statisches Attribut, welches die bereits geladenen Thumbnails zwischenspeichert. Dieses Attribut wird als Cache bezeichnet, es kümmert sich darum, dass Bilder die häufiger benötigt werden länger gespeichert werden als Bilder die nur einmal geladen werden. Der Cache hat einen begrenzten Speicherbereich, sobald dieser voll ist, beginnt der Cache damit Bilder, die nicht oft benötigt werden, zu

verwerfen. Dadurch kann es nicht passieren, dass entweder die Liste ins Stocken gerät oder der Speicher überläuft.

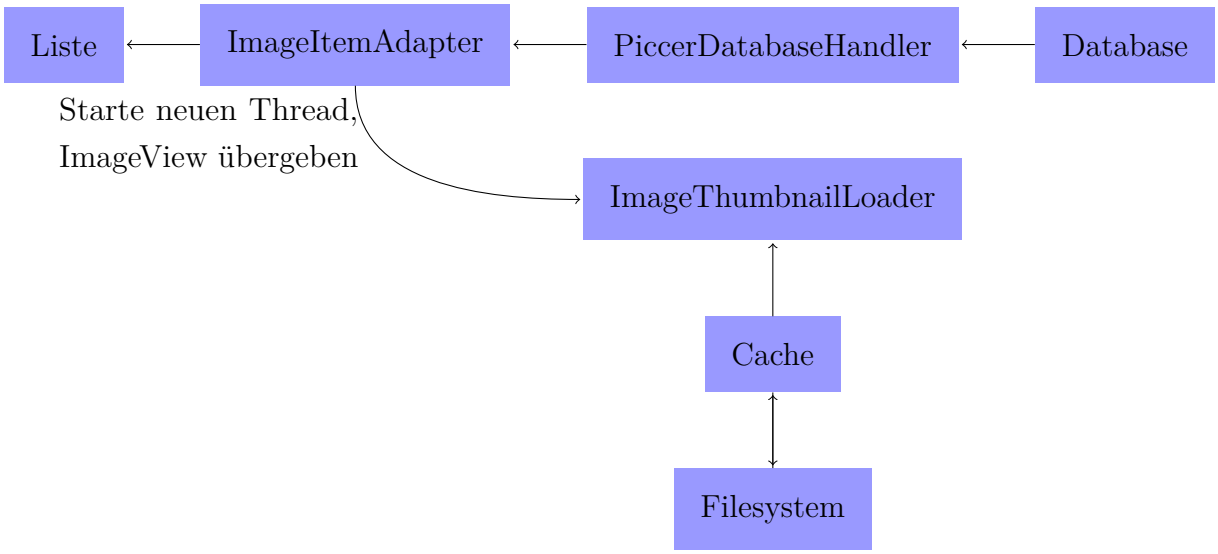

Abbildung 3.1: Ladevorgang von Thumbnails

3.3.6 Rotieren von Bildern

Um ein Bild zu rotieren, muss es anschließend auch in der Datei gespeichert werden. Dieser Vorgang dauert recht lange, denn das Bild muss geladen, gedreht und wieder gespeichert werden. Die meiste Zeit davon wird allerdings zum Laden und speichern benötigt. Deshalb wird in der Applikation „Piccer“ nur der ImageView eines Bildes gedreht. Anschließend wird auf einem eigenen Thread das Bild geladen, gedreht und gespeichert.