



ENTWICKELN VON ANWENDUNGEN FÜR HAND HELD

App für Erfassung von Garantiescheinen

Seminar Arbeit

Studenten: Andreas Grünenfelder
 Micha Schönenberger

Dozent: Christian Vils

© 2012

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung ausserhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

Zusammenfassung

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque mauris pede, blandit sed, hendrerit at, pharetra eget, dui. Sed lacus. Pellentesque malesuada. Cras gravida mi id sapien. Ut risus justo, fermentum non, scelerisque sit amet, lacinia in, erat. Proin nec lorem. Quisque porta, nisl at porta aliquam, felis libero consequat ipsum, vitae scelerisque dolor mi a odio. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis sollicitudin. Proin sollicitudin varius arcu. Morbi eleifend, metus sit amet placerat pharetra, dolor dui lobortis pede, vel imperdiet tellus eros imperdiet lorem. In hac habitasse platea dictumst. Curabitur elit mi, facilisis nec, ultricies id, aliquet et, magna. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam ac est. Mauris turpis enim, feugiat non, imperdiet congue, scelerisque non, purus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam dictum aliquet purus. Maecenas faucibus. Maecenas suscipit.

Abstract

Fusce neque est, tincidunt eu, nonummy nec, tempor iaculis, erat. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum egestas, velit a rhoncus gravida, metus dolor pulvinar diam, sit amet placerat risus dolor sit amet elit. Maecenas eget purus ut est mattis porta. Suspendisse ut mi et mauris lobortis malesuada. Vestibulum dapibus. Duis hendrerit, elit eu venenatis eleifend, sapien ante volutpat odio, ac condimentum tellus massa ut massa. Etiam dapibus imperdiet metus. Sed sapien arcu, pulvinar quis, laoreet quis, venenatis non, justo. Aliquam est ante, pulvinar nec, accumsan sed, auctor sed, augue.

Ut adipiscing ligula. In mattis. Ut varius. In nec nulla at eros molestie viverra. Duis dolor risus, lobortis vel, dictum a, pellentesque id, lectus. Sed suscipit orci ac ligula venenatis condimentum. Maecenas et sem lacinia tortor cursus tempus. Mauris pellentesque risus at nulla. In arcu. Curabitur mattis mi quis dolor. In leo. Vivamus ut libero.

Inhaltsverzeichnis

| | |
|---|------------|
| Abkürzungsverzeichnis | II |
| Abbildungsverzeichnis | III |
| Tabellenverzeichnis | IV |
| Verzeichnis der Listings | V |
| 1. Einleitung | 1 |
| 1.1. Das Projekt | 1 |
| 1.1.1. Ausgangslage | 1 |
| 1.1.2. Ziel der Arbeit | 1 |
| 1.1.3. Aufgabenstellung | 1 |
| 1.1.4. Erwartetes Resultat | 1 |
| 1.1.5. Geplante Termine | 2 |
| 2. Projektplanung | 3 |
| 2.1. Gantt Chart | 3 |
| 2.2. Arbeitsaufwände | 3 |
| 3. Grundlagen App Programmierung | 4 |
| 3.1. Was wird für die App Programmierung benötigt | 4 |
| 3.1.1. Eclipse (IDE) | 4 |
| 3.1.2. Android SDK Plugin for Eclipse | 5 |
| 3.1.3. Testing | 5 |
| 3.2. Aufbau der Architektur | 6 |
| 3.2.1. Java Class | 6 |
| 3.2.2. Layouts | 6 |
| 3.2.3. R.java | 7 |
| 3.2.4. strings.xml | 9 |
| 3.2.5. Manifest.xml | 10 |
| 3.3. App-Vermarktung | 12 |
| 3.3.1. Android Market - Google Play Store | 12 |
| 3.3.2. public link | 13 |
| 3.3.3. Mail, Stick... | 13 |

| | |
|---|------------|
| 4. Warranty App | 14 |
| 4.1. Grundidee | 14 |
| 4.2. Features | 14 |
| 4.2.1. mögliche Erweiterungen | 14 |
| 4.3. Aufbau der App | 15 |
| 4.3.1. Klassendiagramm | 15 |
| 4.3.2. Permissions | 15 |
| 4.3.3. Datenbank | 15 |
| 4.4. Core Komponenten | 16 |
| 4.4.1. Ansteuerung der Kamera | 16 |
| 4.4.2. Datenbank | 16 |
| 4.4.2.1. Datenbanklayout | 16 |
| 4.4.2.2. Datenbank erstellen | 17 |
| 4.4.2.3. Insert und Update | 18 |
| 4.4.2.4. Delete | 19 |
| 4.4.3. Layouts | 19 |
| 5. Fazit | 20 |
| 5.1. Punkt 1 | 20 |
| 5.1.1. Punkt 1.1 | 20 |
| A. Anhang | i |
| A.1. Verwendete Werkzeuge | ii |
| A.1.1. Software | ii |
| A.1.2. Hardware | ii |
| Literaturverzeichnis | iii |

Abkürzungsverzeichnis

| | |
|-----------|--|
| IDE | I ntegrated D evelopment E nvironment |
| SDK | S oftware D evelopment K it |

Abbildungsverzeichnis

| | |
|---|----|
| 3.1. Logo Eclipse IDE | 5 |
| 3.2. Beispiel AVD Manager | 6 |
| 4.1. Datenbankschema Tabelle Warranty | 16 |
| 4.2. Datenbankschema Tabelle Warranty | 17 |

Tabellenverzeichnis

Verzeichnis der Listings

| | |
|--|----|
| 3.1. R.java | 7 |
| 3.2. MainActivity.java | 7 |
| 3.3. strings.xml | 9 |
| 3.4. AndroidManifest.xml | 10 |
| 4.1. Attributdeklaration | 17 |
| 4.2. Tabellen Creation Statement | 17 |
| 4.3. SQL Statement | 18 |
| 4.4. Tabellen Creation Statement | 18 |
| 4.5. deleteCard Methode | 19 |

1. Einleitung

1.1. Das Projekt

1.1.1. Ausgangslage

Aus den ersten beiden Studienjahren haben wir uns die Grundkenntnisse der Java-Programmierung angeeignet. Wir möchten dieses Wissen nutzen, um ein neues Gebiet zu betreten (Native-App Android) und uns einem Thema zu widmen, das uns interessiert, wir aber bis anhin keine Zeit gefunden haben. Keiner von uns hat berufliche Programmiererfahrung. Deshalb liegt all unsere Erfahrung auf den schulischen Kenntnissen.

1.1.2. Ziel der Arbeit

Unser primäres Ziel ist es, einen Einblick in die Programmierung von Android Apps zu haben. Zusätzlich möchten wir unser bereits angeeignetes Java-Wissen auffrischen und vertiefen.

1.1.3. Aufgabenstellung

Wir möchten eine Android App erstellen, die es dem User ermöglicht, Garantiescheine in Form von einem Foto einfach lokal auf dem Smartphone zu verwalten. Die App soll die Möglichkeit bieten, zusätzliche Details in Form von Text zu speichern. Da im Fokus vor allem der Einblick in die App-Programmierung steht, verzichten wir bewusst gänzlich auf Netzwerk-Unterstützung. Des Weiteren ist das Backup der Fotos sowie der dazugehörigen Details nicht Teil dieser Arbeit, da dies unsere Zeitlimiten übersteigen würde.

1.1.4. Erwartetes Resultat

Das erwartete Resultat ist eine Native-App, welche auf Android funktioniert. Folgende Anforderungen müssen erfüllt sein.

- Kein Absturz der Applikation
- Lauffähig auf Geräten mit OS > 2.2 (Froyo) bis hin zum aktuellen 4.1.x (Jelly Bean)
- Fotos können aufgenommen und lokal gespeichert werden

- Es können Details in Form von Freitext zu den Fotos hinzugefügt werden
- Garantiescheine sollen nach folgenden Kriterien sortiert werden können
 - Speicherdatum des Garantiescheins
 - Alphabetisch nach Titel
 - Anzahl Tage bis Garantie ausläuft

1.1.5. Geplante Termine

- 3 Oktober 2012 Einschreiben des Projektes im EBS
- 5. Dezember 2012 Abgabe Teaser im EBS
- 12. Dezember 2012 Arbeitstreffen
- 9. Januar 2013 Abgabe der Dokumentation
- 16. Januar 2013 Präsentation

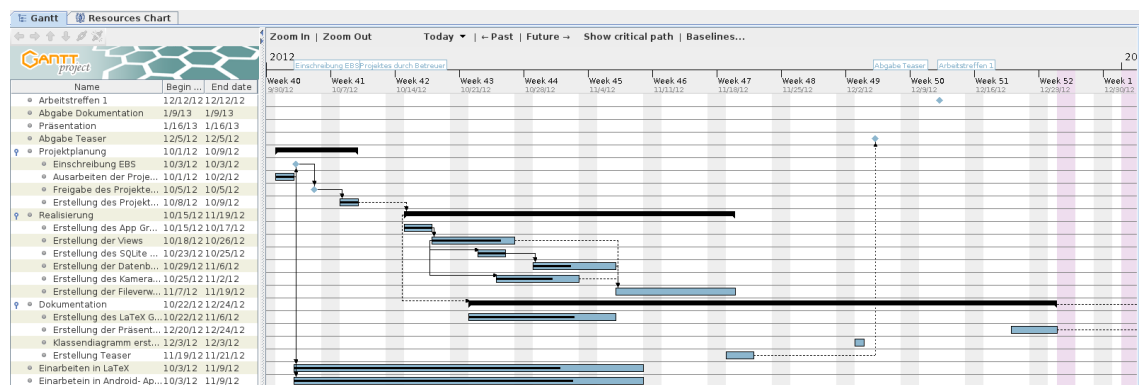
| Datum | Beschreibung |
|-------------------|-----------------------------------|
| 3 Oktober 2012 | Einschreiben des Projektes im EBS |
| 5. Dezember 2012 | Abgabe Teaser im EBS |
| 12. Dezember 2012 | Arbeitstreffen |
| 9. Januar 2013 | Abgabe der Dokumentation |
| 16. Januar 2013 | Präsentation |

2. Projektplanung

2.1. Gantt Chart

-mit Verweis was Gantt Chart ist ;-)

HIER EIN PREVIEW DER GANTT CHART



2.2. Arbeitsaufwände

3. Grundlagen App Programmierung

HIER KOMMT TEXT: KEINE ERFAHRUNG, BLABLABLA...

3.1. Was wird für die App Programmierung benötigt

Es gibt mehrere Möglichkeiten, ein Android App zu programmieren. Neben Software, die es erlaubt, offline zu programmieren, gibt es auch online Tools, bei welchen man keine bis sehr wenig Programmierkenntnisse benötigt.

Unsere Vorkenntnisse in der Programmiersprache Java basieren auf den beiden Module Programmieren 1 und Programmieren 2 des Grundstudiums.

-grosse Palette (einige Beispiele) -Google App Inventor <http://beta.appinventor.mit.edu>
=> ohne Vorkenntnisse

Da wir bereits im Grundstudium die opensource Programmiersoftware Eclipse nutzten, beschränken wir uns hier auch auf diese Software.

3.1.1. Eclipse (IDE)

Eclipse ¹ [2] (vom englischen eclipse = Sonnenfinsternis hergeleitet) ist ein open-source Programmierwerkzeug. Zu Beginn wurde Eclipse als eine Entwicklungsumgebung für Java entwickelt. Im Laufe der Zeit hat sich Eclipse weiterentwickelt und durch die Möglichkeit der Skalierbarkeit wurde vom Java-Programmiertool ein Werkzeug, welches für viele Entwicklungsaufgaben eingesetzt werden kann. Die grosse Community und der modulare Aufbau, welche die Weiterentwicklung vom Modulen und Plugins immer vorantreiben, haben aus diesem Tool ein mächtiges Tool gemacht, welches sich für den Entwickler individuell zuschneiden lässt. Es gibt für Eclipse mittlerweile open-souce sowie auch kommerzielle Erweiterungen. Eclipse selbst basiert auf Java-Technik, seit Version 3.0 auf einem sogenannten OSGi-Framework namens Equinox.

Speziell für die Entwicklung von Android Applikationen existiert das ADT (Android Development Tools) Plug-in. Dieses Plug-in erweitert den Funktionsumfang von Eclipse und ermöglicht somit ein einfaches Entwickeln von Android Projekten. Eclipse wurde als Grundwerkzeug für die App-Programmierung benutzt. In den ersten beiden Studienjahren haben wir mit Eclipse Java-Applikationen entwickelt.

¹ offizielle Website: [http://de.wikipedia.org/wiki/Eclipse_\(IDE\)](http://de.wikipedia.org/wiki/Eclipse_(IDE))



Abbildung 3.1.: Logo Eclipse IDE

3.1.2. Android SDK Plugin for Eclipse

Android Software Development Kit (SDK) ² [1] ist ein Plugin für Eclipse IDE welches als mächtige, integrierte Entwicklungsumgebung konzipiert wurde, um Android Applikationen zu entwickeln.

Will man beginnen, Android Apps zu programmieren, kommt man um das Android SDK nicht herum.

Die Android SDK gibt es für Windows, Mac OS X sowie Linux Plattformen.

Um Android SDK nutzen zu können, ist die Java SDK (Software Development Kit) unabdingbar. Diese ist je nach Betriebssystem bereits vorinstalliert oder kann nachträglich heruntergeladen und installiert werden.

Im Android SDK integriert ist der AVD Manager (Android Virtual Device Manager). Dieser ermöglicht das Testen der App in einer virtuellen Umgebung. Das Betriebssystem, die Speichermöglichkeiten sowie das Telefon können beliebig geändert werden.

offizielle Website: <http://developer.android.com>

Anleitung Installation Plugin: <http://developer.android.com/sdk/installing>

3.1.3. Testing

Da die Zeit für dieses Projekt nicht ausreicht für ein ausgiebiges Testen mit JUnit- und JMock-Klassen, haben wir uns auf ein Testing beschränkt auf das Live-Testing. Zur Auswahl standen:

- Galaxy Nexus, Version 4.0.1

²<http://developer.android.com/sdk/index.html>

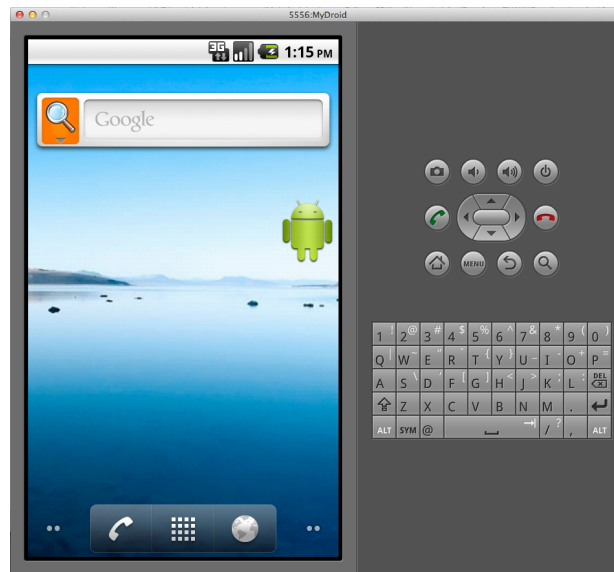


Abbildung 3.2.: Beispiel AVD Manager

- HTC Desire HD, Version 2.3.5
- virtuelle Maschine, welche in Eclipse integriert ist und sich wahlweise die Android Version, aber auch Telefentyp ändern lässt

3.2. Aufbau der Architektur

3.2.1. Java Class

3.2.2. Layouts

3.2.3. R.java

R.java ist eine selbstgenerierte Java Klasse. Sie speichert für jede Ressource eine Integer-Konstante. Für die Applikationsentwicklung ist es nicht notwendig, diese Datei einzusehen. Anbei ein Ausschnitt der R.java Datei unser Applikation.

```
1  /* AUTO-GENERATED FILE. DO NOT MODIFY.
2   * This class was automatically generated by the
3   * aapt tool from the resource data it found. It
4   * should not be modified by hand.
5   */
6
7  package ch.zhaw.warranty;
8  public final class R {
9      public static final class id {
10         public static final int BTQuit=0x7f07000f;
11     }
12 }
```

Listing 3.1: R.java

Für die anderen Klassen hat R.java jedoch eine sehr grosse Bedeutung: Da alle Ressourcen in R.java eine konstante Integer zugewiesen ist, hat jede .java Klasse einen Verweis auf R.java, damit das richtige Layout geladen werden kann. Anbei ein Auszug aus MainActivity.java. Diese Activity ist die erste Activity, die geladen wird. Sie ist verknüpft mit dem Layout **activity_main**.

```
1  package ch.zhaw.warranty;
2
3  import ch.zhaw.warranty.R;
4  import ...;
5
6  public class MainActivity extends Activity {
7      public static TBLWarrantyConnector tblwarranty;
8
9      @Override
10     public void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.activity_main);
13     ...
```

```
14 }
```

Listing 3.2: MainActivity.java

3.2.4. strings.xml

Die Datei **strings.xml** wird verwendet, um alle sichtbaren Texte, welche zur Laufzeit der App auf dem Bildschirm erscheinen, zu verwalten.

Als Programmierer sollte man beachten, dass eine Beschriftung eines Buttons nicht hard-coded wird, da die Flexibilität und die lose Kopplung verloren gehen. Wird stattdessen auf die Variable in der strings.xml Datei verwiesen, können alle App-Texte zentral verwaltet werden.

Ist dies einmal gegeben, ist ein Hinzufügen einer weiteren Sprache kein Problem mehr. Defaultmässig liegt die strings.xml Datei im Verzeichnis `../res/values/strings.xml` und definiert die englische Sprache.

Möchte man nun eine weitere Sprache hinzufügen, erstellt man für die entsprechende Sprache einen neuen Ordner (Beispiel: `../res/values-de/`). Nun kann die Datei strings.xml aus `../res/values/` kopiert werden und die Englischen Texte auf Deutsch angepasst werden.

Es gibt zwei Varianten, die Sprache der App zu ändern:

- Wird nichts eingestellt, wird die App in der Sprache gestartet, welche die Landeseinstellungen des Android Smartphone vorgeben. Ist die entsprechende Sprache in der App nicht vorhanden, wird per default Englisch benutzt.
- In der App kann ein Menüpunkt eingebaut werden, über welchen man auf jede beliebige Sprache, welche die App beherrscht, umstellen kann.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <string name="quit_app">Quit Warranty</string>
4   <string-array name="browse_list_spinner">
5     <item>Mercury</item>
6     <item>Earth</item>
7   </string-array>
8 </resources>
```

Listing 3.3: strings.xml

3.2.5. Manifest.xml

Die Datei **Manifest.xml** ist gewissermassen das Herzstück der App. Sie enthält die wichtigsten Informationen über die App [4]:

- enthält wichtige Informationen, damit die App auf einem System laufen kann
- Manifestdatei enthält Metadaten einer App (Paketname)
- Angabe genutzter Komponenten (Activities, Services, Broadcast Receivers, Content Providers)
- Fähigkeiten der App
- Voraussetzungen zum Betrieb (z.B. nötige Bibliotheken)
- Intent-Filter und IntentReceiver der Activities
- Zugriffsrechte auf andere Dienste und Rechte, die andere Apps für den Zugriff haben müssen
- Angabe von Angeboten für andere Apps
- Geforderte Android API

Welche Elemente die Manifest.xml Datei annehmen kann, sind auf der Android Developer Website ³ ersichtlich.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="ch.zhaw.warranty"
4     android:versionCode="1"
5     android:versionName="1.0">
6     <uses-sdk android:minSdkVersion="4" android:targetSdkVersion="12" />
7     <uses-feature android:name="android.hardware.camera" />
8     <uses-permission android:name="android.permission.
9         WRITE_EXTERNAL_STORAGE" />
10    <application android:icon="@drawable/icon" android:label="@string/app_name">
11        <activity android:name="ch.zhaw.warranty.MainActivity"
12            android:label="@string/app_name">
13            <intent-filter>
14                <action android:name="android.intent.action.MAIN" />
15                <category android:name="android.intent.category.LAUNCHER" />
16            </intent-filter>
17        </activity>
```

³<http://developer.android.com/guide/topics/manifest/manifest-intro.html>

```
16     <activity android:name="ch.zhaw.photobyintent.BrowseList"></activity>
17     ...
18 </application>
19 </manifest>
```

Listing 3.4: AndroidManifest.xml

3.3. App-Vermarktung

Online gibt es hunderte von Abhandlungen über die Vermarktungs-Strategie von Android Apps.

Es gibt Anleitungen, wie man seine App unter die 100 Besten im Play Store bringen kann, wie man den Preis festlegen soll, so dass der maximale Profit erwirtschaftet werden kann, wie man sich einen Namen macht als Entwickler... Da sind keine Grenzen gesetzt.

zwei Beispiele:

<http://www.androidpit.de/Strategische-Herangehensweise-bei-der-App-Entwicklung-Die-Idee-Teil-1>

<http://theappencypress.com/2010/08/04/everything-you-need-to-know-about-being-an-android-app-seller/>

3.3.1. Android Market - Google Play Store

Die wohl bekannteste Variante ist der Android Market. Seit dem 06. März 2012 heisst dieser offiziell Google Play [3] Store.⁴ Gemäss Angaben von Wikipedia [5] waren Ende Januar 2012 über 360'000 Anwendungen verfügbar, welche insgesamt über 10 Milliarden mal heruntergeladen wurden. Zirka 15% der Anwendungen sind Spiele. Der Umsatz beträgt mehr als 5 Millionen US-Dollar pro Monat.

Um eine Anwendung auf dem Play Store anbieten zu können, muss man sich als Entwickler auf <https://play.google.com/apps/publish/signup> registrieren und einmalig 25 US-Dollar bezahlen. Nach der Registrierung stehen einem Tür und Tor offen. Man kann seine Applikationen nun kostenlos oder zu einem selbst ernannten Preis vermarkten. Momentan sind etwa 65% der Applikationen kostenlos verfügbar.

Google verlangt, genauso wie Apple und Microsoft, eine Transaktionsgebühr von 30% des Verkaufswert.

Es gibt jedoch auch Alternativen [5] zu Googles Play Store:

- **F-Droid** Ein Appstore, der als non-profit Projekt von einer Community freiwilliger Unterstützer betrieben wird und über den ausschließlich kostenlose, freie Software-Apps (Open Source) bereitgestellt werden.

<http://f-droid.org/>

- **SlideME** SlidME bietet eine Plattform für Entwickler und Benutzer von Android. Entwickler können ihre Applikation kostenlos oder auch kostenpflichtig bei SlideME

⁴ offizielle Website: <https://play.google.com>

veröffentlichen. Der SlideME Market umfasst ca. 2000 Applikationen.

<http://www.slideme.org>

- **AndroidPIT** AndroidPIT betreibt einen eigenen Store und bietet auch anderen Unternehmen diesen Store für ihre Android-Geräte an. Hierzu gehören Unternehmen wie 1&1, Telefunk, Pearl, Point of View und Interpad.

<http://www.androidpit.de/>

- ... und noch viele mehr ...

3.3.2. public link

Will man seine Applikation nicht in einem online-Store wie Google Play Store anbieten, kann man sie auch direkt verkaufen oder kostenlos anbieten.

Hier liegt ein Vorteil von Google gegenüber anderen Anbietern, wie zum Beispiel Apple. Auf jedem Gerät, auf welchem die Android Plattform als Betriebssystem läuft, kann man die Funktion ein- und ausschalten, welche es erlaubt, auch Applikationen von Fremdanbietern (also nicht Play Store von Google) zu installieren. Dafür ist kein Jailbreak oder dergleichen notwendig.

Jeder, der diese Funktion nun eingeschaltet hat, kann die Applikation von jeglichem Webserver auf der Welt herunterladen.

Sicherheitshinweis: Hier sollte man unbedingt beachten, dass man auch schädliche Software installieren kann, wenn man erlaubt, aus nicht Google-Seiten Applikationen zu laden!

3.3.3. Mail, Stick...

Als dritte Variante bietet sich die direkte Vervielfältigung an. Diese kann per Mail, USB-Stick oder dergleichen erfolgen. Jede Android Applikation endet mit **.apk**. Ist das Smartphone am Computer angeschlossen (Windows, Mac OS X, Linux), kann die Applikation ohne Probleme installiert werden.

4. Warranty App

4.1. Grundidee

Die Grundidee der **Warranty-App** besteht darin, Garantiescheine zu verwalten. Es gibt 3 Möglichkeiten, keine Garantie auf ein Gerät zu erhalten:

- Garantie ist abgelaufen: hier gibt's nicht mehr zu helfen
- Gerät wurde zu unsorgfältig gehandelt. Selbstverschulden.
- Garantieschein ist verloren oder im Haushalt unauffindbar

Am dritten Punkt knüpft dieses App an. Ein Garantieschein kann nur noch dann verloren gehen, wenn die digitalen Daten weg sind. Mit einem Backup auf dem Rechner zu Hause oder in der Cloud ist dies nicht mehr möglich. In vielen Fällen schwirren die Garantiescheine irgendwo im Haushalt herum oder die schwarze Farbe gleich sich mit der Zeit dem weissen Papier an und nach ein paar Monaten bis Jahre ist vom Garantieschein nicht mehr zu lesen..

4.2. Features

Die bereits eingebauten Features sind bereits zum Teil bereits im Kapitel [1.1.4 Erwartetes Resultat](#) erwähnt worden.

4.2.1. mögliche Erweiterungen

Bei den Erweiterungen sind grundsätzlich den Phantasien keine Grenzen gesetzt. Wir beschränken uns auf die sinnvollen, im Alltag nutzbaren Erweiterungen.

- Möglichkeit, die Datensätze zu exportieren inkl. Bilder, welche sehr wichtig sind
- Abgleich der lokalen Datensätzen, bzw. Datenbank mit einem Clouddienst wie Dropbox, GoogleDrive...
- Versenden von einzelnen Datensätzen per E-Mail

Für viele dieser Erweiterungen wäre es sicherlich möglich, auf bereits geschriebenen Code der Android-App-Entwickler zuzugreifen. Somit müsste das Rad nicht neu erfunden werden.

4.3. Aufbau der App

4.3.1. Klassendiagramm

4.3.2. Permissions

Die Warranty App benötigt wenige zusätzliche Rechte. Um Bilder speichern zu können ist jedoch ein Speicherplatz erforderlich. Da die meisten Android basierten Smartphones einen sehr kleinen internen Speicher haben, bietet sich der externe Speicher bestens an. Je nach Modell ist dies entweder eine zusätzliche MicroSD [Glossar] Karte oder einfach eine zusätzliche logische Partition auf dem internen Speicher. Der Vorteil dieses externen Speichers ist, dass man ihn ohne weiteres auf dem Computer einhängen und auf die Dateien zugreifen kann.

Um auf diesen externen Speicher zuzugreifen, wird folgende Zeile im Manifest.xml benötigt.

```
1 <uses-permission android:name="android.permission.  
    WRITE_EXTERNAL_STORAGE" />
```

Nebst dem Zugriff auf den Speicher, bedient sich Warranty den Kamerafunktionalitäten. Da diese ebenfalls explizit erlaubt werden müssen, wird das entsprechende Recht im Manifest.xml hinterlegt.

```
1 <uses-permission android:name="android.permission.CAMERA" />
```

4.3.3. Datenbank

Auch wenn wir auf unserem Smartphone eine Datenbank benötigen, so scheint die Idee, ein vollumfängliches DBMS⁵ wie beispielsweise MySQL oder PostgreSQL zu installieren absurd. Zum einen benötigen wir Features wie ein Client/Server Model, Partitioning oder ein ausgefeiltes Zugriffsberechtigungssystem nicht, zum anderen steht die dazu benötigte Performance auf einem Smartphone schlicht und einfach nicht zur Verfügung.

Um dennoch eine Datenbank auf einem Smartphone verwenden zu können, bietet sich

⁵Database Management System

SQLite an. SQLite ist eine Programmbibliothek, die sich direkt in der Applikation einbinden lässt und somit keinen Server- Prozess benötigt, also ressourcensparend ist. Die gesamte Datenbank inklusive aller Tabellen, Indizes und Werten werden in einer einzigen Datei abgelegt, was ein paralleles Schreiben auf die Datenbank unmöglich macht.

Dank der nativen SQLite Unterstützung von Android, fällt ein aufwändiges Einbinden einer 3rd Party Library weg.

4.4. Core Komponenten

4.4.1. Ansteuerung der Kamera

4.4.2. Datenbank

4.4.2.1. Datenbanklayout

Da die Anzahl der Datentypen in SQLite auf NULL, INTEGER, REAL, TEXT und BLOB beschränkt ist, <http://www.sqlite.org/datatype3.html>, ist das Datenbankschema von Warranty schnell definiert.

Um die Einzigartigkeit der Einträge zu gewährleisten, ist die ID als auto- increment gekennzeichnet. Somit können wir sicherstellen, dass auch IDs von gelöschten Einträgen nicht wiederverwendet werden.

Da SQLite keine Datentypen für Datum und Zeit zur Verfügung stellt, wir aber trotzdem Datumsstempel benötigen, greifen wir auf den Datentyp TEXT und Datumsfunktionen wie `date` von SQLite zurück http://www.sqlite.org/lang_datefunc.html.

| Table warranty | | | | |
|----------------|---------|------|------|-------------------------|
| Column | Type | Null | Auto | Comment |
| _id | INTEGER | no | yes | ID |
| title | TEXT | | | Titel der Warranty Card |
| description | TEXT | | | Beschreibung |
| img_path | TEXT | | | Pfad zum Bild |
| created_at | TEXT | | | Erstellungsdatum |
| valid_until | TEXT | | | Ablaufdatum |
| price | REAL | | | Preise |
| reseller | TEXT | | | Verkaufsstelle |

Abbildung 4.1.: Datenbankschema Tabelle Warranty

Der einzige Index liegt auf der ID, da in der App sämtliche Aktionen mittels dieser ID als Referenz ausgeführt werden. Da wir keine Volltextsuche implementiert haben, benötigen wir kein weiteren Indizes.

| Indexes | | |
|-----------|-------------|-----------|
| Column(s) | Type | Sort |
| _id | Primary Key | ascending |

Abbildung 4.2.: Datenbankschema Tabelle Warranty

4.4.2.2. Datenbank erstellen

Ist das Datenbankschema definiert, ist die Implementierung in der App eine kleine Sache. Zuerst werden die Attribute benannt. Wachsamem Lesern fällt auf, dass sämtliche Strings `public static final` sind. Dies erlaubt es den Programmierern, von überall her dynamisch auf die Namen der Attribute zuzugreifen, sodass bei einer allfälligen Anpassung des Schemas immer die korrekten Namen in den SQL Statements verwendet werden.

```

1 public class TBLWarrantyHelper extends SQLiteOpenHelper {
2     public static final String TBL_NAME="warranty";
3     public static final String CLMN_ID = "_id";
4     public static final String CLMN_TITLE = "title";
5     public static final String CLMN_DESC = "description";
6     public static final String CLMN_IMGPATH = "img_path";
7     public static final String CLMN_CREATEDAT = "created_at";
8     public static final String CLMN_VLDTIL = "valid_until";
9     public static final String CLMN_PRICE = "price";
10    public static final String CLMN_RESELLER = "reseller";

```

Listing 4.1: Attributdeklaration

Anschliessend wird das Creation- Statement der Tabelle mit den dazugehörigen Datentypen vorbereitet.

```

1 private static final String DB_CREATE="create table " + TBL_NAME + " (" +
2     CLMN_ID + " integer primary key autoincrement," +
3     CLMN_TITLE + " text," +
4     CLMN_DESC + " text," +
5     CLMN_IMGPATH + " text," +
6     CLMN_CREATEDAT + " text," +
7     CLMN_VLDTIL + " text," +
8     CLMN_PRICE + " real," +
9     CLMN_RESELLER + " text);";

```

Listing 4.2: Tabellen Creation Statement

Zum Schluss wir die von SQLiteOpenHelper [Glossar] geerbte Methode onCreate(SQLiteDatabase db) Ãberschrieben und darin das Statement als SQL ausgefÃhrt.

```

1  @Override
2  public void onCreate(SQLiteDatabase db) {
3      db.execSQL(DB_CREATE);
4  }
```

Listing 4.3: SQL Statement

4.4.2.3. Insert und Update

Die aus Anwendersicht vermutlich wichtigsten Methoden sind die Insert- und die Update-Methoden. Da diese von der FunktionalitÃt sehr Ãhnlich sind, ist aus Sicht des Programmierers sinnvoll, diese zusammen zulegen. Der grundlegende Unterschied ist , dass beim Insert ein neuer Eintrag erstellt, beim Update ein bereits vorhandener Eintrag angepasst wird.

Im Javacode ist die Differenzierung denkbar einfach.

```

1  public void insertWarrantyCard(WarrantyCard card){
2      ...
3      if (card.get_id() == 0) {
4          db.insert(TBLWarrantyHelper.TBL_NAME, null, values);
5      } else {
6          db.update(TBLWarrantyHelper.TBL_NAME, values, TBLWarrantyHelper.
              CLMN_ID + "=" + card.get_id(), null);
7      ...
8      }
```

Listing 4.4: Tabellen Creation Statement

Der Grund fÃr diese einfache Unterscheidung liegt in der Auflistung der Quittungen im Homescreen [Glossar] der App. Diese sogenannte ListView [Glossar] kennt von jeder aufgelisteten Quittung ihre dazugehÃrige ID. MÃchte der User eine Quittung bearbeiten, wird in der App intern diese Referenz auf die Quittung weitergegeben. MÃchte der User eine neue Quittung hinterlegen, wird das ID Feld nicht gefÃhrt. Dies fÃhrt dazu, dass der Standard Integer- Wert, in Java eine 0, weitergegeben wird. <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

4.4.2.4. Delete

Da die Quittungen nach deren Ablauf nicht automatisch gelöscht werden, kann es durchaus sein, dass ein User die Quittungen manuell löschen möchte. Wie bereits im Kapitel „Insert und Update“ erwähnt, ist der Listview auf dem Homescreen die ID jeder aufgelisteten Quittung bekannt. Somit kann eine Quittung direkt aus der Listview mit dem Methodenaufruf `deleteCard` und der dazugehörigen ID, die anschliessend das entsprechende SQL- Statement an die Datenbank kommuniziert, gelöscht werden.

Wie bereits beim Insert und Update haben wir uns auch hier die Tatsache, dass das auto increment von SQLite bei eins beginnt zu nutzen gemacht. Wird null als ID übergeben, hat dies zur Folge, dass ausnahmslos alle gespeicherten Quittungen gelöscht werden.

```
1 public void deleteCard(int cardID) {  
2     ...  
3     if (cardID == 0) {  
4         db.delete(TBLWarrantyHelper.TBL_NAME, null, null);  
5     } else {  
6         db.delete(TBLWarrantyHelper.TBL_NAME, TBLWarrantyHelper.CLMN_ID + "="  
7             + cardID, null);  
8     }  
9     ...  
}
```

Listing 4.5: deleteCard Methode

Eine Funktion zum Löschen aller Quittungen wurde im Menü des Homescreens untergebracht.

4.4.3. Layouts

Die Warranty App basiert auf zwei Layouts, dem Homescreen mit der Übersicht über alle gespeicherten Quittungen sowie einer Detailansicht pro Quittung.

Der Homescreen der Applikation besteht aus einer Listview, die den oberen Teil des Bildschirms ausfüllt. Darin werden sämtliche gespeicherten Quittungen aufgelistet.

Mit der sich darunter befindenden Checkbox können zusätzlich auch bereits abgelaufene Garantiescheine angezeigt werden.

5. Fazit

5.1. Punkt 1

5.1.1. Punkt 1.1

A. Anhang

A.1. Verwendete Werkzeuge

Im Folgenden werden die Hardware und Software vorgestellt, welche die Autoren zum Erstellen dieser Arbeit und vor allem zur Entwicklung der App verwendet haben. Es wurden ausschliesslich Open-Source-Programme eingesetzt.

Hier benutzte Beschreibungen können von Website (offizielle Site der Software, Wikipedia...) übernommen sein. Dieser Abschnitt dient zur Information für die verwendeten Werkzeuge.

A.1.1. Software

- **L^AT_EX**


Diese Arbeit wurde mit L^AT_EX geschrieben. Als Distribution und Editor wurde auf dem Mac OS Mountain Lion TexShop verwendet, auf Basis von Linux ????????????

Websites: <http://pages.uoregon.edu/koch/texshop/>


A.1.2. Hardware

- **Galaxy Nexus**

Auf diesem Smartphone läuft das brandaktuelle Andoid OS 4.1.1 (Nelly Bean).

| Bezeichnung | Version |  |
|-------------------|--------------------|---|
| model number | Galaxy Nexus | |
| Android-Version | 4.1.1 (Jelly Bean) | |
| Baseband-Version | I9250XXLF1 | |
| Kernel-Version | 3.0.31-g6fb96c9 | |
| Build number | JR003C.I9250XWLH2 | |
| Screen Resolution | 1280 x 720 pixel | |
| diagonal | 4.65 inch | |

- **HTC Desire HD A9191**

| Bezeichnung | Version |  |
|-------------------|---|---|
| model number | HTC Desire HD A9191 | |
| Android-Version | 2.3.5 (Gingerbread) | |
| Baseband-Version | 12.65.60.29U ₂ 6.14.04.28 _M | |
| Kernel-Version | 2.6.35.10-g931a37e | |
| Build number | | |
| Screen Resolution | 800 × 480 pixel | |
| diagonal | 4.3 inch | |

Literaturverzeichnis

- [1] *Android Development Tools (ADT)*. <http://developer.android.com/tools/sdk/eclipse-adt.html>, november 2012. 3.1.2
- [2] *Eclipse (IDE)*, *Wikipedia*. [http://de.wikipedia.org/wiki/Eclipse_\(IDE\)](http://de.wikipedia.org/wiki/Eclipse_(IDE)), november 2012. 3.1.1
- [3] *Google Play Store*. <https://play.google.com>, december 2012. 3.3.1
- [4] *Manifest Datei - Uni Dortmund*. <http://ls13-www.cs.uni-dortmund.de/dokuwiki-fachprojekt-ss11/lib/exe/fetch.php?media=lammers-vortrag.pdf>, november 2012. 3.2.5
- [5] *Wikipedia - Google Play Store*. http://de.wikipedia.org/wiki/Google_Play, december 2012. 3.3.1