

Projektarbeit

Name: Ramon Michel Leber

Thema: Entwicklung eines Allergie Beschwerdebooks
als Android-Applikation

Dozent: Prof. Dr. Alznauer

Abgabetermin: 31.08.2014

Kurzfassung

Die vorliegende Projektarbeit befasst sich mit der Entwicklung eines Allergie-Beschwerdebuchs als Android-Applikation. Dabei wird die Entwicklung einer Android-Applikation im Allgemeinen erörtert und auf das Beispiel des Allergie-Beschwerdebuchs angewandt.

Einige der grundlegendsten und wichtigsten Funktionen, die ein Android-Smartdevice ausmachen, werden in dieser Projektarbeit behandelt. Im Besonderen spielen der Umgang mit dem GPS-System und der SQLite-Datenbank, welche standardmäßig auf den Endgeräten installiert sind, eine wichtige Rolle.

Es werden GPS Daten ermittelt und verwertet, sowie Datenbanken erzeugt und deren Inhalt als anwenderfreundliche Dateien ausgegeben.

Schlagwörter: Projektarbeit, Android-Applikation, Datenbanken, Deadlocks, Allergie-Beschwerdebuch

Abstract

The presented student project concerns the development of a allergy diary as an Android application. Therefore the common way of Android application development is explained and applied to the example of the allergy diary.

Some of the most basic and important features that distinguishes an Android smart device are covered by this student project. Especially the handling of the GPS system and the SQLite database which are installed at the devices by default, are playing a major role.

GPS data will be received and proceeded, as well as databases will be generated and its content will be exported as a user-friendly file.

Keywords: Student project, Android application, databases, deadlocks, allergy diary

Inhaltsverzeichnis

Abbildungsverzeichnis	4
1 Einleitung.....	5
2 Problemstellung	6
2.1 Ausgangssituation	6
2.2 Zielsetzung	6
3 Lösungen	8
3.1 Händische Protokollierung.....	8
3.2 Manuelle Protokollierung mittels mobilen Endgeräten	8
3.3 Android-Applikation	8
4 Entwicklung	11
4.1 Allgemeines.....	11
4.2 Analyse	11
4.3 Design	13
4.3.1 Benutzeroberfläche	13
4.3.2 Software	20
4.4 Implementierung.....	23
5 Zusammenfassung.....	30
6 Ausblick	32
Literatur- und Webverzeichnis	33

Abbildungsverzeichnis

Abbildung 3.1: Use-Case Diagramm [1]	12
Abbildung 3.7: MainActivity Klasse [2]	20
Abbildung 3.8: MySQLiteHelper Klasse [3]	21
Abbildung 3.9: PlaceholderFragment Klasse [4]	22
Abbildung 3.10: MyLocationListener Klasse [5]	22

1 Einleitung

In der heutigen Zeit sind mobile Endgeräte wie Smartphones oder Tablet-Computer unsere ständige Wegbegleiter und Unterstützer bei der Lösung und Bewältigung verschiedenster Probleme. Computer und mobile Endgeräte verschmelzen immer mehr zu einer Einheit. Die meisten Geräte verfügen über einen GPS-Empfänger und auf Android-Geräten ist eine SQLite-Datenbank standardmäßig installiert. Die Kombination dieser Features lassen App-Entwicklern eine Menge Freiraum zur Gestaltung neuer Anwendung und zur Lösung alltäglicher Aufgaben, Bereitstellung, verschiedener Dienste und Medien.

Hierzu zählen Programme zur Lösung mathematischer Probleme, Spiele und Unterhaltungsprogramme und Medien aber auch Kommunikations- oder Navigationsdienste.

Selbst aus medizinischer Sicht kann die Anwendung eines Smartphones oder Tablet-Computers sinnvoll sein. So gibt es beispielsweise bereits Programme zur Dokumentation des Blutdrucks oder diverse Anwendungen für Diabetiker.

In dieser Projektarbeit soll ein Dienst für Allergiker entwickelt werden, welcher benutzerfreundlich ist, und vor allem einen brauchbaren Mehrwert darstellt. Dabei soll der Allergie-Patient ein Profil seiner Beschwerden erstellen können und diese brauchbar, aufbereitet seinem Arzt zur Verfügung stellen, damit eine weitere und vor allem gezielte Behandlung vorgenommen werden kann.

Im Folgenden soll die Entwicklung eines Allergie-Beschwerdebuchs als Android-Anwendung erläutert werden. Insbesondere sollen dabei die GPS-Funktion der mobilen Geräte, sowie die SQLite Datenbank zur Anwendung kommen.

2 Problemstellung

2.1 Ausgangssituation

Die Gesellschaft unserer Zeit leidet immer mehr unter Allergien verschiedenster Arten. Die Volkskrankheit Heuschnupfen greift immer mehr um sich und so leidet Schätzungen zufolge jeder sechste Deutsche unter einer Pollenallergie[3]. Doch dabei kann solch eine Allergie vielseitigen Ursprungs sein. Jeder Allergiker reagiert anders auf den Blütenstaub und vor allem, auf andere Arten von Blütenstaub.

So lässt sich, ohne ausgiebige Untersuchung, nur schwer eine passende Behandlung für jeden individuellen Patienten finden, die auch wirklich zur entsprechenden Person passt. Daher bitten Allergologen und Spezialisten ihre Patienten genau zu dokumentieren, welche Symptome auftreten und wo sich diese Person zu dem relevanten Zeitpunkt aufgehalten hatte. Solch ein Allergietagebuch beinhaltet außerdem noch Aufzeichnungen darüber, wie der Patient die Symptome behandelt hat. Mit den gesammelten Daten und Informationen kann sich der Allergologe dann einen Überblick verschaffen und eine entsprechende Diagnose stellen, die hoffentlich zu einer erfolgreichen Behandlung der Allergie führt. Die dabei erhobenen Daten sollten so präzise wie möglich sein, um die Chancen auf eine gute Behandlung zu erhöhen.

Heutzutage besitzt ein Großteil der Bevölkerung ein mobiles Endgerät und führt es ständig mit sich. Ein Android-Gerät mit GPS-Empfänger besitzt alle technischen Voraussetzungen, um eine benutzerfreundliche Anwendung zur Datenerfassung aller relevanten Informationen zu sammeln und festzuhalten, sowie eine verwertbare Auswertung zu erzeugen.

Die Zahl der Menschen, die ein mobiles Endgerät mit sich führen, steigt zudem ständig und somit ist in Zukunft mit einer höheren Anzahl an Android-Applikationsnutzer als potentieller Zielgruppe für ein Allergie-Tagebuch zu rechnen.

2.2 Zielsetzung

Es soll ein Weg gefunden werden, wie ein Patient dazu angehalten ist, durch möglichst bequeme und einfache Bedienung, seine Beschwerdebild aufzuzeichnen. Es sollen die Symptome der Augen, Nase, Mund, Bronchien, Hände und Haut im Grad der Schwere von ganz leicht bis sehr stark protokolliert werden. Ebenso soll die Einnahme von Medikamenten protokolliert werden können. Wichtige Informationen sind in diesem Fall, der Medikamentenname, die Darreichungsform und die eingenommene Menge. Ein zusätzlicher Freitext soll die Möglichkeit bieten weitere Notizen und Informationen zu hinterlegen.

Um den genauen Zeitpunkt der Beschwerden fest zu halten, sollen das aktuelle Datum und die Uhrzeit konfigurierbar dargestellt werden, damit auch im Nachhinein Einträge

2 Problemstellung

erstellt werden können. Standortdaten sollen erfasst und dargestellt werden und ebenfalls konfigurierbar sein.

Um die Daten konsistent und persistent verfügbar zu haben, müssen alle Eingaben in einem Datensatz abgelegt werden, welcher zur weiteren Verarbeitung gut geeignet ist und dem behandelnden Arzt in vernünftiger Form vorgelegt werden kann. Bestenfalls ist der Gesamtdatensatz am Ende auch noch editierbar und kann händisch um weitere Einträge erweitert werden.

Bei einer Netto-Arbeitszeit von 150 Stunden, werden auf optische- und grafische-Feinheiten keinen besonderen Wert gelegt, sondern rein die Funktion und vor allem Benutzerfreundlichkeit stehen im Vordergrund. Auch auf eine detaillierte Kartenansicht zur Darstellung der Standort Daten muss zunächst verzichtet werden, da dies den Rahmen sprengen würde.

3 Lösungen

3.1 Händische Protokollierung

In der Regel notieren sich Patienten bislang ihre Symptome auf einem Notizblock und legen dieses Dokument letztendlich ihrem Arzt vor. Natürlich erfordert diese Methode am wenigsten technischen Einsatz und ist für jeden Patienten sofort durchführbar, ohne große Anschaffungen. Diese einfache Vorgehensweise hat aber den Nachteil, dass man ständig seinen Notizblock und Schreibzeug mit sich führen müsste und dies im entscheidenden Moment nicht zur Hand ist und ein Eintrag vergessen oder vernachlässigt wird.

Oft stellt sich mit dieser Methode schon nach kurzer Zeit eine Nachlässigkeit ein und letztendlich wird überhaupt keine Protokollierung mehr vorgenommen. Zudem sollte ein Patient immer wieder daran erinnert werden, sein Allergie-Beschwerdebuch weiter zu führen und konsequent zu pflegen. Des Weiteren besteht die Gefahr, dass das Tagebuch verloren wird.

Für einen konsequenten und sorgfältigen Patienten ist diese Methode eventuell ausreichend. Dennoch ist die Gefahr zu groß, dass das Tagebuch nicht ausreichend gepflegt wird und somit nicht genügend verwertbare Daten am Ende zur Verfügung stehen, die der Arzt benötigt.

3.2 Manuelle Protokollierung mittels mobilen Endgeräten

Oftmals werden die Allergie-Beschwerdebücher auch einfach mittels Texteditor auf einem mobilen Endgerät geführt. Dies hat im Gegensatz zur Protokollierung mit Stift und Papier den Vorteil, dass die Endgeräte meist in Reichweite sind und nicht immer extra mitgeführt werden müssen. Man ist eher dazu angehalten sich Notizen zu machen, da die Geräte einfach öfter zur Hand sind. Fehleingaben können leichter und schneller korrigiert werden.

Dennoch führt diese Vorgehensweise schnell zu einem sehr unübersichtlichen und nur schwer verwertbaren Datensatz, welcher auch oftmals nur auf dem mobilen Endgerät editiert und gelesen werden kann. Außerdem muss auch hier der Standort eventuell erst über Kartenmaterial ermittelt werden und kann unter anderem nicht besonders präzise notiert werden. Die formlose Eingabe der Daten führt zu einem uneinheitlichen Datengemenge.

3.3 Android-Applikation

Im Gegensatz zur händischen Protokollierung erfordert ein Lösungsansatz als Android-Applikation den Einsatz von technischem Equipment. Der große Vorteil ist, dass heutzutage ein Großteil der Bevölkerung ständig ein Smartphone oder Tablet mit sich führt

und somit nicht immer an zusätzliche Utensilien gedacht werden muss, sobald der Patient das Haus verlässt. Die am weitesten verbreitete Plattform für die mobilen Endgeräte ist das Betriebssystem Android von Google. Eine einfache Applikation zu entwickeln, welche auf einem Android-Betriebssystem basiert ist relativ schnell zielführend. Ausgehend von der bereits standardmäßig installierten SQLite-Datenbank und dem GPS-System bringt ein Android-Gerät meist alle erforderlichen Hard- und Software-Voraussetzungen mit sich, die für eine bequeme und schnelle Bedienung eines Allergie-Tagebuchs und dessen Entwicklung erforderlich sind.

[1]. Die SQLite-Datenbank, die für den Einsatz auf mobilen Endgeräten optimiert wurde, unterstützt bei einem Minimum an Speicherplatzbedarf, fast alle Funktionen ähnlicher Datenbanksysteme, die auf großen Server ihre Anwendung finden.

SQLite grenzt sich durch folgende Merkmale von anderen Datenbanksystemen ab:

- **Keine Installation, keine zentrale Konfiguration** Bei der SQLite handelt es sich um ein serverloses Datenbanksystem, welches ohne zusätzlichen Serverprozess auskommt.
- **Zugriffe aus mehreren Anwendungen erlaubt** Die zeitgleiche Nutzung einer Datenbank durch mehrere Anwendungen ist erlaubt.
- **Eine Datenbank = eine Datei** Es wird pro Datenbank nur eine Datei im Dateisystem von Android angelegt. Alle Datenbanken einer Anwendung sind im Verzeichnis `/data/data/package.name/databases` zu finden. Der Zugriff auf die Datenbank im Datei-Explorer der Geräte kann mitunter schwierig sein, da der Benutzer über Administrationsrechte auf dem Root-Verzeichnis des Gerätes verfügen muss. Als `package.name` muss der Paketname der Anwendung verwendet werden.
- **Dynamische Feldlänge** Bei Textfeldern nutzt SQLite immer die tatsächliche Länge des Feldes zum Speichern aus und ignoriert dabei die definierte Feldlänge.

Um einen konsistenten und persistenten Datensatz zu erzeugen und verwalten zu können, ist die SQLite Datenbank gut geeignet. Hier kann für jeden Eintrag eine Zeile erzeugt und die entsprechenden Daten abgelegt werden. Diese Daten sind für den Nutzer zunächst nicht sichtbar, jedoch sicher gespeichert.

Um mit dem erzeugten Datensatz weiter arbeiten zu können, soll eine Export-Funktion eine CSV-Datei mit den entsprechenden Daten erzeugen. Diese Excel-Datei kann dann auf einem Computer entsprechend editiert und ausgedruckt werden, um sie dem Arzt zur Verfügung zu stellen.

Jede Tabellenzeile der exportierten Excel-Datei hat folgende Einträge:

- Art und Schwere der Symptome
- Datum und Uhrzeit, wann die Symptome aufgetreten sind

3 Lösungen

- Den Namen, die Form und die Menge der eingenommenen Medikamente gegen die Symptome, sowie dem Freitext, welcher weitere Notizen zur Sachlage beinhaltet
- Längen- und Breitengrad, wo die entsprechenden Symptome des Patienten aufgetreten sind

Die exportierten Daten erfüllen den Anspruch auch im Nachhinein noch editierbar und erweiterbar zu sein und um falsche oder vergessene Einträge nachträglich zu ändern oder hinzuzufügen.

Mit dem eingebauten GPS-Empfänger auf dem Android-Gerät ist es ein Leichtes, in kürzester Zeit, den Standort sehr genau zu bestimmen und zur weiteren Verarbeitung bereitzustellen. Mit den Koordinaten für Längen- und Breitengrad kann auch im Nachhinein mittels Online-Kartendienst der genaue Standort nachvollzogen und so eventuell sogar ein Kartenprofil gespeichert werden, wo die Beschwerden am häufigsten aufgetreten sind.

Der Name einiger Medikamente, welche häufig eingenommen werden, können im System hinterlegt werden und somit gleich als Standardname hinterlegt sein, sobald der Patient die Applikation öffnet. Durch eine Alarmfunktion kann der Patient sogar ständig dazu angehalten werden, seine Medikamente, zu von ihm definierten Zeiten, einzunehmen.

Die Möglichkeiten eines Allergie-Beschwerdebuchs als Android-Anwendung, sind vielfältig und zahlreich. Solch eine Applikationsentwicklung ist daher fast hauptsächlich durch zeitliche oder organisatorische Faktoren eingeschränkt.

4 Entwicklung

4.1 Allgemeines

[1] Zur Entwicklung eines Android-Projekts wird zunächst Eclipse mit einem Android-Plug-In benötigt. Zu Beginn wird als erstes ein neues Projekt erzeugt, die sogenannte Activity. Activities implementieren die Bildschirmmasken einer Android-Anwendung. Die darstellbaren Elemente einer Seite sind die so genannten Views. Diese Views sind in XML definiert. Alle ausführbaren Aktionen werden in Java-Klassen implementiert.

Nach Beendigung des Eclipse-Projekt-Wizards findet man folgende Projektstruktur für Android-Projekte vor:

- src: Java-Quelltexte
- res: Ressourcen, d.h. alle Nicht-Java-Texte wie zum Beispiel Bibliotheken. Hier werden unter anderem Dateien zur Definition der Oberflächen, Bilder oder Textdefinitionen abgelegt. Die AndroidManifest.xml zur Definition von Rechten der Applikation und einigen Metadaten.
- bin: .apk Datei, welche die ausführbare Datei auf den Geräten ist
- libs: mit der android-support.jar und eventuell weiteren, speziell für die Anwendung benötigten Libraries.

4.2 Analyse

Aus der bereits erläuterten Zielsetzung ergeben sich folgende Anforderungen an die Applikation:

- Informationen über den Grad der Symptome für Augen, Nase, Mund, Bronchien, Hände und Haut
- Die Medikamenteneinnahme protokollieren
- Das aktuelle Datum und Uhrzeit ermitteln und editierbar darstellen
- Den Standort über GPS ermitteln
- Sämtliche Daten in der SQLite Datenbank speichern
- Daten als CSV-Datei exportieren
- Einen Hilfetext anzeigen, welcher dem Nutzer die Funktionen der Applikation erklärt

Um einen groben Überblick über die Anforderung der Software zu gewinnen, wurde Use-Case-Diagramm erstellt. Hier werden alle Aktionen, die der Patient durchführen kann, grob dargestellt. Jede dieser Aktionen stellt später ein Menüpunkt dar und wird so in der Software abgebildet.

Zuerst kommt die Eingabe der Beschwerden für Augen, Nase, Mund, Bronchien, Hände und Haut in fünf Stufen. Der zweite Menüpunkt ist die Dokumentation der Medikamenteneinnahme. Dieses Menü hat die vier Aktionen der Eingabe des Medikamenten-

namens, der Form des Medikaments, sowie die Menge und der Möglichkeit einer Kurznotiz. Der dritte Menüpunkt ist die Erfassung der Systemzeit und des Datums. Diese sind jeweils editierbar. Als vierter Menüpunkt dient die Ermittlung der aktuellen Position. Alle erfassten Daten werden durch Aufruf des fünften Menüpunkt in der Datenbank gespeichert und sind somit zur weiteren Verarbeitung verfügbar. Der sechste Menüpunkt exportiert alle Daten aus der Datenbank, indem alle Daten aus der Datenbank erfasst werden und daraus die CSV-Datei erzeugt wird, welche unter dem Hauptverzeichnis des Geräts abgelegt wird. Der siebte und letzte Menüpunkt soll den Hilfetext anzeigen um dem Nutzer eine kurze Erklärung zur Funktion der Anwendung bereit zu stellen.

Der Sachverhalt ist in dem Use-Case-Diagramm in Abbildung 3.1 zusammengefasst.

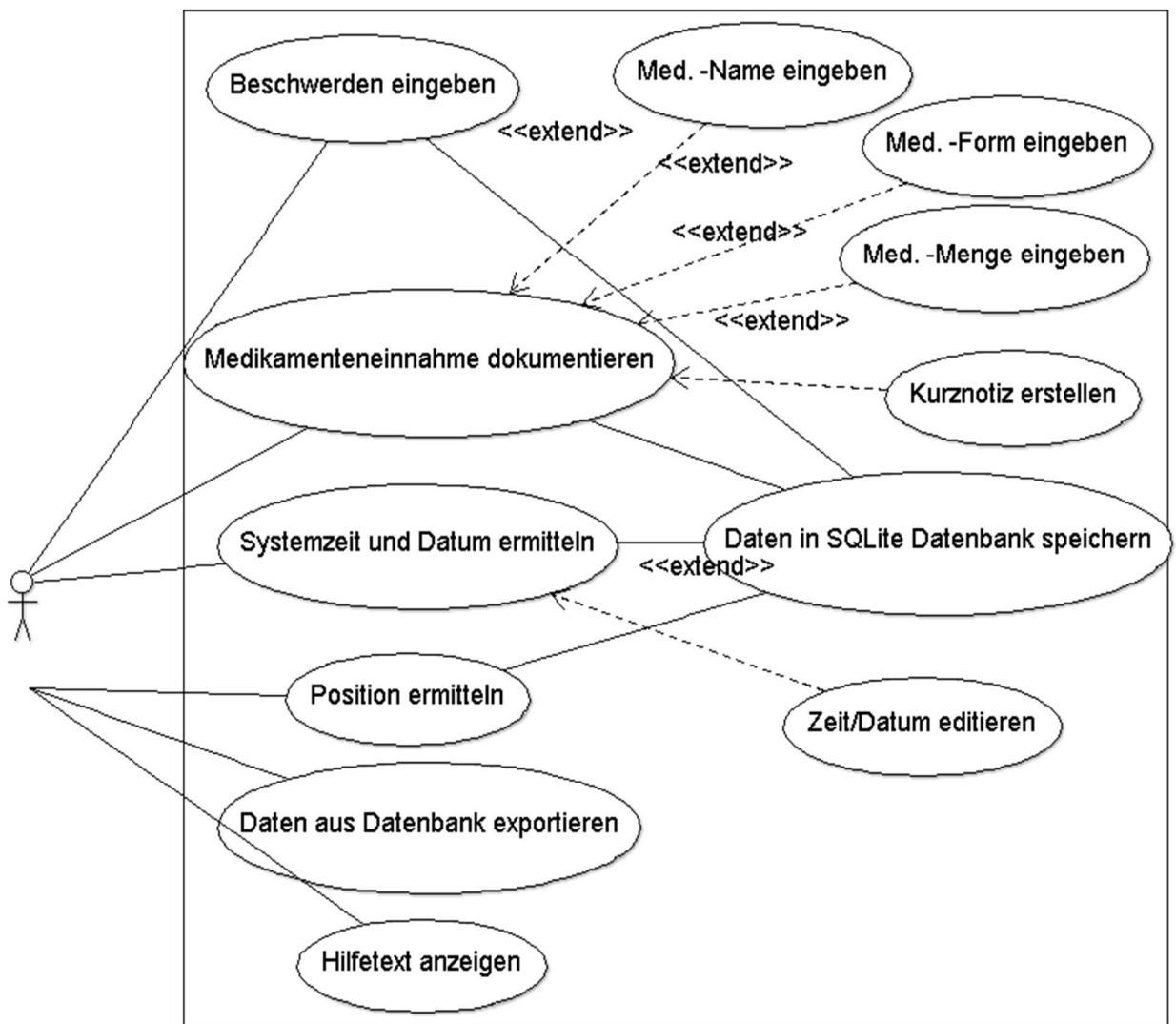


Abbildung 3.1: Use-Case Diagramm [1]

4.3 Design

4.3.1 Benutzeroberfläche

Die Layouts der Benutzeroberflächen werden per XML definiert. In Eclipse steht zudem noch ein What-You-See-Is-What-You-Get-Editor zur Verfügung, mit dem man sich die Oberfläche nach Bedarf erstellen und anpassen und im XML-Editor nochmals modifizieren kann.

Als erstes wird ein Layout erstellt, welches als Container für alle weiteren Layouts dient. Dieses Container-Layout kann schon vordefinierte Elemente enthalten, welche in jedem weiteren Layout zur Verfügung stehen sollen. In unserem Fall dient es als leerer Container und soll deshalb nicht weiter betrachtet werden.

The screenshot shows a mobile application titled "Allergie - Beschwerdebuch". It features a list of body regions with corresponding radio buttons for rating from 1 to 5. The regions and their current ratings are: Augen (1), Nase (1), Mund (1), Bronch. (1), Hände (1), and Haut (1). The radio buttons are arranged horizontally next to each region name.

Um die Symptome aller Körperregionen gut bewerten zu können, wurde für jede Körperregion eine Radio-Group erstellt, mit der man von 1 = ganz leicht, bis 5 = sehr stark ausgeprägte Symptome beziehungsweise seine Beschwerdebild festhalten kann (siehe Abbildung 3.2). Die Bezeichnungen und Überschriften der Layouts werden in einer string.xml Datei abgelegt, damit man bei einer mehrsprachigen Anwendung keine neuen Layouts erstellen, sondern nur die Stringtable erweitern muss.

Abbildung 3.2: Benutzeroberfläche zur Eingabe der Beschwerden [2]

Der folgende Ausschnitt der XML Datei zeigt das Layout aus Abbildung 3.2:

```
<RadioGroup
    android:id="@+id/main_eyes_radioGroup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:orientation="horizontal" >

    <RadioButton
        android:id="@+id/main_eyes_radio1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
```

```
        android:text="@string/one" />

        <RadioButton
            android:id="@+id/main_eyes_radio2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/two" />

        <RadioButton...

    .../>
</RadioGroup>
```

Jeder `RadioButton` gehört zu einer `RadioGroup`. Durch die `android:id` kann jedes Element im Java Quellcode verwendet werden. Die `android:layout` Tags dienen zur Positionierung der Elemente in der Benutzeroberfläche. Die `RadioGroup` mit der `android:id="@+id/main_eyes_radioGroup"` wird oben rechts angefügt. Folgende `RadioGroups` orientieren sich immer an der Übergeordneten. Also fügt sich die folgende `RadioGroup` an die `main_eyes_radioGroup` an:

```
<RadioGroup
    android:id="@+id/main_nose_radioGroup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/main_eyes_radioGroup"
    android:layout_below="@+id/main_eyes_radioGroup"
    android:orientation="horizontal" >
```

Die Layout-Parameter `width` und `height` spezifizieren die Größe der einzelnen Elemente. Das Attribut `wrap_content` bedeutet, dass das Element gerade so groß sein soll, dass seine beinhalteten Elemente eingeschlossen werden können. Mit dem Tag `android:orientation` kann die Ausrichtung des Elements bestimmt werden, in diesem Fall ist sie horizontal.

Der Tag `android:text` ist eine Referenz auf die `string.xml`, wo vordefinierte Strings abgelegt sind. Ein kleiner Ausschnitt aus unserer Stringtable sieht folgendermaßen aus:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Allergie - Beschwerdebuch</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="header">Allergie - Beschwerdebuch</string>
    <string name="title_activity_date">DateActivity</string>
    <string name="title_activity_date_time">DateTimeActivity</string>
    <string name="title_activity_position">PositionActivity</string>
    <string name="title_activity_medicine">MedicineActivity</string>
    <string name="one">1</string>
    <string name="two">2</string>
    <string name="three">3</string>
    <string name="four">4</string>
    <string name="eyes">Augen</string>
```

```

<string name="nose">Nase</string>
<string name="mouth">Mund</string>
<string name="bronchia">Bronchien</string>
  <string name="hands">Hände</string>
<string name="skin">Haut</string>
<string name="save">Speichern</string>
...
</resources>

```

Neben der Möglichkeit eine mehrsprachige Anwendung zu erstellen, hat ein Stringtable auch den Vorteil, dass jeder String nur einmal definiert werden muss. Haben alle Benutzeroberflächen der Anwendung dieselbe Überschrift und diese ändert sich, so muss nur ein String geändert werden, und nicht jedes einzelne Layout der Anwendung.

Datum und Uhrzeit werden mit dem DatePicker und TimePicker erfasst. Die angezeigten Zeiten können dabei leicht durch verschieben der Scrollbars verändert werden. Beim Aufruf des Menüs der Datum- und Uhrzeiteingabe werden automatisch das aktuelle Datum und die aktuelle Zeit voreingestellt. Deshalb darf nach Eingabe einer editierten Zeitangabe das Menü nicht noch einmal aufgerufen werden, bevor alle Daten abgespeichert wurden. Da dieses Menü für ein 4 Zoll Display zu groß ist, wurde es in einem ScrollView Tag erstellt. Dadurch kann das Layout in der Vertikalen verschoben werden.

Das Layoutattribut `fill_parent` bedeutet, dass das Element so groß wie sein Parent-Element sein soll.

Abbildung 3.3: Benutzeroberfläche zur Erfassung von Datum und Uhrzeit [3]

```
<ScrollView
```

```

    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">


        <TextView
            android:id="@+id/date_time_time_textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/time"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <TimePicker
            android:id="@+id/timePicker1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <TextView
            ...
        />
    </LinearLayout>
</ScrollView>

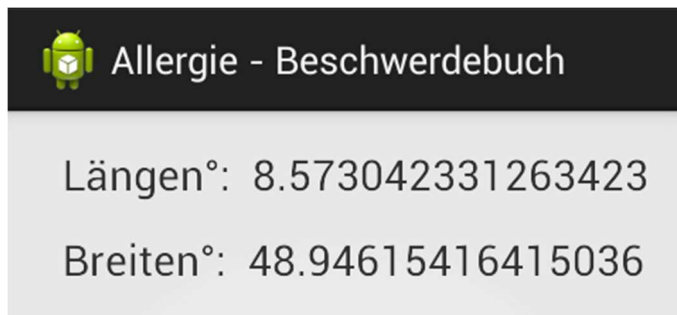
```

Das `android:orientation` Tag im `LinearLayout` erzeugt die vertikale Scrollrichtung der Oberfläche. In der `TextView` sind neben den bereits erläuterten Layout-Ausrichtungen und String-Zuweisungen, durch das Tag `android:textAppearance`, auch die Schriftart und Größe spezifiziert. Für den `DatePicker` und `TimePicker` sind nur die Bestimmung von Id und Ausrichtung im Layout notwendig.



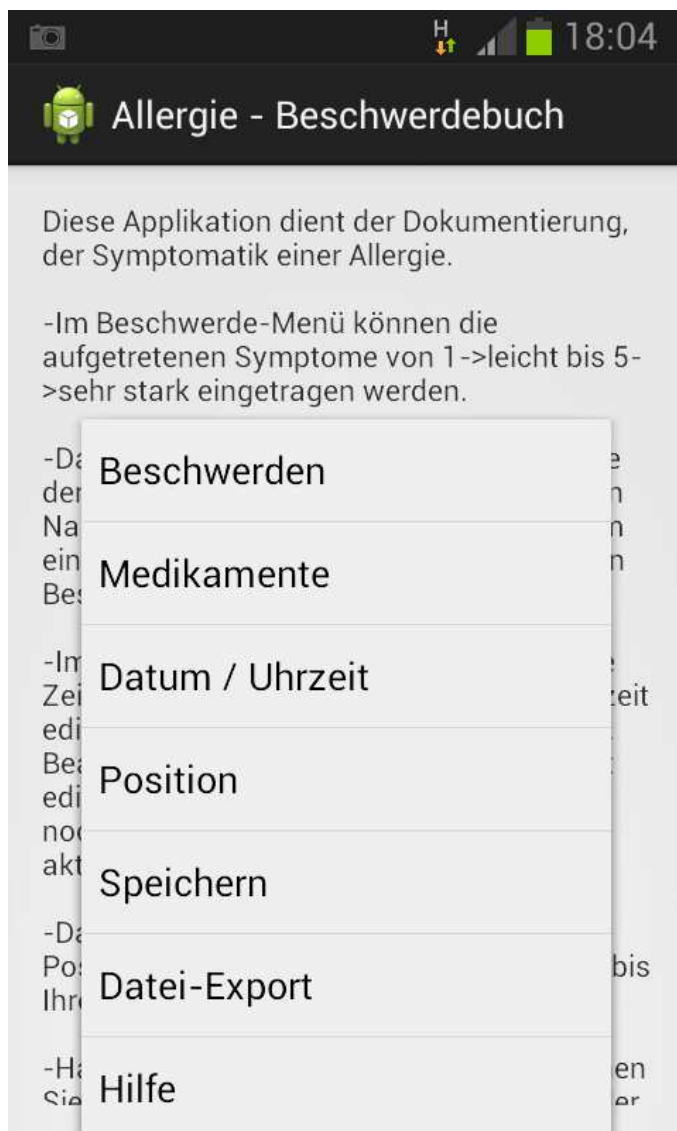
Die Benutzeroberfläche zur Erfassung der eingenommenen Medikamente und der Kurznotiz, besteht nur aus einigen Text- und Editfeldern. Editfelder werden prinzipiell genauso wie Textfelder erstellt. Über die Id des wird der eingegebene Text im Code ausgelesen.

Abbildung 3.4: Benutzeroberfläche zur Erfassung der Medikamenten-Einnahme [4]



Auch zur Erfassung der Längen- und Breitengrade werden nur Textfelder benötigt. Hier werden die Textfelder, die die Dezimalwerte enthalten, mittels `android:id` zur Laufzeit mit den GPS-Daten beschrieben.

Abbildung 3.5: Benutzeroberfläche zur Erfassung der Positionsdaten [5]



Im letzten Menüpunkt wird der Hilfetext für den Anwender des Allergie-Beschwerdebuchs angezeigt. Dieses Layout besteht nur aus einem Textfeld in einer Scrollview. Die Menüpunkte Speichern und Datei-Export haben keine eigenen Layouts. In der Abbildung 3.6 ist auch die Menüleiste sichtbar. Das Menü ist unter Projektordner/`res/layout` als `menu.xml` abgelegt und spezifiziert. Die Menüpunkte werden in der Reihenfolge, wie sie in der `menu.xml` erstellt wurden, auch aufgelistet. Jeder Menüeintrag wird wieder über die `android:id` im Quellcode angesprochen. Zur Benennung der Menüpunkte wird ein String in der Stringtable hinterlegt. Je nach Art des Menüs kann jeder Menüpunkt auch durch einen Icon dargestellt werden. In diesem Fall muss jeweils ein Icon im Ordner `drawable` hinterlegt werden.

Abbildung 3.6: Hilfetext und Menüliste [6]

Das Menü ist wie folgt spezifiziert:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- Single menu item
    Set id, icon and Title for each menu item
  -->
  <item android:id="@+id/menu_symptoms"
    android:icon="@drawable/icon_main"
    android:title="@string/menu1" />

  <item android:id="@+id/menu_medicine"
    android:icon="@drawable/icon_main"
    android:title="@string/menu2" />

  <item android:id="@+id/menu_date"
    android:icon="@drawable/icon_date"
    android:title="@string/menu3" />
  ...
</menu>
```

[2] Jede Anwendung muss eine AndroidManifest.xml-Datei im Root-Verzeichnis haben. Die Manifest-Datei liefert dem Betriebssystem wichtige Informationen über die Anwendung, bevor die Anwendung ausgeführt werden kann. Außerdem ist die Manifest.xml für folgendes zuständig:

- Es benennt das Java-Package der Anwendung
- Es beschreibt die Komponenten der Anwendung
- Es deklariert welche Rechte die Anwendung haben muss
- Es beschreibt welche Rechte andere Anwendungen haben müssen, um mit der Anwendung agieren zu dürfen
- Es beinhaltet die Programmbibliotheken, auf die, die Anwendung Zugriff haben muss

Im Folgenden einige Ausschnitte der Manifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.allergietb"
  android:versionCode="1"
  android:versionName="1.0" >

  <uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="19" />

  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

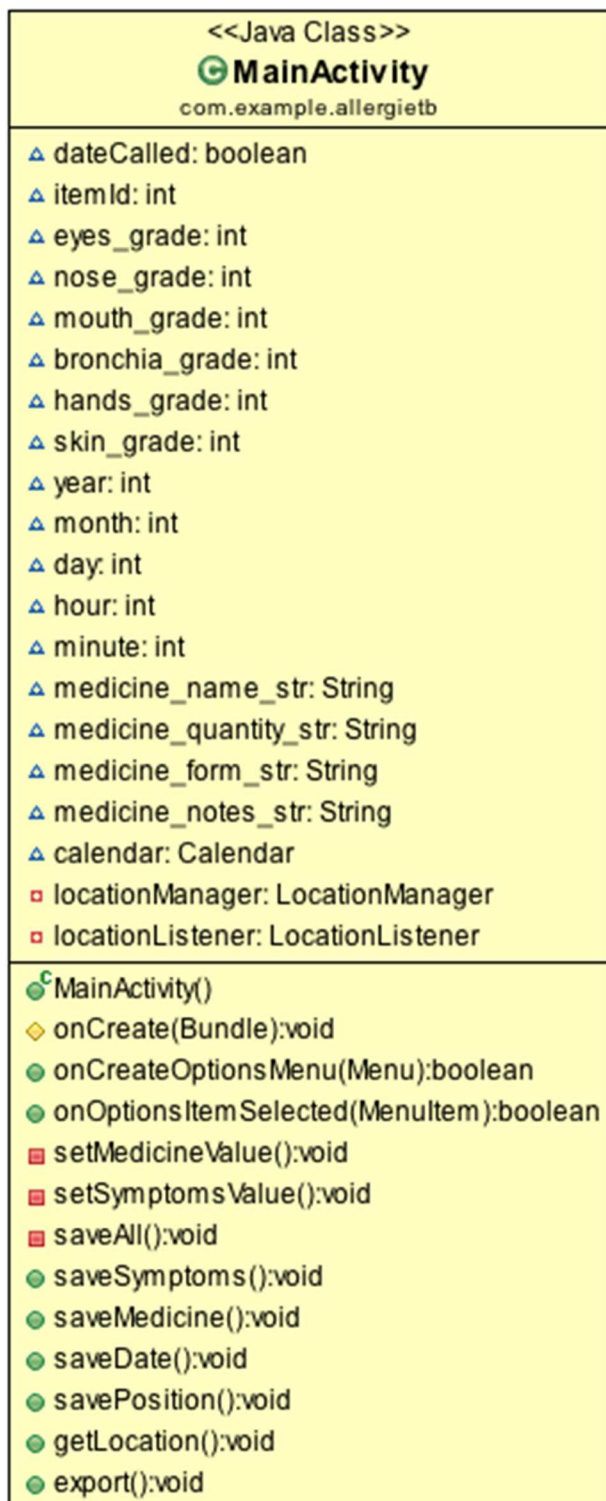
  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
```

```
    android:theme="@style/AppTheme" >
    <activity
        android:name="com.example.allergietb.MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
</manifest>
```

Zunächst werden der Package-Name und die Versionen der Anwendung und des verwendeten Software-Development-Kit festgelegt. In den Tags `uses-permission` holt sich die Anwendung einige Rechte aus dem System. Dies sind die Rechte das Internet und die GPS-Funktionen des Gerätes zu nutzen sowie auf den Speicher des Gerätes schreiben zu dürfen. Im `application` Tag werden Namen, Style und Icon der Anwendung bestimmt. Unsere Activity "MainActivity" ist eine Unterklasse von "Activity" und muss in der Manifest.xml deklariert werden. In dem Tag `Intent-filter` wird die MainActivity gestartet.

4.3.2 Software



-db
0..1

Die wichtigsten Methoden unserer Anwendung sind in der MainActivity implementiert. Es wird ein Calendar-Objekt erstellt um Datum und Uhrzeit zu erhalten. Die LocationManager- und LocationListener-Objekte werden beim Starten der Anwendung erstellt, um den Zugriff auf die GPS-Daten zu erhalten. Bei der MainActivity() handelt es sich um einen Standardkonstruktor. Beim Starten der Anwendung wird die onCreate(Bundle):void Methode aufgerufen und dadurch die Anwendung initialisiert und eine erste View geladen und angezeigt. onCreateOptionsMenu(Menu):boolean erzeugt die Menüleiste und gibt true oder false zurück, je nachdem ob das Menü erfolgreich erstellt werden konnte.

Abbildung 3.7: MainActivity Klasse [2]

Die Methode onOptionsItemSelected(MenuItem):boolean ist der Handler der Menüleiste. Die Methode bekommt ein Menu-Item übergeben und weiß somit welches Menü gewählt wurde. Alle drei genannten Methoden werden standardmäßig beim Erzeugen des Projekts erstellt. Die Methoden saveSymptoms():void, saveMedicine():void, saveDate():void und savePosition():void sollen die Werte zwischen speichern die in den

entsprechenden Views eingetragen wurden, um sie nach Verlassen der View und erneutem Aufruf mit `setMedicineValue():void` `setSymptomsValue():void` in die View einzutragen. Somit gehen die Daten nicht durch blättern im Menü verloren.

`getLocation():void` soll die aktuellen GPS-Daten ermitteln und `saveAll():void` schreibt alle erhobenen Daten in die Datenbank. Letztendlich werden alle gespeicherten Daten mit `export():void` in eine csv-Datei geschrieben und auf dem Datenspeicher abgelegt. Wie in dem Klassendiagramm der Abbildung 3.7 ersichtlich ist, wird in der MainActivity eine Datenbank erzeugt.

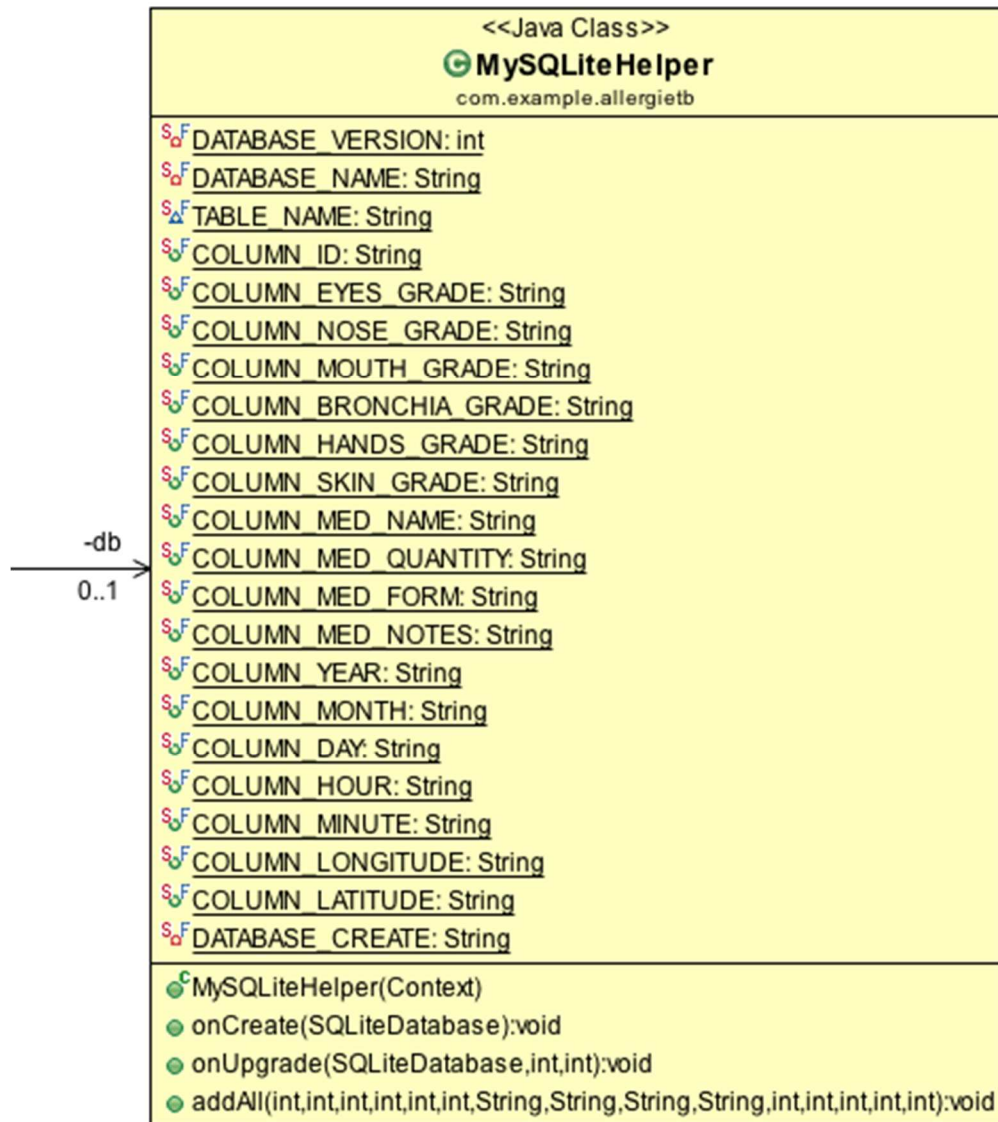


Abbildung 3.8: MySQLiteHelper Klasse [3]

Nach erfolgreichem erzeugen des Datenbankobjekts durch den Standardkonstruktor `MySQLiteHelper(Context)`, wird `onCreate(SQLiteDatabase):void` aufgerufen, falls noch keine Datenbank vorhanden ist. In diesem Fall wird nun die Datenbank erstellt. Falls es schon eine Datenbank gibt, wird mit `onUpgrade(SQLiteDatabase, int, int):void` diese erweitert bzw. erneuert. Mit dem Aufruf der `addAll(int, int, int, int, int, int, String, String, String, String, int, int, int, int, int):void`,

String, String, int, int, int, int, int):void werden sämtliche Daten in der Datenbank abgelegt.

Die PlaceholderFragment Klasse ist eine Unterklasse der Fragment-Klasse und dient zur Erstellung der Benutzeroberflächen zur Laufzeit. Die Fragment-Klassen sind sehr eng verknüpft mit den Activity-Klassen, in denen sie verwendet werden und können auch nur in deren Verbindung benutzt werden.



Abbildung 3.9: PlaceholderFragment Klasse [4]

Neben dem Standardkonstruktor der PlaceholderFragment Klasse, benötigen wir die onCreateView(LayoutInflater, ViewGroup, Bundle):View Methode. Diese erzeugt eine View-Hierarchy und gibt diese View zurück. Das heißt, hier wird das Container-Layout mit dem im Menü selektierten Layout gefüllt und zurückgegeben.

Die Klasse MyLocationListener implementiert das Interface LocationListener. Die Klasse wird vom LocationManager benachrichtigt, wenn bestimmte Ereignisse eintreten.



onLocationChanged(Location):void wird aufgerufen, wenn eine neue GPS-Position ermittelt wurde. Als Parameter wird ein Location-Objekt übergeben, welches die Positionsangabe als Längen- und Breitengrad beinhaltet.

Abbildung 3.10: MyLocationListener Klasse [5]

Die Methoden onProviderDisabled(String):void und onProviderEnabled(String):void werden aufgerufen, wenn das GPS des Gerätes deaktiviert oder aktiviert wird. onStatusChanged(String, int, Bundle):void wird aufgerufen, wenn sich der Status des GPS-Providers ändert.

Mit diesen vier Klassen steht das Grundgerüst für das Allergie-Beschwerdebuch. Somit sind alle Funktionen vorhanden, die in der Anforderung erhoben wurden. Aus den Klassendiagrammen kann nun der Quellcode erstellen und mit der Implementierung begonnen werden.

4.4 Implementierung

Zunächst soll ein leeres Grundgerüst mit den vier Klassen und allen Methoden aus dem Design erstellt werden. Die Imports und Variablendeklarationen werden aus Platzgründen hier nicht aufgeführt.

```
package com.example.allergietb;

public class MainActivity extends ActionBarActivity {

    Calendar calendar = Calendar.getInstance();

    /**LocationListener declarations*/
    private LocationManager locationManager=null;
    private LocationListener locationListener=null;
    private MySQLiteHelper db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {}

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        switch (item.getItemId())
        {
            case R.id.menu_symptoms:
return true;

            case R.id.menu_medicine:
return true;
            ...
        }
    }

    private void setMedicineValue() {}

    private void setSymptomsValue() {}

    private void saveAll() {}

    public void saveSymptoms() {}

    public void saveMedicine() {}

    public void saveDate() {}

    public void savePosition() {}

    public void getLocation(){ }

    public void export(){ }
}
```

Die **protected void** onCreate(Bundle savedInstanceState) Methode, zeigt die erste View auf dem Display des Geräts und erzeugt über den Aufruf des PlaceholderFragment die View-Hierarchy, um das Container-Layout zu füllen. Außerdem werden die Datenbank und der LocationListener erzeugt.

```
setContentView(R.layout.activity_main);

/**Database declarations*/
db = new MySQLiteHelper(this);
db.getWritableDatabase();
locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

if (savedInstanceState == null) {
    getSupportFragmentManager().beginTransaction()
        .add(R.id.container, new PlaceholderFragment())
        .commit();
}
```

In der onCreateOptionsMenu(Menu menu) Methode wird die Menüleiste erzeugt und mit den bereits erstellten Menüpunkten aus der menu.xml befüllt.

```
getMenuInflater().inflate(R.layout.menu, menu);
```

Um nun den Handler der Menüleiste zu erstellen, wird die onOptionsItemSelected(Menuitem item) Methode benötigt. Die Methode wird automatisch bei betätigen der Menüleiste ausgelöst und bekommt als Menu-Item den Menütyp übergeben, welcher betätigt wurde. Mit einer switch-case –Anweisung werden dann die entsprechenden Aktionen ausgeführt. Es kann beispielsweise eine neue View angezeigt oder eine neue Activity gestartet werden. Wir betrachten hier zunächst nur den Aufruf des Menüs zur Symptomeingabe.

```
switch (item.getItemId())
{
    case R.id.menu_symptoms:
        Toast.makeText(MainActivity.this, "Beschwerden is Selected",
        Toast.LENGTH_SHORT).show();
        setContentView(R.layout.fragment_main);
        setSymptomsValue();
        return true;

    case R.id.menu_medicine:
        ...
    case R.id.menu_date:
        ...
    case R.id.menu_position:
        ...
    case R.id.menu_export:
        ...
    case R.id.menu_help:
        ...
    case R.id.menu_save:
        ...
}
```



```
}
```

Zunächst wird bei jedem Menü-Aufruf eine kleine Textmeldung ausgegeben, um zu sehen, in welchem Menüpunkt man sich befindet. Anschließend wird die entsprechende View geladen und über `setSymptomsValue()` die View mit den alten Werten initialisiert. Dies funktioniert, indem die alten Werte der View, beim Aufruf eines anderen Menüpunktes mit `saveSymptoms()` zwischengespeichert werden.

```
RadioGroup rg_eyes = (RadioGroup)findViewById(R.id.main_eyes_radioGroup);
RadioGroup rg_nose = (RadioGroup)findViewById(R.id.main_nose_radioGroup);
...
if(rg_eyes.getCheckedRadioButtonId() == R.id.main_eyes_radio1)eyes_grade = 1;
else if(rg_eyes.getCheckedRadioButtonId() == R.id.main_eyes_radio2)eyes_grade = 2;
else if(rg_eyes.getCheckedRadioButtonId() == R.id.main_eyes_radio3)eyes_grade = 3;
else if(rg_eyes.getCheckedRadioButtonId() == R.id.main_eyes_radio4)eyes_grade = 4;
else if(rg_eyes.getCheckedRadioButtonId() == R.id.main_eyes_radio5)eyes_grade = 5;
...
```

Diese Werte werden dann beim erneuten Aufruf der Symptoms-View neu geladen.

```
RadioGroup rg_eyes = (RadioGroup)findViewById(R.id.main_eyes_radioGroup);
RadioGroup rg_nose = (RadioGroup)findViewById(R.id.main_nose_radioGroup);
...
if(eyes_grade == 1)rg_eyes.check(R.id.main_eyes_radio1);
else if(eyes_grade == 2)rg_eyes.check(R.id.main_eyes_radio2);
else if(eyes_grade == 3)rg_eyes.check(R.id.main_eyes_radio3);
else if(eyes_grade == 4)rg_eyes.check(R.id.main_eyes_radio4);
else if(eyes_grade == 5)rg_eyes.check(R.id.main_eyes_radio5);
...
```

Die Abfragen der weiteren Eingabemasken sind analog zu der, der Symptom-View.

In der `getLocation()` Methode wird der `locationListener` erzeugt. Dieser wird benötigt um vom `LocationManager` benachrichtigt zu werden, wenn eine Positionsänderung per GPS detektiert wurde. Dafür muss der `locationListener` noch beim `locationManger` registriert werden. Den Zugriff zum GPS des Gerätes bekommt die Anwendung über die `locationManager` Klasse.

```
locationListener = new MyLocationListener();
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000,
10,locationListener);
```

Mit `saveAll()` werden alle Werte in die Datenbank geschrieben.

```
db.addAll(eyes_grade, nose_grade, mouth_grade, bronchia_grade, hands_grade,
skin_grade, medicine_name_str, medicine_quantity_str, medicine_form_str,
medicine_notes_str, year, month, day, hour, minute);
```

Die letzte wichtige Methode der `MainActivity` ist die `export()` Methode. Um die Export-Funktion nutzen zu können, muss zuerst die `opencsv.jar` Datei dem `BuildPath` von Eclipse hinzugefügt werden, um einige Funktionen nutzen zu können. In der Export-Methode wird zuerst eine csv-Datei mit dem Namen "Allergie-Beschwerdebuch" im

Hauptverzeichnis des Gerätes erstellt. Ein Cursor geht anschließend iterativ alle Reihen der Datenbank durch und speichert die darin vorhandenen Werte mit einer while-Schleife, in der csv-File ab. Fehler werden dabei in einer Exception abgefangen.

```
File exportDir = new File(Environment.getExternalStorageDirectory(), "");

if (!exportDir.exists())
{ exportDir.mkdirs();
}

File file = new File(exportDir, "Allergie-Beschwerdebuch.csv");
try
{
    file.createNewFile();
    CSVWriter csvWrite = new CSVWriter(new FileWriter(file));
    SQLiteDatabase dbSql = db.getReadableDatabase();
    Cursor curCSV = dbSql.rawQuery("SELECT * FROM "
+MySQLiteHelper.TABLE_NAME, null);
    csvWrite.writeNext(curCSV.getColumnNames());
    while(curCSV.moveToNext())
    {
        //Which column you want to export
        String arrStr[] = { curCSV.getString(0),
                           curCSV.getString(1),
                           curCSV.getString(2),
                           curCSV.getString(3),
                           ...
                           curCSV.getString(16)};
        csvWrite.writeNext(arrStr);
    }
    csvWrite.close();
    curCSV.close();
}
catch (Exception sqlEx)
{
    Log.e("MainActivity", sqlEx.getMessage(), sqlEx);
}
}
```

Die Klasse MyLocationListener ist sehr überschaubar. Sie implementiert die Location-Listener Klasse und besteht im Wesentlichen aus der onLocationChanged (Location Loc) Methode, welche bei Positionsänderung die neuen Koordinaten übergeben bekommt. Zwei TextView-Objekte müssen erstellt werden, um die erhobenen Daten in die Position-View im Layout einzutragen. Der Längen- und Breitengrad der aktuellen Position wird einfach als String konvertiert in den beiden TextView-Objekten eingetragen. Die weiteren Methoden der MyLocationListener Klasse sind für die Anwendung des Allergie-Beschwerdebuchs nicht von Bedeutung.

```
@Override
public void onLocationChanged(Location loc) {
```

```

    TextView longitude_textView, latitude_textView;
    String longitude = "" +loc.getLongitude();
    String latitude = "" +loc.getLatitude();
    longitude_textView = (TextView)findViewById(R.id.position_longitude_TextView);
    longitude_textView.setText(longitude);
    latitude_textView = (TextView)findViewById(R.id.position_latitude_TextView);
    latitude_textView.setText(latitude);
}

@Override
public void onProviderDisabled(String arg0) {}

@Override
public void onProviderEnabled(String provider) {}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {}
}

```

Die MySQLiteHelper Klasse ist eine Unterklasse von SQLiteOpenHelper und implementiert einige derer Methoden. In dieser Klasse wird die Datenbank erstellt und verwaltet. Falls noch keine Datenbank existiert, wird diese mit onCreate(SQLiteDatabase db) erstellt mit alle 17 Spalten, die im `DATABASE_CREATE` String deklariert wurden. Mit onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) wird die existierende Datenbank gelöscht und eine Neue, mit neuem Datensatz, erstellt. Die wichtigste Methode ist addAll(...), da hier alle Werte in der Datenbank abgesetzt werden. Zunächst wird dabei eine Referenz zur Datenbank erzeugt und ein ContentValues-Objekt erstellt. Mit den ContentValues kann eine Reihe von Werten abgespeichert werden. Diese ContentValues werden dann in der Datenbank abgelegt. Dazu muss zunächst mit einem Cursor in der Datenbank nach dem Spaltennamen gesucht werden, zu dem ein Wert abgespeichert werden soll. Falls dieser Spaltenname noch nicht existiert, wird er zur Datenbank hinzugefügt. Dem ContentValues-Objekt wird dann der Wert, in Verbindung mit dem entsprechenden Spaltenname, eingefügt. So entsteht am Ende ein ContentValues, mit allen Werten und Spaltennamen, die in die Datenbank einzufügen sind. Mit dem db.insert(TABLE_NAME, null, values) Befehl werden dann alle Werte letztendlich in die Datenbank eingetragen. Im Folgenden sind einige Variablendeklarationen und Methoden etwas gekürzt dargestellt:

```

private static final int DATABASE_VERSION = 1;
private static final String DATABASE_NAME = "Allergie-Beschwerdebuch.db";
static final String TABLE_NAME = "disorders";
public static final String COLUMN_ID = "_id";
public static final String COLUMN_EYES_GRADE = "eyes_grade";
public static final String COLUMN_NOSE_GRADE = "nose_grade";
...
// Database creation sql statement
private static final String DATABASE_CREATE = "create table "
    + TABLE_NAME + "(" + COLUMN_ID
    + " integer primary key autoincrement, "
    + COLUMN_EYES_GRADE + " integer"
    + COLUMN_NOSE_GRADE + " integer"

```

```
+ COLUMN MOUTH GRADE + " integer"
...
+ COLUMN LATITUDE + " text");

public MySQLiteHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(DATABASE_CREATE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
    onCreate(db);
}

public void addAll(int eyes_grade, int nose_grade, int mouth_grade, int
bronchia_grade, int hands_grade, int skin_grade, String name, String quantity, String form,
String notes, int year, int month, int day, int hour, int minute){

    // 1. get reference to writable DB
    SQLiteDatabase db = this.getWritableDatabase();

    // 2. create ContentValues to add key "column"/value
    ContentValues values = new ContentValues();
    Cursor cursor = db.rawQuery("SELECT * FROM " + TABLE_NAME, null);

    //Symptoms
    if(cursor.getColumnIndex(COLUMN EYES GRADE) == -1)
        db.execSQL("ALTER TABLE " + TABLE_NAME + " ADD COLUMN " +
COLUMN EYES GRADE + " integer");
    values.put(MySQLiteHelper.COLUMN EYES GRADE, eyes_grade);

    if(cursor.getColumnIndex(MySQLiteHelper.COLUMN NOSE GRADE) == -1)
        db.execSQL("ALTER TABLE " + TABLE_NAME + " ADD COLUMN " +
COLUMN NOSE GRADE + " integer");
    values.put(MySQLiteHelper.COLUMN NOSE GRADE, nose_grade);

    ...
    // 3. insert() if exists, else update()
    db.insert(TABLE_NAME, null, values); //table, nullColumnHack, key/value -> keys =
column names/ values = column values

    // 4. close
    db.close();
}
}
```

Nach dem Datelexport ist unter dem Root-Verzeichnis nun die csv-Datei zu finden, mit allen Daten und Notizen die eingetragen wurden. Diese Datei kann nun mit Microsoft-

4 Entwicklung

Excel geöffnet- und bearbeitet- oder ausgedruckt werden, um sie dem behandelnden Arzt vorzulegen. Die csv-Datei hat folgende Form:

A	B	C	D	E	F	G
id	eyes	nose	mouth	bronchia	hands	skin
1	2	3	3	4	1	2
H	I	J	K	L	M	N
med_name	med_qunatity	med_form	notes	year	month	day
Antiallergikum	1mg	Tablette	"Kurznotiz..."	2014	08	20
O	P	Q	R			
hour	minute	longitude	latitude			
8	11	-1.2409654073417187	44.71561774145812			

5 Zusammenfassung

Um Allergikern, die unter Heuschnupfen leiden, eine Tool zur Hand zu geben, ihre Beschwerden zu dokumentieren und dieses Dokument einem Arzt vorzulegen um eine gezieltere Diagnose und somit eine besser Behandlung durchzuführen, wurde das Android-Beschwerdebuch entwickelt.

Als erstes musste ich mich mit der Entwicklungsumgebung vertraut machen und mich in das Android Software-Development-Kit unter Eclipse einarbeiten und mich mit den Editoren vertraut machen. Die Analyse ergab folgende Anforderungen.

- Informationen über den Grad der Symptome für Augen, Nase, Mund, Bronchien, Hände und Haut speichern
- Die Medikamenteneinnahme protokollieren
- Das aktuelle Datum und Uhrzeit ermitteln und editierbar darstellen
- Den Standort über GPS ermitteln
- Sämtliche Daten in der SQLite-Datenbank speichern
- Daten als CSV-File exportieren
- Einen Hilfetext anzeigen, welcher dem Nutzer die Funktionen der Applikation erklärt

Mit dem daraus entstandenen Anwendungsfalldiagramm, konnte ich mit dem Softwaredesign beginnen und mir eine Klassenstruktur erstellen, mit allen Methoden die für die Umsetzung der Anwendung nötig sind.

Anschließend wurden alle Layouts erstellt und im Manifest der Anwendung alle Rechte die vom System benötigt werden eingetragen. Nach Erstellung des Menüs und der Stringtable, konnte ich mit der Implementierung des Quellcodes beginnen. In diesem Schritt musste ich mich intensiv mit der SQLite Datenbank vertraut machen. Zunächst wurden alle Methoden der MainActivity implementiert. Eine besondere Herausforderung war die Implementierung der MySQLiteHelper Klasse, da ich anfangs nicht wusste, wie ich den Inhalt der Datenbank auf dem Android-Gerät einsehen kann. Nach dem hinzufügen der Export-Funktion war dieses Problem gelöst, da die csv-Datei den Inhalt der Datenbank abbildet. Zunächst sollte eine Google-Maps Karte den genauen Standort des Anwenders erfassen und somit eine schöne grafische Darstellung ermöglichen. Dies war jedoch zu aufwendig, da die Arbeit mit dem dafür erforderlichen Google API, den zeitlichen Rahmen gesprengt hätte. Auch die Idee, mittels GPS-Daten und einer Adressliste, den Standort des Nutzers als Ortsnamen auszugeben, musste aus zeitlichen Gründen verworfen werden. Auf grafische Feinheiten musste auch verzichtet werden.

Das Ergebnis ist eine Android-Applikation, die durch einfache und schnelle Bedienung jedem Allergiker eine echte Hilfestellung bietet. Die Symptome des Allergikers werden mit Radio-Buttons in 5 Schweregrade bewertet. Durch die Protokollierung der eingenommenen Medikamente und einer Kurznotiz, kann der Allergiker später noch nachvollziehen in weit ihm die Arznei helfen konnte oder nicht. Den aktuellen Standort be-

5 Zusammenfassung

kommt der Nutzer einfach angezeigt, durch Auswahl des Positions-Menüs und kann mit den Längen- und Breitengrade auch später noch ermitteln wo genau er sich aufgehalten hatte. Datum und Uhrzeit werden vom System erfasst und lassen sich bequem durch zwei Scrollbars editieren.

Der eigentliche Nutzen der Anwendung ist aber letztendlich die csv-Datei, die dem Anwender die Möglichkeit gibt, all seine Daten nochmals einzusehen, zu editieren und seinem Arzt vorzulegen. Der letzte Menüpunkt der Anwendung ist der Hilfetext, welcher dem Nutzer eine kleine Unterstützung bei der Benutzung sein soll.

6 Ausblick

Um das Android-Beschwerdebuch noch zu verbessern und weiter zu optimieren gibt es bereits eine Reihe weiterer Ideen und Anregungen. Zum einen wäre eine grafische Oberfläche zur Eingabe der Beschwerden eine gute Alternative, zu den klassischen und vielleicht nicht mehr ganz zeitgemäßen Radio-Buttons. Eine grafische Darstellung eines menschlichen Körpers, oder zumindest Teile davon, an dem man die Körperpartie selektieren kann, an der die Beschwerden auftreten, wäre ein guter Lösungsansatz, um die Benutzerfreundlichkeit zu verbessern.

Um die Standortansicht zu verbessern, wäre die Verwendung der Google Maps API eine schöne Lösung. Um dies zu realisieren, muss man sich aber zunächst intensiv mit der Verwendung der Google Maps API beschäftigen, was ein etwas größerer Zeitaufwand ist. Eine leichtere Alternative zur API, könnte die Verwendung einer Adressliste sein, mit der man mittels GPS-Daten, den aktuellen Ortsnamen des Standorts ermitteln kann.

Um die Medikamenteneinnahme schöner und benutzerfreundlicher zu gestalten, könnten alle je eingetragenen Medikamentennamen in einer Datenbank gespeichert, und als Select-Box zur Auswahl bereitgestellt werden. Außerdem könnte dies mit einer Meldefunktion kombiniert werden, welche dem Nutzer zu einer definierten Zeit, an die Einnahme seiner Medikamente erinnert. Des Weiteren könnte in einer Settings-Funktion ein Medikament als Standard festgelegt werden.

Als letzte Feinheit würde eine Verbesserung der Grafik im Allgemeinen und die Erstellung eines eigenen Icons, das Erscheinungsbild der Android-Anwendung abrunden.

Literatur- und Webverzeichnis

[1] Arno Becker; Markus Pant: „Android – Grundlagen und Programmierung”.
Auflage 2 : dpunkt.verlag

[2] Google-Android: „Android Developers”.
URL: <http://developer.android.com/guide/topics/manifest/manifest-intro.html> [Stand: 20.08.2014].

[3] Spiegel Online: „Heuschnupfen – Wehe, wenn die Pollen kommen”.
URL: <http://www.spiegel.de/gesundheit/diagnose/pollen-allergie-immer-mehr-menschen-leiden-unter-heuschnupfen-a-830187.html> [Stand: 20.08.2014].