UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4 : Informatik

Systematische Untersuchung der Android-Plattform im konzeptuellen Rahmen des 101companies Projektes

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science
im Studiengang Informatik

Vorgelegt von
Hakan Aksu

Erstgutachter: Ralf Lämmel
Institut für Informatik

Zweitgutachter: Andrei Varanovich
Institut für Informatik

Koblenz, im Oktober 2012

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	□	□
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	□	□

(Ort, Datum) (Unterschrift)

Abstract

In this thesis I explore the Android platform. I demonstrate three different implementations, covering features and technology of the 101companies project [1] and I add new Android specific features into the project. The implementations show applications that can be used in the real world.

Zusammenfassung

In der vorliegenden Arbeit untersuche ich die Android-Plattform. Ich demonstriere drei verschiedene Implementationen, die sowohl Features und Technologien des 101companies Projektes [1] abdecken als auch neue Android spezifische Besonderheiten in das Projekt integrieren. Die Implementationen zeigen Anwendungen die in der realen Welt benutzt werden können.

Acknowledgements

I gratefully acknowledge collaboration with Ralf Lämmel (Software Languages Team, Koblenz) and Andrei Varanovich (Software Languages Team, Koblenz) on the underlying 101companies software system.

Inhaltsverzeichnis

1 Einleitung

Heutzutage benutzt ein großer Teil der europäischen Gesellschaft mobile Geräte im Alltag. Im Dezember 2010 hatte jeder vierte Deutsche ein Smartphone. Innerhalb eines Jahres gab es 50% mehr Smartphone Nutzer, so dass im Dezember 2011 mehr als ein Drittel der Deutschen ein Smartphone nutzten [2]. Dabei sind die Funktionen wie das Telefonieren und SMS schreiben schon lange nicht mehr ausreichend. Nicht nur telefonieren und SMS schreiben, sondern auch Musik hören, Videos anschauen, Nachrichten lesen, im Internet surfen, chatten, einkaufen und vieles mehr haben die sogenannten Smartphones und Tablet-PC's zu bieten. Je nach Hard- und Software gibt es die mobilen Geräte in den verschiedensten Leistungs- und Preisklassen. Die Smartphones sind nicht mehr mit den früheren Mobiltelefonen vergleichbar. Die Prozessorleistung hat die Gigahertz Grenze überschritten. Hinzu kommt die Multicore Technologie, die in den neuen Smartphones verbaut ist. Doch nicht nur die Leistung hat zugenommen, sondern es sind zahlreiche Sensoren bzw. Hardwarekomponenten für Smartphones und Tablet-PC's entwickelt worden. Diese ersetzen eine Vielzahl von Geräten, die wir alltäglich brauchen. Dazu gehören Wasserwaagen, Temperaturmessgeräte, Sprachaufzeichnungsgeräte, Kompasse, Digitalkameras, Navigationsgeräte und viele mehr. Um diese umfangreichen Hardwareteile richtig zu steuern und zu verwenden gibt es verschiedene Software- bzw. Plattformentwickler. Die drei Marktführer sind zurzeit die Plattformen Android, IOS und Symbian [3]. In meiner Arbeit werde ich die Android Plattform im konzeptuellen Rahmen des 101companies Projektes systematisch untersuchen.

Zuerst werde ich im zweiten Kapitel die Android Plattform beschreiben und danach die abgeleiteten Features und Konzepte darstellen. Im dritten Kapitel demonstriere ich die drei Implementationen, die einen Realitätsbezug haben. Am Ende erfolgt eine Zusammenfassung und ein Ausblick auf weitere mögliche Themen, die zu dem 101companies Projekt integrierfähig wären.

2 Planung

2.1 Beschreibung der Android Plattform

Es existieren sehr viele Versionen der Android Plattform. In der folgenden Tabelle sind die wichtigsten Versionen aufgelistet.

Version	Codename	API	Distribution
1.5	Cupcake	3	0.1%
1.6	Donut	4	0.4%
2.1	Eclair	7	3.4%
2.2	Froyo	8	12.9%
2.3 - 2.3.2	Gingerbread	9	0.3%
2.3.3 - 2.3.7		10	55.5%
3.1	Honeycomb	12	0.4%
3.2		13	1.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	23.7%
4.1	Jelly Bean	16	1.8%

Tab 1: Android Versionen [4]

Die zurzeit am meisten genutzte Version ist Gingerbread (API 10) mit 55,5%. Diese Version habe ich auch bei meinen Implementationen verwendet. Alle Android Plattformen mit höheren API Level als zehn können die Applikationen, die für API 10 geschrieben wurden, auch verwenden. Somit habe ich 82,9% der Android Nutzer abgedeckt.

Android Applikationen auch kurz Apps genannt werden grundsätzlich per Touchscreen bedient. Je nach Gerät kann man die Bedienung und Funktionen mithilfe Sensoren bzw. Hardwarekomponenten erweitern. Sensoren wie dem Beschleunigungs-, Schwerkraft-, Gyroscope(Kreisel)-, Licht-, Linear Beschleunigungs-, Magnetfeld-, Ausrichtungs-, Druck-, Näherungs-, Drehgeschwindigkeits- und Temperatur-Sensor erlauben dem Programmierer und dem Nutzer eine variationsreiche Nutzung der

mobilen Geräte. Hinzu kommen noch weitere Hardwarekomponenten wie Touchscreen, Kamera, WLAN, GPS, Mikrofon, Bluetooth und Near Field Communication (NFC). Die lange Aufzählung soll zeigen, dass es sehr viele Möglichkeiten gibt verschiedene Applikationen zu entwickeln. Doch nicht nur die Hardware macht Android variationsreich, sondern auch die Designmöglichkeiten der einzelnen Applikationen.

In meiner Arbeit werde ich mich auf einige Hardwarekomponenten und Designmöglichkeiten konzentrieren und diese in das 101companies Projekt integrieren. Dafür werde ich im Folgenden analysieren welche Features vom 101companies Projekt benötigt werden und welche Konzepte ich aus Android verwende.

2.2 Beschreibung der Implementationen

Ich habe drei Implementationen, die ich in meiner Arbeit demonstrieren werde. Meine Applikationen sind nicht nur fiktiv, sondern in der realen Welt anwendbar. Ich unterteile meine Apps in zwei Anwendungsgebiete.

Zwei meiner Implementationen, simpleAndroid und richerAndroid, sind zur Verwaltung einer oder mehrerer Firmen gedacht. SimpleAndroid soll die Grundlagen einer Android Applikation demonstrieren und kann somit nur eine Beispielfirma öffnen und diese anzeigen. RicherAndroid hingegen kann nicht nur eine Firma anzeigen, sondern sie kann auch mehrere Firmen verwalten. Dazu gehören das Erstellen, Bearbeiten, Kopieren, Einfügen und Löschen von Firmen, Abteilungen und Arbeitern. Ganz typisch wäre diese Applikation für eine Personalabteilung.

Meine dritte Implementation deckt das zweite Anwendungsgebiet ab. Es ist für den Arbeiter selbst gedacht. Der Arbeiter hat dabei die Möglichkeit sein zu erwartendes Gehalt einzusehen. Per Standortbestimmung wird ermittelt, ob der Arbeiter sich an seinem Arbeitsplatz befindet. Wenn das der Fall sein sollte, dann wird die Zeitdifferenz zwischen der letzten und jetzigen Prüfung zu seiner Arbeitszeit hinzugefügt. Damit der Rahmen meiner Arbeit nicht gesprengt wird, habe ich Überprüfungen wie der aktuellen Uhrzeit oder die Beachtung der Pausen weggelassen.

2.3 Ableitung von Features zur Abdeckung

Hier demonstriere ich die benötigten Features des 101companies Projektes [5].

	Behavioral				Quality		Structural	UI					
	Data export	Data import	Type-driven query	Type-driven transformation	Reliability	Serialization	Tree structure	Attribute editing	Exploration	Geolocation	Localization	Structural editing	Touch control
Simple					•		•		•				•
Richer			•	•	•	•	•	•	•		•	•	•
Mobile	•	•			•					•			•

simple = simpleAndroid richer = richerAndroid mobile = mobileAndroid

Alle Implementationen:

- Die Bedienung erfolgt über den Touchscreen (Touch control)
- Die Absturzsicherheit bzw. Zuverlässigkeit (Reliability) ist gewährleistet.

simpleAndroid …

- … stellt die Firma mit ihren Abteilungen und Arbeitern als Baum Struktur (Tree structure) dar.
- … ermöglicht die Erkundung (Exploration) der einzelnen Bestandteile der Firma

richerAndroid …

- … enthält die Basisfunktionen *cut* (Type-driven transformation) und *total* (Type-driven query).
- … ermöglicht die Serialisierung (Serialization)
- … stellt die Firma mit ihren Abteilungen und Arbeitern als Baumstruktur (Tree structure) dar.
- … unterstützt das Erstellen, Bearbeiten, Löschen, Kopieren und Einfügen von Firmen, Abteilungen und Arbeitern (Structural editing & Attribute editing).

mobileAndroid …

- … Ermöglicht das Importieren und Exportieren der Arbeiterdaten in eine und aus einer XML Datei (Data import & Data export).
- … Arbeitet mit realen Geokoordinaten um die Position des Arbeiters festzustellen (Geolocation).

2.4 Ableitung von Konzepten zur Abdeckung

								Android UI										Location			
	Android project	Android Resource	Android Manifest	Intent	Android Service	Android Preference	Connectivity	Activity	Adapter	Android Menu	AppWidget	Dialog	Layout	Listener	Menu Inflater	Toast	View	Location Listener	Location Manager	Map view	Overlay
simple	•	•	•	•				•	•				•	•		•	•				
richer	•	•	•	•				•	•	•		•	•	•	•	•	•				
mobile	•	•	•	•	•	•	•	•	•		•	•	•	•		•	•	•	•	•	•

- Android project

  A generated project with Android Development Tool for Android applications [6]

- Android Resource

  The resource files of an Android application [7]

- Android Manifest

  The essential information about the application to the Android system [8]

- Intent

  An abstract description of an operation to be performed [9]

- Android Service

  A component that runs in the background without Android-UI [10]

- Android Preference

  An Android feature to save and load settings [11]

- Connectivity Manager

  A manager to give information about the state of network connectivity [12]

Android UI

- Activity

  The visible part of the Android application [13]

- Adapter

  A bridge between an adapter view and the underlying data. [14]

- Android Menu

  A menu in Android [15]

- AppWidget

  A miniature application view that can be embedded in other applications. [16]

- Dialog

  A small window that appears in front of the current activity [17]

- Layout

  The architecture for the user interface in an activity [18]

- Listener

  An interface in the view class that contains a single callback method [19]
  Look also observer pattern

- Menu Inflater

  A tool to inflate the context menu from a menu resource [20]

- Toast

  A view containing a quick little message for the user [21]

- View

  Something on the screen that the user can interact with [22]

Android Location

- Location Listener

  A listener to get the geolocation from the location manager [23]

- Location Manager

  Provides access to the system location services [24]

- Map view

  A view which displays a map [25]

- Overlay

  An overlay which may be displayed on top of a map [26]

3 Implementationen

3.1 simpleAndroid

3.1.1 Headline

--- Basic Android programming ---

3.1.2 Languages

Java

XML

3.1.3 Technologies

Android

Android_SDK (API 10 / Android 2.3.3)

Android Development Tool

3.1.4 Features

Tree_structure

Touch_control

Exploration

Reliability

Visualization

3.1.5 Motivation

The implementation demonstrates basic style of Android programming using the Android SDK and the Android Development Tool. The application is programmed in a way that touch control is used exclusively. It shows basic concepts like the Android Manifest, the layout and the use of activities. Other concepts are demonstrated that are needed to represent the 101features Tree structure and Exploration. This includes the views like buttons, the click listener and the use of an adapter with a list view. For the communication between activities and to start them are used intents. The application uses text views to display the name, address and salary of an employee. It shows toasts for any notifications.

3.1.6 Illustration

3.1.6.1 The Android Manifest

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.     package="org.softlang.simpleAndroid"
4.     ...
5.     <uses-sdk android:minSdkVersion="10" />
6.     <application
7.         android:icon="@drawable/ic_launcher"
8.         android:label="@string/app_name" >
9.         <activity
10.            android:name=".SimpleAndroidActivity" ... >
11.            <intent-filter>
12.                <action android:name="android.intent.action.MAIN" />
13.                <category android:name="android.intent.category.LAUNCHER" />
14.            </intent-filter>
15.        </activity>
16.
17.        <activity android:name=".CompanyClickActivity"></activity>
18.        ...
19.    </application>
20. </manifest>
```

The Android Manifest has all activities, permissions and all other essential settings of the application.

3.1.6.2 Button & ClickListener

In the following we explain how a button is designed and how functionality can be assigned.

Button

```
1.  <Button
2.      android:id="@+id/bt_loadtestCompany"
3.      android:layout_width="match_parent"
4.      android:layout_height="match_parent"
5.      android:layout_weight="50"
6.      android:text="@string/load_test_company"
7.      android:onClick="onButtonClick"
8.      android:textSize="15dp"
9.      android:padding="20dp" />
```

In line two we assign the ID to the button. in line seven we define the on click listener for the button.

ClickListener

```
1.  public void onButtonClick(View v) {
2.  switch(v.getId()){
3.  case R.id.bt_Company:
4.    if (loadedCompany != null){
5.      Intent i = new Intent(this,CompanyClickActivity.class);
6.      i.putExtra("Company",loadedCompany);
7.      startActivity(i);
8.    }else {
9.      Toast.makeText(this, R.string.company_not_loaded,
                              Toast.LENGTH_SHORT).show();
10.   }
11. break;
12. }
```

If the button with the ID "R.id.bt_Company" is clicked then in line five we create a new intent with the name of the activity which we want to start. In line six we put the company object into the intent with a name (key). In line seven the new activity with the information of the intent will be started. If the company object is not loaded then the Activity creates a toast and displays it on the screen.

3.1.6.3 List view & Adapter

In this section we explain how a list view is designed and how to use a list adapter.

ListView

```
1.  <ListView
2.      android:id="@+id/android:list"
3.      android:layout_width="fill_parent"
4.      android:layout_height="fill_parent" >
5.  </ListView>
```

In line two we have the standart ID for a list view in Android.

Adapter

```
1.  ...
2.  adapter = new ArrayAdapter<String>(   this,
3.                          android.R.layout.simple_list_item_1,
4.                          deptsString);
5.  setListAdapter(adapter);
6.  ...
7.  @Override
8.  protected void onListItemClick(ListView l, View v, int position, long id) {
9.    super.onListItemClick(l, v, position, id);
10.   Intent i = new Intent(this, DepartmentClickActivity.class);
11.   i.putExtra("Department", company.getDepts().get(position));
12.   i.putExtra("Company", company);
13.   startActivity(i);
14. }
```

The names of the departments are in an array of String "deptsString". Android.R.layout.simple_list_item_1 is a standard design of listelements. With these informations we generate an array adapter and set the list adapter of the activity with "setListAdapter(adapter)". The onListItemClick-listener gives each element functionality.

3.1.7 Architecture

- In the package *org.softlang.company* we define the object-model of the Company.
- We implement the activities in the package *org.softlang.simpleAndroid*.
- The sample company is in the package org.softlang.tests.
- In layout we define various layouts of the different views.

3.1.8 Usage

- You need the Android SDK Manager to instal the Android SDK (API 10)
- Instal the Android Development Tool in Eclipse.
- Import the Project into Eclipse
- Run the Project as Android Application

3.2 richerAndroid

3.2.1 Headline

--- Advanced Android programming ---

3.2.2 Languages

- Java
- XML

3.2.3 Technologies

- Android
- Android_SDK (API 10 / Android 2.3.3)
- Android Development Tool

3.2.4 Features

- Tree_structure
- Touch_control
- Exploration
- Type-driven_query
- Type-driven_transformation
- Structural_editing
- Attribute_editing
- Serialization
- Localization
- Reliability

3.2.5 Motivation

The implementation demonstrates advanced style of Android programming using the Android_SDK and the Android Development Tool. It is based on simpleAndroid with more features and functionalities. The application is in the position to explore and edit the company structure. The feature Serialization can save and load company objects. The total and cut features are also implemented. The application demonstrates more listeners like the onLongClickListener. This implementation shows how to use different dialogs and how to use options menu. Also there are more views like the edit text. The feature Localization is also implemented. If you change your system language then the application language will also change.

3.2.6 Illustration

3.2.6.1 Dialogs

AlertDialog

```
1.  Builder build = new Builder(DepartmentClickActivity.this);
2.    build.setTitle(R.string.areyousure)
3.    .setPositiveButton(R.string.yes, new DialogInterface.OnClickListener() {
4.                @Override
5.                public void onClick(DialogInterface dialog, int which) {
6.                ...
7.    })
8.    .setNegativeButton(R.string.no, new DialogInterface.OnClickListener() {
9.                @Override
10.               public void onClick(DialogInterface dialog, int which) {
11.                       dialog.dismiss();
12.               }
13.   })
14.   .show();
```

An alert dialog is generated with a builder. In line two we set the title of the dialog and in the next lines we set two buttons with their on click listener. In the last line we show the Dialog in front of the active activity.

Custom Dialog

```
1. Dialog dialog = new Dialog(this);
2. dialog.setTitle(R.string.createtheemployee);
3. dialog.setContentView(R.layout.newemployee);
4. ((Button)dialog.findViewById(R.id.bt_Cancel_newemployee)).setOnClickListene
   r(new OnClickListener() {
5.    @Override
6.    public void onClick(View arg0) {
7.                dialog.dismiss();
8.    }
9. });
10. dialog.show();
```

In a custom Dialog we put a view in line three, which we have created in the resources as XML layout. In the last line we show the dialog in front of the active activity.

3.2.6.2 Options Menu

Create Menu

```
1. @Override
2. public boolean onCreateOptionsMenu(Menu menu) {
3.    getMenuInflater().inflate(R.menu.menue, menu);
4.    menu.getItem(1).setTitle(R.string.addEmployee);
5.    menu.getItem(0).setTitle(R.string.addDepartment);
6.    return super.onCreateOptionsMenu(menu);
7. }
```

If we click on the options button of our Smartphone then the onCreateOptionsMenu method will be started. In line three we put the created menu layout with the menu inflater. In line four and five we edit the entries of the Menu.

Give functionality

```
1. @Override
2. public boolean onOptionsItemSelected(MenuItem item) {
```

```
3.    switch (item.getItemId()) {
4.      case R.id.opt_total:{
5.        Toast.makeText(this, dept.total()+"", Toast.LENGTH_SHORT).show();
6.        return true;
7.      }
8.      case R.id.opt_cut:{
9.        dept.cut();
10.       Toast.makeText(this,R.string.successful, Toast.LENGTH_SHORT).show();
11.       return true;
12.     }
13.     ...
14.   }
15.   return false;
16. }
```

With the method onOptionsItemSelected we give the elements of the menu a funtionality.

3.2.6.3 Edittext

```
1. EditText ed_name = ((EditText)findViewById(R.id.ed_name));
2. ...
3. ed_name.setText(employee.getName());
4. ...
5. ((EditText)findViewById(R.id.ed_name)).getText().toString().trim();
```

With an EditText we can edit the data of an Employee, the name of a Department and the name of the Company. In line three we put a String for example the employee's name, address or salary. After changing the String or after an action of the user we get the String from the edit text in line five.

3.2.7 Architecture

- In the package *org.softlang.company* we define the Object-Model of the company.
- The Type-driven_query and Type-driven_transformation is integrated in the Object-Model of the Company.
- In values/strings.xml we specify the Strings of the application.
- In values-de/strings.xml we specify the Strings of the application to demonstrate the 101feature Localization.
- In layout we define various layouts of the different views.
- We implement the activities in the package *org.softlang.activities*.
- Package *org.softlang.features* provides functionality for the Structural_editing and Serialization.
- The sample company is in the package *org.softlang.tests*.

3.2.8 Usage

- See Usage of simpleAndroid.

3.3 mobileAndroid

3.3.1 Headline

--- Advanced Android programming ---

3.3.3 Languages

- Java
- XML

3.3.4 Technologies

- Android
- Android SDK (API 10 / Android 2.3.3)
- Android Development Tool

3.3.5 Features

- Touch control
- Data export
- Data import
- Geolocation
- Reliability

3.3.6 Motivation

The implementation demonstrates advanced style of Android programming using the Android_SDK and the Android Development Tool. The application uses other approaches compared to the two other implementations simpleAndroid and richerAndroid. This application is intended for employees. It aims to track his worked time. At the end of the month the Employee get an approximate salary expectation. It uses the geolocation feature by checking into a Android Service which is a background application to determine whether the employee is in the workplace. The time difference between the last and current review is added to the working hours.

The application demonstrates how to use...

- ... different activities and layouts like the Tab activity.
- ... the export and import features.
- ... a Android Service which is a background application.
- ... the connectivity manager and location listener
- ... a map view with different overlays
- ... the toggle button or table layout which are different views
- ... Android preferences

3.3.7 Illustration

3.3.7.1 AppWidget

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <appwidget-provider
   xmlns:android="http://schemas.android.com/apk/res/android"
3.    android:minWidth="147dip"
4.    android:minHeight="72dip"
5.    android:initialLayout="@layout/main"
6. />
```

In the XML-Code we define the default properties of our AppWidget. In line three and four we define the size of the AppWidget-Layout and in line five we define the Layout of the AppWidget.

```
1. public class CompanyWidget extends AppWidgetProvider{
2.    @Override
3.    public void onUpdate(Context context, AppWidgetManager
   appWidgetManager, int[] appWidgetIds) {
4.       super.onUpdate(context, appWidgetManager, appWidgetIds);
5.       ...
6.       appWidgetManager.updateAppWidget(appWidgetIds, views);
7. }
```

In the class CompanyWidget we update the AppWidget

3.3.7.2 TabActivity

```
1. intent = new Intent().setClass(this, GeneralSettings.class);
2. spec = tabHost.newTabSpec("general").setIndicator("General",
3.           res.getDrawable(android.R.drawable.ic_dialog_map))
4.           .setContent(intent);
5. tabHost.addTab(spec);
6. // Do the same for the other tabs
7. ...
8. tabHost.setCurrentTab(1);
```

In line one we create an intent to launch an Activity for the tab. In line two till five we initialize a TabSpec for each tab and add it to the TabHost. In the last line we set our default TabSpec.

3.3.7.3 Manager & Provider

```
1. lManager = (LocationManager) getSystemService(LOCATION_SERVICE);
2. cManager=(ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
```

We use the system services Location Manager and Connectivity Manager to get the status of GPS and internet connection.

3.3.7.4 Service

```
1. Intent i = new Intent(this, WorkService.class);
2. i.putExtra("profile", profile);
3. startService(i);
```

In line one we create an Intent with the Service Class. In line two we put the profile object in the intent and in line three we start the Service.

3.3.7.5 Geolocation

```
1. if(gps_enabled) {
2.    lManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
   UPDATE_TIME, 0, updateListenerForGPS);
3. }
4.    else
5.       if(network_enabled)
6.
   lManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,
   UPDATE_TIME, 0, updateListenerForNetwork);
```

In line two and five we start the requests for the location.

```
1. private LocationListener updateListenerForGPS = new LocationListener() {
2.    @Override
3.    public void onStatusChanged(String provider, int status, Bundle extras) {
4.    }
```

```
5.  @Override
6.  public void onProviderEnabled(String provider) {
7.    lManager.removeUpdates(updateListenerForNetwork);
8.  }
9.  @Override
10. public void onProviderDisabled(String provider) {
11. }
12. @Override
13. public void onLocationChanged(Location location) {
14. ...
15. profile.addWorkedTime((double)(actualTime-lastUpdateTime)/60000);
16. ...
17. }
18. };
```

In line twelve we get in the onLocationChanged-method the current location. We use it, to look if the Employee is in the company. If he is then he get in line 14 a salary increase.

3.3.7.6 Map view & Overlay

Map view

```
1. <RelativeLayout
2.   ...>
3.   <com.google.android.maps.MapView
4.     android:id="@+id/mv_google_map"
5.     ...
6.     android:apiKey="@string/map_key"/>
7. </RelativeLayout>
```

In line 4 we give the map view an ID. In line 6 we use a map_key. Here you find more information about the Map key. We show the Map in a MapActivity.

Overlay

We use two overlays in our implementation. The MapActivity constructs a rectangle with the CompanyOverlay and the MyLocationOverlay is a standard Android overlay to draw your own position.

```
1. MapView mView;
```

```
2. List<Overlay> mapOverlays;
3. MyLocationOverlay myLocationOverlay;
4. ...
5. myLocationOverlay = new MyLocationOverlay(this, mView);
6. ...
7. mView.getOverlays().add(myLocationOverlay);
```

In line 5 we create a MyLocationOverlay. In line 7 we get a list with overlays and we add the created mylocationOverlay to draw our position.

3.3.7.7 Preferences

```
1. <PreferenceScreen
   xmlns:android="http://schemas.android.com/apk/res/android" >
2.    <PreferenceCategory>
3.       <Preference android:key="profile" />
4.    </PreferenceCategory>
5. </PreferenceScreen>
```

In line three we define the key for the preference.

```
1. private SharedPreferences settings;
2. ...
3. settings = getSharedPreferences(getPackageName()+"_preferences",
   MODE_PRIVATE);
4. ...
5. settings.edit().putInt("profile", pos).commit();
6. ...
7. settings.getInt("profile", 0);
```

In line three we get the preference-object. In line five we edit the preference with putInt(key,value) and we commit the changes. In line seven we get the value with getInt(key,default).

3.3.8 Architecture

- In the package *org.softlang.company.mobileAndroid* we define the app widget provider and the tab activities.
- In strings.xml we specify the Strings of the Application.
- In layout we define various layouts of the different views.
- In the package *org.softlang.company.services* we implement the Android Service.
- In the package *org.softlang.company.location* we demonstrate the map view and overlay.
- In the package *org.softlang.company.data* we define the profile class and the xml Parser to import and export the data.
- In company_widget_provider.xml we have settings for the AppWidget.
- In settings.xml we have the Android preference xml-file.

3.3.9 Usage

- See Usage of simpleAndroid.
- optional: if you want to use the Google MapView -> Here you find more information about the Map key

4 Abschließende Bemerkungen

4.1 Zusammenfassung

Mit den drei Implementationen, die ich vorhin demonstriert habe, habe ich einige Konzepte der Android Programmierung systematisch gezeigt. Dabei habe ich den Inhalt so gewählt, dass es in der realen Welt anwendbar ist. Das 101companies Projekt habe ich mit der Android Programmierung bereichert und die verwendeten Konzepte integriert. Somit ist das Ziel die Android Plattform im konzeptuellen Rahmen des 101companies Projektes systematisch zu untersuchen gelungen.

4.2 Ausblick

Die demonstrierten Konzepte können noch in verschiedenen Variationen implementiert werden. Es gibt auch weitere Konzepte, die auf anderer Art und Weise in Apps angewendet werden können. Hier werde ich einige Fallbeispiele geben, die in den konzeptuellen Rahmen des 101companies Projektes passen.

Wie schon in der Einleitung erwähnt, gibt es zahlreiche Hardwarebestandteile und viele Konzepte, die mithilfe der Android Plattform bedient werden können. Die Frage, die man sich hier stellt ist: Welche Hardware oder Konzepte kann man sinnvoll mit dem 101companies Projekt verbinden?

Die demonstrierten Apps könnte man zum Beispiel durch eine andere Bedienung erweitern. Möglichkeiten wären durch die Sprachsteuerung gegeben. Eine andere Möglichkeit wäre der Bewegungssensor, wo man durch bestimmte Bewegungen sich innerhalb der Abteilungen und Arbeiter bewegen kann. Solche Erweiterungen können auch kombiniert werden, indem man dem Nutzer erlaubt die Steuerungsmöglichkeit einzustellen. Abgesehen von der Steuerung gibt es noch die Anzeigevarianten der Firma. Die Firma mit ihren Abteilungen und Arbeitern kann durch Symbole, anstatt Listen, dargestellt werden. Somit könnte man auch eventuell *drag&drop* Funktionen einbauen. Dieser Aspekt würde jedoch nicht von der Hardware, sondern von der spezifischen Android Version abhängen.

Eine seit neuestem in den Smartphones vorhandene Technologie kann in einem realen Bezug in das 101companies Projekt integriert werden. Die Near Field Communication Technologie kurz NFC ist für kurze und gleichzeitig schnelle Datenübertragungen gedacht. Dabei muss das Smartphone in unmittelbarer Nähe ca. vier Zentimeter an einem Partnergerät sein. Diese Technologie könnte ein Arbeiter zur Authentifizierung am Eingang zur Firma bzw. zum Arbeitsbereich oder beim Ausführen bestimmter Tätigkeiten innerhalb der Firma genutzt werden. Die übertragenen Daten könnten dabei die persönlichen Daten des Arbeiters sein.

5 Literaturverzeichnis

[1] 101companies Projekt
http://101companies.org/index.php/101companies:Project

[2] Statista: Besitz und geplante Anschaffung von ausgewählten internetfähigen Geräten im Januar 2012
http://de.statista.com/statistik/daten/studie/219326/umfrage/besitz-und-geplante-anschaffung-internetfaehige-geraete/

[3] Statista: Marktanteil der führenden mobilen Betriebssysteme an der Internetnutzung mit Mobilgeräten in Deutschland von Januar 2009 bis Juli 2012
http://de.statista.com/statistik/daten/studie/184332/umfrage/marktanteil-der-mobilen-betriebssysteme-in-deutschland-seit-2009/

[4] Plattform Versionen (Stand 01.10.2012)
http://developer.android.com/about/dashboards/index.html

[5] Features des 101companies Projektes
http://101companies.org/index.php/Category:101feature

[6] Android Projekt
Managing Projects @ developer.android.com
http://developer.android.com/tools/projects/index.html

[7] Android Resource
Resource types @ developer.android.com
http://developer.android.com/guide/topics/resources/available-resources.html

Providing Resources @ developer.android.com
http://developer.android.com/guide/topics/resources/providing-resources.html

Accessing Resources @ developer.android.com
http://developer.android.com/guide/topics/resources/accessing-resources.html

[8] Android Manifest
Android Manifest @ developer.android.com
http://developer.android.com/guide/topics/manifest/manifest-intro.html

[9] Intent

Intent @ developer.android.com

http://developer.android.com/reference/android/content/Intent.html

Intent & IntentFilter guide @ developer.android.com

http://developer.android.com/guide/components/intents-filters.html

[10] Android Service

Services @ developer.android.com

http://developer.android.com/guide/components/services.html

Documentation @ developer.android.com

http://developer.android.com/reference/android/app/Service.html

[11] Android Preference

Preference @ developer.android.com

http://developer.android.com/guide/topics/ui/settings.html

Preference documentation @ developer.android.com

http://developer.android.com/reference/android/preference/Preference.html

preference package summary @ developer.android.com

http://developer.android.com/reference/android/preference/package-summary.html

[12] Connectivity Manager

Connectivity Manager @developer.android.com

http://developer.android.com/reference/android/net/ConnectivityManager.html

[13] Activity

Activity @ developer.android.com

http://developer.android.com/reference/android/app/Activity.html

[14] Adapter

Adapter @ developer.android.com

http://developer.android.com/reference/android/widget/Adapter.html

Adapter View @ developer.android.com

http://developer.android.com/guide/topics/ui/binding.html

[15] Android Menu

Menu @ developer.android.com

http://developer.android.com/guide/topics/ui/menus.html

Documentation @ developer.android.com

http://developer.android.com/reference/android/view/Menu.html

[16] AppWidget

App Widget @ developer.android.com

http://developer.android.com/guide/topics/appwidgets/index.html

App Widget Design Guidelines @ developer.android.com

http://developer.android.com/guide/practices/ui_guidelines/widget_design.html

[17] Dialog

Dialog @ developer.android.com

http://developer.android.com/guide/topics/ui/dialogs.html

Dialog Documentation @ developer.android.com

http://developer.android.com/reference/android/app/Dialog.html

AlertDialog @ developer.android.com

http://developer.android.com/reference/android/app/AlertDialog.html

[18] Layout

Layout @ developer.android.com

http://developer.android.com/guide/topics/ui/declaring-layout.html

[19] Listener

Input Events @ developer.android.com

http://developer.android.com/guide/topics/ui/ui-events.html

Input Control @ developer.android.com

http://developer.android.com/guide/topics/ui/controls.html

[20] Menu Inflater

Menu Inflater @ developer.android.com

http://developer.android.com/reference/android/view/MenuInflater.html

Menu @ developer.android.com

http://developer.android.com/guide/topics/ui/menus.html

[21] Toast

Toast @ developer.android.com

http://developer.android.com/guide/topics/ui/notifiers/toasts.html

Documentation @ developer.android.com

http://developer.android.com/reference/android/widget/Toast.html

[22] View

View guide @ developer.android.com

http://developer.android.com/guide/topics/ui/overview.html

View documentation @ developer.android.com

http://developer.android.com/reference/android/view/View.html

ViewGroup @ developer.android.com

http://developer.android.com/reference/android/view/ViewGroup.html

[23] Location Listener

Location Listener @ developer.android.com

http://developer.android.com/reference/android/location/LocationListener.html

[24] Location Manager

Location Manager @ developer.android.com

http://developer.android.com/reference/android/location/LocationManager.html

[25] Map view

Map View @ developer.google.com

https://developers.google.com/maps/documentation/android/reference/com/google/android/maps/MapView

Map View Documentation @ developer.google.com

https://developers.google.com/maps/documentation/android/hello-mapview

[26] Overlay

Overlay @ developers.google.com

https://developers.google.com/maps/documentation/android/reference/com/google/android/maps/Overlay

Hello MapView @ developers.google.com

https://developers.google.com/maps/documentation/android/hello-mapview