1 Dokumentation der Proof-Of-Concepts

1.1 GPS Lokalisierung

Die Lokalisierung des aktuellen Standorts in der Google Maps Karte ist von grosser Bedeutung für den Erfolg der Applikation. Durch den Erfolg dieses Proof-Of-Concepts soll die Implementierung der Google Maps Schnittstelle nicht mehr als Risiko zählen.

Ziel des Proof-Of-Concepts ist die Erstellung der Google Maps Karte, welches den aktuellen Standort anzeigt.

Die Programmierung erfordert die Registrierung eines API-Schlüssels auf der Google Developers Console. Hierfür ist die Anmeldung mit dem Google Konto pflicht. Danach kann das Projekt erstellt werden und der Benutzer erhält den automatisch generierten API-Schlüssel. Dieser Schlüssel muss in der 'google-maps-api.xml' an die passende Stelle hinzugefügt werden.

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE

    You can also add your credentials to an existing key, using this line:
    DF:78:E1:16:49:32:BD:4B:CB:8E:9F:56:C8:CF:87:CE:65:FE:7F:40;com.example.sinemkaya.myapplication

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" translatable="false" templateMergeStrategy="preserve">
        xxxxxxxxxxxxxAPI_SCHLÜSSEL_HIER
    </string>
</resources>
```

Figure 1: Datei 'google-maps-api.xml' mit dem generierten API-Schlüssel

Als nächstes werden vier Textviews in der 'activity maps.xml' Datei erstellt, mit jeweils einem Label und einem Textfeld für Breiten- und Längengrad für die GPS-Koordinaten.

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/lblLatitude"
    android:text="Latitude:"/>

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/tvLatitude"/>

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/lblLontitude"
    android:text="Longtitude:"/>

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/tvLongtitude"/>
```

Figure 2: Ausschnitt aus der Datei 'activity maps.xml'

Um die Positionsdaten beziehen zu können, musste eine Verbindung mit einem Location-Provider hergestellt werden. Dies geschieht in der 'MapsActivity.java'. Erst werden die TextViews anhand der ID's abgerufen, die in der XML-Datei erstellt worden sind. Danach wird der Handle für den LocationManager abgerufen, der über den Code getSystemService(Context.LOCATION SERVICE) eine Verbindung mit dem LocationManager herstellt. Mit getLastKnownLocation("provider") wird dem LocationManager eine Anfrage nach dem aktuellen Standort gestellt, dessen Ergebnisse der Breiten- und Längengrad, die TextViews füllen.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);

    //TextViews abrufen
    TextView tvLatitude = (TextView)findViewById(R.id.tvLatitude);
    TextView tvLongtitude = (TextView)findViewById(R.id.tvLongtitude);

    //Handle für LocationManager abrufen
    LocationManager lm = (LocationManager)
            getSystemService(Context.LOCATION_SERVICE);

    //Mit GPS-Provider verbinden
    Location loc = lm.getLastKnownLocation("gps");

    //TextViews füllen
    tvLatitude.setText(Double.toString(loc.getLatitude()));
    tvLongtitude.setText(Double.toString(loc.getLongitude()));


}
```

Figure 3: OnCreate-Methode der Datei 'MapsActivity.java'

Da der Emulator keine GPS-Daten enthält, wurde die Ausführung lediglich auf einem mobilen Device durchgeführt.

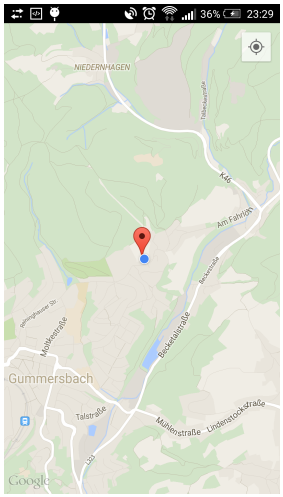

Figure 4: Anwendung in Ausführung - GPS-Lokalisierung

Testergebnisse

Die Anwendung sollte an 10 verschiedenen Orten getestet werden, um sicher zu gehen, dass die Präzision der Standorte wahrhaftig sind. Dabei wurden die Standorte bis auf den Straßennamen genau getestet. Die Ergebnisse waren erfolgreich, sodass das Exit-Kriterium - Lokalisierung des aktuellen Standorts - erfüllt worden ist.

Die Implementierung der Google Maps Schnittstelle sollte kein Risiko mehr für die Fortführung des Projekts darstellen.

1.2 Wetterdaten mit der GET-Methode abrufen

Der Abruf der Wetterdaten soll über die openweather Schnittstelle geschehen. Hierfür gibt es drei Such-Alternativen, wie man die aktuellen Daten abrufen kann. Zum Einen kann die Suche über eine Eingabe des erwünschten Ortes oder der Postleitzahl passieren und zum Anderen können die geografischen Koordinaten zur Initialisierung des aktuellen Standorts benutzt werden. Da sich die Benutzer nicht unbedingt immer am selben Standort aufhalten, ist die Alternative der manuellen Eingabe des Ortes sicherlich vorteilhaft.

Es sind zwei Ausgabeformate vorhanden, die von openwether angeboten werden, XML und JSON. Im Architekturdiagramm ist die Entscheidung letztendlich auf XML gefallen.

Zunächst wurden fünf TextViews und fünf EditTexts in der 'activity main.xml'

erstellt, um die Werte zu Standort, Land, Temperatur Luftfeuchtigkeit und Luftdruck in der XML-Datei ablegen zu können.

Es wird eine zusätzliche Datei 'HandleXML' erstellt, worin ein XML-Parser die Verarbeitung der bereitgestellten Wetterdaten übernimmt. Es ist möglich ein eigenes Programm für das Parsen der XML-Datei zu schreiben, jedoch ist dies recht komplex und ist mit einem großen Zeitaufwand einzuschätzen. Aus dem Grund wurde das XMLPullParser verwendet, da es einfach in der Nutzung ist. Die Verarbeitung der Daten und somit das Parsen geschieht in der Methode 'parseXMLAndStoreIt'. Die 'fetchXML' ist für die Verbindung mit dem Server zuständig, welches die HTTP-Methode GET nutzt, um die Ressource anzufragen und aufzurufen. Hierfür wird die Verbindung zur URL hergestellt und über die setRequest-Methode die GET-Methode aufgerufen.

```
public void fetchXML(){
    Thread thread = new Thread(new Runnable(){
        @Override
        public void run() {

            try {
                URL url = new URL(urlString);
                HttpURLConnection conn = (HttpURLConnection)
                        url.openConnection();
                conn.setReadTimeout(10000 /* milliseconds */);
                conn.setConnectTimeout(15000 /* milliseconds */);
                conn.setRequestMethod("GET");
                conn.setDoInput(true);
                conn.connect();
                InputStream stream = conn.getInputStream();
```

Figure 5: fetchXML-Methode der Datei 'HandleXML.java'

Das Relevante in der 'MainActivity.java' ist die open-Methode. Dort wird der manuell eingegebene Ortsname in einer URL zusammengesetzt. Anschliessend wird ein neues HandleXML-Objekt erzeugt, welches für das Parsen der XML-Datei zuständig ist und die Ressourcen werden in der Anwendung angezeigt.

```
public void open(View view){
    String url = location.getText().toString();
    String finalUrl = url1 + url + url2;
    country.setText(finalUrl);
    obj = new HandleXML(finalUrl);
    obj.fetchXML();
    while(obj.parsingComplete);
    country.setText(obj.getCountry());
    temperature.setText(obj.getTemperature());
    humidity.setText(obj.getHumidity());
    pressure.setText(obj.getPressure());


}
```

Figure 6: open-Methode der Datei 'MainActivity.java'

Die Anwendung wurde durch Eingabe von 20 verschiedenen Städtenamen getestet. Die resultierenden Werte wurden mit den Werten auf der Webseite verglichen. Das Ergebnis erwies sich als positiv. Alle Werte stimmen überein, sodass die Anwendung ihren Zweck erfüllt. Dieses Proof-Of-Concept konnte erfolgreich abgeschlossen werden.

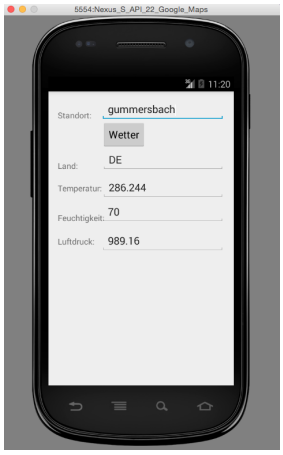

Figure 7: Anwendung in Ausführung - Erhalt der Wetterdaten

1.3 Kommunikation über Google Cloud Messaging

Um die Kommunikation der Nutzer untereinander zu gewährleisten soll eine Google Cloud Messaging API eingebunden werden. Dieser Proof-Of-Concepts soll über Google Cloud Messaging realisiert werden. Für diese Realisierung musste zunächst ein Projekt über die Google API Console**?** registriert werden. Nach dem Erstellen eines Projekts bekommt man eine Projektnummer. Diese Nummer braucht man im späteren Verlauf für die Clients. Weiterhin muss man die gewünschte API aktivieren die über **APIs u.auth.-¿Google Cloud Messaging for Android**zu erreichen ist. Außerdem braucht man einen Server-Schlüssel, der erstellt werden muss über **APIs u. auth -¿Zugangsdaten Öffentlicher API-Zugriff-¿Server Schlüssel erstellen**.

Über developer.android.com bekommt man eine ausführliche Anleitung fürs Implementieren von Server und Client. Es wird über OpenSourceFile die Source Dateien bereitgestellt die frei zugänglich sind.

Leider ist das Proof of Concept nicht erfüllt, da die gewünschten Funktionen nicht realisiert werden konnten. Das Problem ist darauf zurück zu führen, das mehrere Fehler aufgetreten sind und diese nicht korrekt aufgehoben werden konnten.