Dokumentation Proof Of Concept #11 – Einbindung eines bestehenden Mapping Dienstes

Dieser PoC verfolgt das Ziel einen externen Kartendiens, der über eine Reihe von Kernfunktionalitäten verfügen muss, in eine Android App zu integrieren. Diese Funktionalitäten bestehen zum einen im Tagging von Positionen (im Projekt : Lebensmittelagenboten) auf Karten sowie der Berechnung von benötigter Zeit für das Zurücklegen einer Strecke mit verschiendenen Verkehrsmitteln. Auf Basis der Funktionalitäten dieses Kartendienstes soll mobilen Clients außerdem eine Übersicht über Lebensmittelangebote und ihre Standorte ermöglicht werden.

Als erste Wahl für diesen Dienst wurde der Mapbox Service, eine Open Source Implementierung auf Basis von Leaflet(OpenStreetMap Contributor) beschlossen. Dies liegt zum einen an der Quelloffenheit dieser Lösung, zum anderen an den verfügbaren SDKs für die gewählten Zielplattformen. Zudem bietet Mapbox sowohl mit dem Maps,Directions und Geocoding Dienst alle benötigten Funktionalitäten.

Bei der Durchführung dieses PoC wurde zunächst die Android Studio IDE sowie diverse Teile des Android SDK eingerichtet. Da einige Android-Geräte im Projektteam vorhanden waren konnten diese für Debuggingzwecke in Android Studio verwendet werden. Nach einem vom Hersteller angebotenen First Steps Tutorial[1] wurde im nächsten Schritt der Versuch unternommen eine Karte des Dienstes in eine App zu integrieren. Dieses Tutorial beginnt mit der Installation von Mapbox über das in Android Studio verwendete Build-Management Tool „Gradle“. Zur Durchführung dieses PoC wurde das „latest stable Build“ (Version 2.2.0) verwendet.Die Installation geschieht durch Einbinden der folgenden Zeilen :

```
repositories {
    mavenCentral()
}
dependencies {
   compile ('com.mapbox.mapboxsdk:mapbox-android-sdk:2.2.0@aar'){
 transitive=true    }
 }
```

in das modulspezifische gradle-file.
Pfad: ..AndroidProjects > <Projektname> > App > build.gradle

Die wesentlichen Schritte umfassen dann das generieren eines API Keys, die Zusage diverser Permissions im App Manifest.xml sowie die Einbindung eines Mapviews.Mithilfe des API Keys werden Aufrufe der API einem Account zugeordnet werden und ab einer gewissen Anzahl in Rechnung gestellt. Zum Einbinden des Mapviews muss das zuvor über die Accountverwaltung des Anbieters generierte API-Access Token in einem beliebigen Android-Layout.xml angegeben werden.

```
<com.mapbox.mapboxsdk.views.MapView
   android:id="@+id/mapboxMapView"
   android:layout_width="match_parent"
   android:layout_height="match_parent"/
mapbox:access_token="@string/accessToken"
```

Pfad:
../AndroidStudioProjects> <Projektname> >app>src>main>res>layout>activity_layout.xml

@string/accessToken ist hier der Name einer String Ressource,die im File Strings.xml wie folgt definiert wurde:

```
<resources>
   <string name="app_name">PoC_Mapping</string>
   <string name="action_settings">Settings</string>
   <string name="accessToken">HIER STEHT DAS API ACCESS-TOKEN</string>
</resources>
```

Pfad : ..>AndroidStudioProjects> <Projektname> > app > src > main > res > values > strings.xml

Der so definierte Mapview kann dann in den einzelnen Android Acitivites angesprochen werden.

```
private MapView mapView = null;
/** Wird beim Start der Activity gefeuert */
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

/** Finde xml Mapview element */
mapView = (MapView) findViewById(R.id.mapview);

/** Tagge Location */
mapView.setStyleUrl(Style.MAPBOX_STREETS);
mapView.setCenterCoordinate(new LatLng(41.885, -87.679));
mapView.setZoomLevel(11);
 mapView.onCreate(savedInstanceState);    }
```

Diese wenigen Schritte sollten die im PoC defnierten Exit-Kriterien bereits erfüllen. Tatsächlich führten sie zu einer Situation, in denen schon die in Android Studio integrierte Layout-Preview nicht in der Lage war die Map darzustellen[2]. Nach Ablauf der zur Durchführung dieses PoCs eingeplanten Arbeitszeit wurde zunächst die Fallbackoption implementiert.

Diese Option bestand in der Nutzung der Google Maps;- und Google Directions API, die ebenfalls den angestrebten Funktionsumfang (Maps, Directions, Umrechnung von Standort zu Adressdaten) bereitstellen. Um die Verwendung dieser APIs zu ermöglichen musste auch bei Google zunächst ein Projekt angelegt und ein API Key generiert werden. [3]
Zur Standortlokalisierung wird seit Android API Level 23 die Nutzung der Google Play Services empfohlen, dieser SDK-Bestandteil wurde im ersten Schritt nicht verwendet und musste in Android Studio über Tools > Android > SDK Manager > SDK Tools > Play Services

nachinstalliert werden.

Weitere Schritte richteten sich nach einem von Google angebotenen Leitfaden [4]. Die Einbindung einer Map erfolgte weitestgehend problemlos, was einer Lückenlosen Dokumentation zu verdanken ist. Die Ermittlung des Gerätestandortes über Google Play Services konnte ebenfalls problemlos implementiert werden. Dazu wurde in der onCreate-Methode der Main Activity ein ApiClient erzeugt:

```
//Instanziiere Client für Google Play Services API
 GoogleApiClient mGoogleApiClient = new GoogleApiClient.Builder(this.getBaseContext())
       .addConnectionCallbacks(this)
       .addOnConnectionFailedListener(this)
       .addApi(LocationServices.API)
       .build();
```

und der Verbindungsaufbau über ein Button-Onclickevent angestoßen.

```
//Onclick für den Button der den Verbindungsaufbau zu GooglePlay Services auslöst
btnTraceGeo.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      Log.d("Buttonclick", "TraceGeoButton Clicked");
      mGoogleApiClient.connect();
   }
});
```

Bei erfolgreicher Verbindung wird der ermittelte Standort in einem Textview angezeigt und ein Marker auf der eingebundenen Map platziert. Zudem wird der sichtbare Kartenausschnitt auf den ermittelten Standort verschoben.

```
//Triggered bei erfolgreichem Verbindungsaufbau über den GoogleApiClient
@Override
public void onConnected(Bundle bundle) {
   letzterStandort =
LocationServices.FusedLocationApi.getLastLocation(this.mGoogleApiClient);

   if (letzterStandort != null) {
      laengengrad=letzterStandort.getLongitude();
      breitengrad=letzterStandort.getLatitude();
      tvGeoLoc.setText("Akuteller Standort: " + String.valueOf(laengengrad) + "°N und " +
String.valueOf(breitengrad) + "°W");

      LatLng demolocation = new LatLng(breitengrad,laengengrad);
      mMap.addMarker(new MarkerOptions().position(demolocation).title("Standort
während Demo"));
      mMap.moveCamera(CameraUpdateFactory.newLatLng(demolocation));
   }
}
```

Die Fallbackoption für diesen PoC erfüllt damit das Exit Kriterium (erfolgreiches Tagging eines Standorts auf einer in der App eingebundenen Karte).

Quellen

[1] Mapbox Android SDK erste Schritte
https://www.mapbox.com/guides/first-steps-android-sdk
(Zuletzt eingesehen 05.11.15)

[2] Aufgetretener Bug
https://github.com/mapbox/mapbox-gl-native/issues/2830
(Zuletzt eingesehen 05.11.15)

[3] Genauere Anleitung zur Einrichtung eines Google Projects
https://developers.google.com/maps/documentation/android-api/signup (Zuletzt eingesehen 06.11.2015)

[4]
https://developers.google.com/maps/documentation/android-api/start
(Zuletzt eingesehen 06.11.2015)