**Routen**

Das Erstellen und Bearbeiten von Routen ist ein integraler Bestandteil der Veranstalter-App. Dem Nutzer sollte es möglich sein auf intuitive Weise Strecken zu erstellen ohne diese selber abzulaufen oder umständlich manuell auf der Karte zu zeichnen. Stattdessen sollte die Strecke möglichst automatisch über die bestehenden Straßen gelegt werden. Da das Erstellen einer Strecke eine einzelne unabhängige Aktivität darstellt, wird mit der *RouteEditorActivity* eine eigene [Activity](http://developer.android.com/reference/android/app/Activity.html) bereitgestellt die sich allein mit dieser Aufgabe beschäftigt. Diese Activity beinhaltet einen [ViewPager](http://developer.android.com/reference/android/support/v4/view/ViewPager.html) welcher zwei [Fragments](http://developer.android.com/reference/android/app/Fragment.html) verwaltet zwischen denen sowohl über die Tab Leiste, als auch durch horizontales wischen gewechselt werden kann. Verwaltet werden das *EditorMapFragment*, welches die Darstellung und Bearbeitung der Strecke auf der Karte ermöglicht und das *EditorWaypointsFragment*, welches die Bearbeitung einzelner Wegpunkte in der Liste ermöglicht. Die Fragment Klassen beinhalten lediglich die Methoden zur Abstraktion des UI welche von der eigentlichen Programmlogik in der *RouteEditorActivity* aufgerufen werden.

Von der *RouteEditorActivity* verwaltete Daten werden hauptsächlich durch zwei statische innere Klassen repräsentiert. Diese kombinieren die von der „Google Maps Android API v2“ gestellten Klassen zur Darstellung der Daten auf der Strecke mit zusätzlichen Metainformationen. Um die Verwendung dieser Klassen zu erleichtern implementieren sie das Parcelable Interface. Somit können sie beispielsweise in der *onSaveInstanceState(Bundle)* problemlos im Bundle hinterlegt werden. Referenzen auf UI Elemente werden jedoch nicht gespeichert (siehe Abschnitt „Google Maps Android API v2“). Die erste dieser Klassen ist die *RouteEditorActivity.Route*, welche den Streckenverlauf der berechneten Route darstellt. Sie kombiniert die von Google Maps genutzten *PolylineOptions* und *Polyline* mit der als *int* gespeicherten Länge der Strecke in Metern. Die zweite der Klassen ist die *RouteEditorActivity.Waypoint*, welche einen einzelnen Wegpunkt auf der Strecke darstellt. Diese bestehen aus den von Google Maps genutzten *MarkerOptions* und *Marker*.

**Google Maps Android API v2**

Die Google Maps Android API v2 ist Teil der Google Play Services und bietet die Möglichkeit einfach zu verwendende, interaktive Karten in Android Apps zu verwenden. Kernstück der API ist die GoogleMap Klasse, welche Interaktion mit der dargestellten Karte ermöglicht. Zugriff erfolgt wahlweise über die *getMap()* Methoden der *MapView* oder *MapFragment* Klassen, welche zur Darstellung der Karte verwendet werden. Da die Karte Zugriff auf den Activity-Lifecycle benötigt, wird von Google die Verwendung des MapFragment empfohlen. Die GoogleMap stellt eine Vielzahl von Möglichkeiten bereit die Karte der erforderten Nutzung anzupassen. Insgesamt werden 6 mögliche Varianten für Kartenannotationen bereitgestellt: Circle, GroundOverlay, Marker, Polygon, Polyline und TileOverlay. Für die Planung der Route benötigen wir jedoch lediglich Marker für Wegpunkte und eine Polyline für die Strecke. Diese Annotationen bestehen jeweils aus zwei Teilen. So repräsentiert die MarkerOptions Klasse Anweisungen um einen Marker auf der Karte darstellen zu können. Um diesen tatsächlich darzustellen wird die *addMarker(MarkerOptions)* Methode der GoogleMap aufgerufen. Diese liefert eine Referenz auf das gezeichnete Marker Objekt zurück. Da die MarkerOptions lediglich die Anweisungen zum Zeichnen des Markers repräsentieren, müssen Änderungen wie z.B. ein neuer Titel an das Marker Objekt übermittelt werden. Da die *addMarker(MarkerOptions)* die einzige Möglichkeit darstellt auf dieses zuzugreifen, ist es wichtig das Objekt für eine mögliche spätere Verwendung zu behalten. Zudem muss bedacht werden, dass die Marker nur zeitlich begrenz gültig sind. Wird z.B. durch drehen des Gerätes das UI neu gezeichnet, werden die Marker ungültig. Für diesen Fall sollten die MarkerOptions gespeichert werden um die erneute Darstellung zu ermöglichen. Dieses Konzept überträgt sich auf die anderen Annotationstype z.B. mit *PolygonOptions* und *Polygon*, *PolylineOptions* und *Polyline*, usw. Es