```java
package fr.insalyon.dasi.gustatif.util;

import com.google.maps.DirectionsApi;
import com.google.maps.DirectionsApiRequest;
import com.google.maps.GeoApiContext;
import com.google.maps.GeocodingApi;
import com.google.maps.model.DirectionsResult;
import com.google.maps.model.DirectionsRoute;
import com.google.maps.model.GeocodingResult;
import com.google.maps.model.LatLng;
import com.google.maps.model.TravelMode;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author DASI Team
 */
public class GeoTest {
        final static String MA_CLÉ_GOOGLE_API =
        "AIzaSyAQ1rgsSDdetI6uhC9egwf_OqdDprHwB-g";

//    final static String MA_CLÉ_GOOGLE_API =
"AIzaSyAhf3JleYpal9S-xouJYH8lf7Dvz5Y2Nko";


    final static GeoApiContext MON_CONTEXTE_GEOAPI = new
    GeoApiContext().setApiKey(MA_CLÉ_GOOGLE_API);

    public static LatLng getLatLng(String adresse) {
        try {
            GeocodingResult[] results =
            GeocodingApi.geocode(MON_CONTEXTE_GEOAPI, adresse).await();

            return results[0].geometry.location;

        } catch (Exception ex) {
            return null;
        }
    }

    public static double toRad(double angleInDegree) {
        return angleInDegree * Math.PI / 180.0;
    }

    public static double getFlightDistanceInKm(LatLng origin, LatLng
    destination) {

        // From: http://www.movable-type.co.uk/scripts/latlong.html
        double R = 6371.0; // Average radius of Earth (km)
        double dLat = toRad(destination.lat - origin.lat);
        double dLon = toRad(destination.lng - origin.lng);
        double lat1 = toRad(origin.lat);
        double lat2 = toRad(destination.lat);
```

```
50
51          double a = Math.sin(dLat / 2.0) * Math.sin(dLat / 2.0)
52                  + Math.sin(dLon / 2.0) * Math.sin(dLon / 2.0) *
                    Math.cos(lat1) * Math.cos(lat2);
53          double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1.0 - a));
54          double d = R * c;
55
56          return d;
57      }
58
59      public static Double getTripDurationByBicycleInMinute(LatLng
        origin, LatLng destination, LatLng... steps) {
60          return getTripDurationOrDistance(TravelMode.BICYCLING, true,
            origin, destination, steps);
61      }
62
63      public static Double getTripDistanceByCarInKm(LatLng origin,
        LatLng destination, LatLng... steps) {
64          return getTripDurationOrDistance(TravelMode.DRIVING, false,
            origin, destination, steps);
65      }
66
67      public static Double getTripDurationOrDistance(TravelMode mode,
        boolean duration,
68              LatLng origin, LatLng destination, LatLng... steps) {
69
70          DirectionsApiRequest request =
            DirectionsApi.getDirections(MON_CONTEXTE_GEOAPI,
71                  origin.toString(), destination.toString());
72          request.mode(mode);
73          request.region("fr");
74
75          if (steps.length > 0) {
76
77              String[] stringSteps = new String[steps.length];
78              for (int i=0; i<steps.length; i++) {
79                  stringSteps[i] = steps[i].toString();
80              }
81
82              request.waypoints(stringSteps);
83          }
84
85          double cumulDistance = 0.0;
86          double cumulDuration = 0.0;
87
88          try {
89              DirectionsResult result = request.await();
90              DirectionsRoute[] directions = result.routes;
91
92              for (int legIndex = 0; legIndex <
                directions[0].legs.length; legIndex++) {
93
94                  cumulDistance +=
                    directions[0].legs[legIndex].distance.inMeters / 1000.0;
```

```
95                      cumulDuration +=
                        Math.ceil(directions[0].legs[legIndex].duration.inSecond
                        s / 60.0);
96                  }
97
98              } catch (Exception ex) {
99                  return null;
100             }
101
102             if (duration) {
103                 return cumulDuration;
104             }
105             else {
106                 return cumulDistance;
107             }
108     }
109
110     public static void main(String[] args) {
111         Logger logger = Logger.getLogger(GeoApiContext.class.getName());
112         logger.setLevel(Level.WARNING);
113
114         String adresse1 = "7 Avenue Jean Capelle Ouest, Villeurbanne";
115         LatLng coords1 = getLatLng(adresse1);
116         System.out.println("Lat/Lng de Adresse #1: " + coords1);
117
118         String adresse2 = "37 Avenue Jean Capelle Est, 69100
            Villeurbanne";
119         LatLng coords2 = getLatLng(adresse2);
120         String adresse3 = "61 Avenue Roger Salengro, Villeurbanne";
121         LatLng coords3 = getLatLng(adresse3);
122
123         Double duree = getTripDurationByBicycleInMinute(coords1,
            coords3, coords2);
124         System.out.println("Durée de Trajet à Vélo de Adresse #1 à
            Adresse #3 en "
125                 + "passant par Adresse #2: " + duree + " min");
126
127         Double distance = getTripDistanceByCarInKm(coords1, coords3);
128         System.out.println("Distance en Voiture de Adresse #1 à
            Adresse #3 "
129                 + "(trajet direct par la route): " + distance + " km");
130
131         Double distanceVolDOiseau = getFlightDistanceInKm(coords1,
            coords3);
132         System.out.println("Distance à Vol d'Oiseau de Adresse #1 à
            Adresse #3"
133                 + " (distance géographique): " + distanceVolDOiseau +
                    " km");
134     }
135 }
136
```