Notas SPOT

Limites aproximados de Capital Federal:
———————————

Arriba a la izquierda: Latitude: -34.525793 | Longitude: -58.551292
Abajo a la derecha: Latitude: -34.707751 | Longitude: -58.319206

Llamada a API de OSM (GET /api/0.6/map?bbox=left,bottom,right,top):
/api/0.6/map?bbox=-58.551292, -34.707751, -58.319206, -34.525793

Llamada a Overpass API:
Or bounding box clauses (51.0,7.0,52.0,8.0)
Bounding box clauses always start with the lower latitude followed by lower longitude, then upper latitude then upper longitude. Note that this is different from the ordering in the XAPI syntax. The XML syntax is safeguarded by using named parameters.

http://overpass-turbo.eu/#

http://wiki.openstreetmap.org/wiki/Key:highway#Roads

Estaciones de bondis, pero no todas!
node(-34.707751,-58.551292,-34.525793,-58.319206)[highway=bus_stop];
out;
node(-34.707751,-58.551292,-34.525793,-58.319206)[public_transport=platform];out body;

Calles!
way(-34.707751,-58.551292,-34.525793,-58.319206)[highway=residential];
(._;>;);
out;

recorridos
relation(-34.707751,-58.551292,-34.525793,-58.319206)[ref="130M"];out body;

PARADAS Y RECORRIDOS

http://overpass-api.de/api/interpreter?data=(%20node%20[public_transport=platform]%20(-34.707751,-58.551292,-34.525793,-58.319206);%20rel%20[route=bus]%20(-34.707751,-58.551292,-34.525793,-58.319206);%20);%20(._;%3E;);%20out;

(
node
[public_transport=platform]
(-34.707751,-58.551292,-34.525793,-58.319206);
rel
[route=bus]
(-34.707751,-58.551292,-34.525793,-58.319206);

);
(._;>;);
out;

Objetivo: avisar al pasajero rapidamente la calle que viene en base a su vector de direccion
Necesitamos:
- puntos que forman la polilinea que representa la calle (de ahora en adelante ”calle”)
- vector de direccion del usuario. GPS + brujula digital
Algoritmo:
1- Identificar posicion y direccion del usuario
2- Identificar calle por la que va el usuario
3- Identificar calles cortadas por la calle por la que va el usuario

La informacion de calles es muy estatica

ESQUEMA DE DATOS
Habria que tener una base de calles donde cada calle tiene
- una serie de puntos geograficos que forman la polilinea de la ruta
- una serie de puntos geograficos que representan cruces con otras calles, donde cada uno de estos puntos indica que calle es.

¿Como obtener estos datos?
1- Sacar de OSM todas las lineas correspondientes a calles de la ciudad -> verificar con Overpass Turbo
Bajar todo esto a base.
2- Preprocesamiento de la data de Open Street Map que incluye
Por cada polilinea correspondiente a una calle, indentificar cuales otras polilineas la atraviesan y donde
(Punto de contacto entre polilineas)
Por cada segmento de polilinea ver si se cruza con todos los otros segmentos de la otra polilinea (=> interseccion entre dos lineas) RESUELTO EN (*)

Pasaje de datos desde OSM a base de datos
Revisar ”Osmosis”

Prueba de factibilidad:
Procesamiento en el telefono
Visualización en el telefono en forma de lista de calles.
Pruebas con gps fake

Motor de deteccion de subida y bajada de bondi
Armar prototipo para ”salir a probar”
- Trazado de recorrido en google maps (Tracking GPS)
http://blog.teamtreehouse.com/beginners-guide-location-android
- Marcado de cambios de estado en mapa de google (Activity Recognition API)

---

distnacia entre puntos gps, y por las calles?
http://www.movable-type.co.uk/scripts/latlong.html

(*)
Codigo:

```
public class test {

public static void main(String[] args) {
System.out.println("Hola");

boolean hayInter = true;

//Line A = new Line(-34.579525, -58.432980, -34.578438, -58.431639);
//Line B = new Line(-34.578367, -58.432336, -34.579242, -58.431210);

Line A = new Line(-34.578438, -58.431639, -34.579525, -58.432980);
Line B = new Line(-34.578367, -58.432336, -34.579242, -58.431210);

double XAsum = A.getFirstLong() - A.getLastLong();
double XBsum = B.getFirstLong() - B.getLastLong();
double YAsum = A.getFirstLat() - A.getLastLat();
double YBsum = B.getFirstLat() - B.getLastLat();

double LineDenominator = XAsum * YBsum - YAsum * XBsum;
if(LineDenominator == 0.0)
hayInter = false;

double a = A.getFirstLong() * A.getLastLat() - A.getFirstLat() * A.getLastLong();
double b = B.getFirstLong() * B.getLastLat() - B.getFirstLat() * B.getLastLong();

double x = (a * XBsum - b * XAsum) / LineDenominator;
double y = (a * YBsum - b * YAsum) / LineDenominator;

System.out.println(x + " " + y );
}
}
```

Salida:

```
public class Line {

private double firstLat;
private double firstLong;

private double lastLat;
private double lastLong;


public Line(double firstLat, double firstLong, double lastLat,
double lastLong) {
```

```
super();
this.firstLat = firstLat;
this.firstLong = firstLong;
this.lastLat = lastLat;
this.lastLong = lastLong;
}

public double getFirstLat() {
return firstLat;
}
public void setFirstLat(double firstLat) {
this.firstLat = firstLat;
}
public double getFirstLong() {
return firstLong;
}
public void setFirstLong(double firstLong) {
this.firstLong = firstLong;
}
public double getLastLat() {
return lastLat;
}
public void setLastLat(double lastLat) {
this.lastLat = lastLat;
}
public double getLastLong() {
return lastLong;
}
public void setLastLong(double lastLong) {
this.lastLong = lastLong;
}

}
```

Graficado de mapas simples (a la vista)
* En vez de graficar calles queremos graficar manzanas
Podemos usar los datos de esquinas y puntos que forman una calle para graficar cada manzana diferente
* Necesitamos una pantalla donde graficar lineas (posiblemente mediante Java2D)

Montar un servidor de Tiles o de datos?
Alternativas:
* Montar un servidor de tiles de un mapa simplificado
* Enviar la data de puntos de calles al telefono
* Envian al telefono los puntos que forman las manzanas en lugar de las calles
* Envian datos de calles pero se redondean las esquinas

Que informacion necesito?

- Ruta recorrida
- Marcas de [tiempo - espacio - evento] generadas por el usuario
- Marcas de [tiempo - espacio - evento] generadas por el motor

Formato de registro
Timestamp Lat Long Activity(*)

(*) Puede que sean varios datos diferentes

Interfaz de usuario:
boton para ”Iniciar” y ”Detener”, lo que define los limites de cada prueba.
Al ”Iniciar” se abre el csv
El cierre de ciclo que culmina con ”Detener”...

Librerias especiales de Logueo para Android
https://github.com/tony19/logback-android
http://logback.qos.ch/manual/layouts.html

Server Base de Datos

Server

- IP: 200.32.8.33
- Windows 2012 R2
- User: Administrator
- Pass: Hackathon1234
- Puerto RDP: 33389

Base de Datos Postgresql

- Puerto: 35432
- User: postgres
- Pass: Hackathon1234
- Base: colectivo

TABLAS Y VISTAS

- Nodes: es una tabla bajada directamente de OpenStreetMaps con información de nodos (vinculados a transporte público) de la Ciudad de Buenos Aires. Los nodos tienen un ID, y un campo de geometría (propio de Postgis)

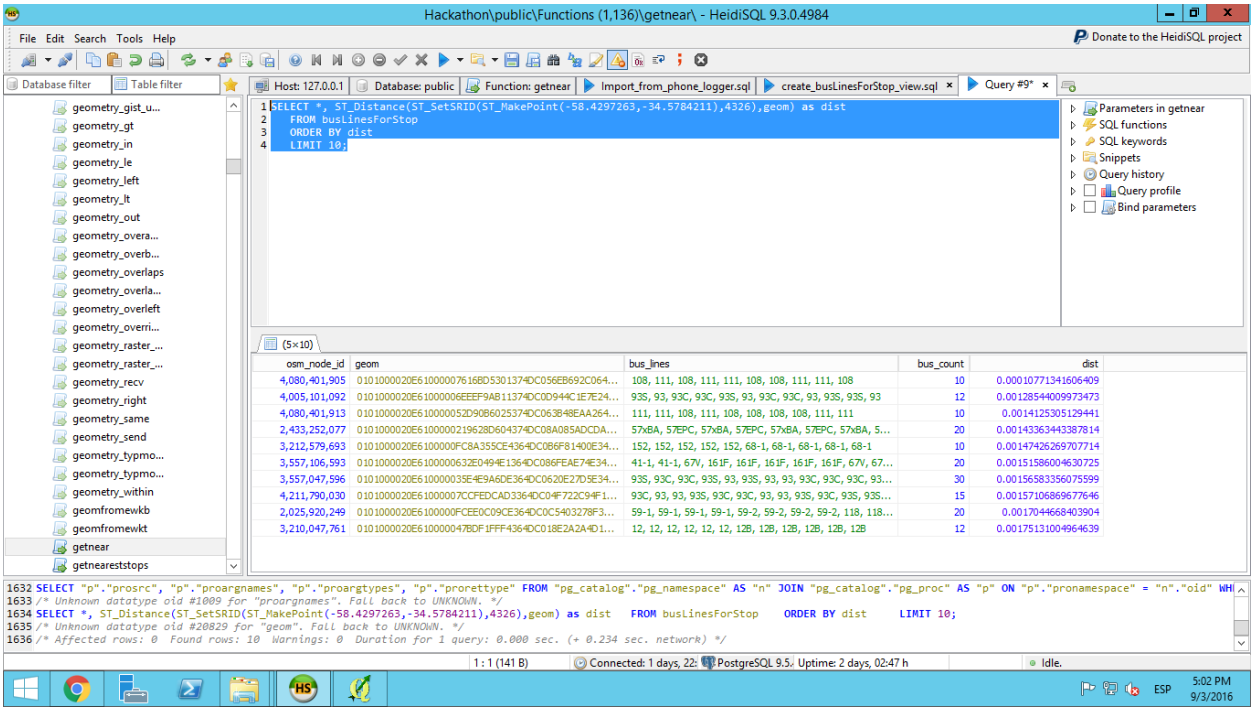
- Rels: es otra tabla bajada de OpenStreetMaps con información de las relaciones. Las relaciones toman nodos (nodes), caminos (ways) y campos sin tipo para formar los recorridos de los colectivos.
- busStops: vista con el id de la parada y la línea a la que corresponde, junto con el campo de geometría (punto)
- busLinesForStop: vista con id de cada parada, pero agrupadas las líneas de colectivo que paran en la misma parada.
- tracking: tabla con los registros del logger que tengo instalado en el celular. Contiene la posición, velocidad, precisión (también dice si es un dato de GPS o estimado de la red celular). También tiene el campo geometría (punto).

SCRIPTS

Es clave aprovechar las funciones de postgis que ya contienen mucho respecto a cuestiones espaciales. Por ejemplo:
Obtener las **10** paradas más cercanas a una posición:

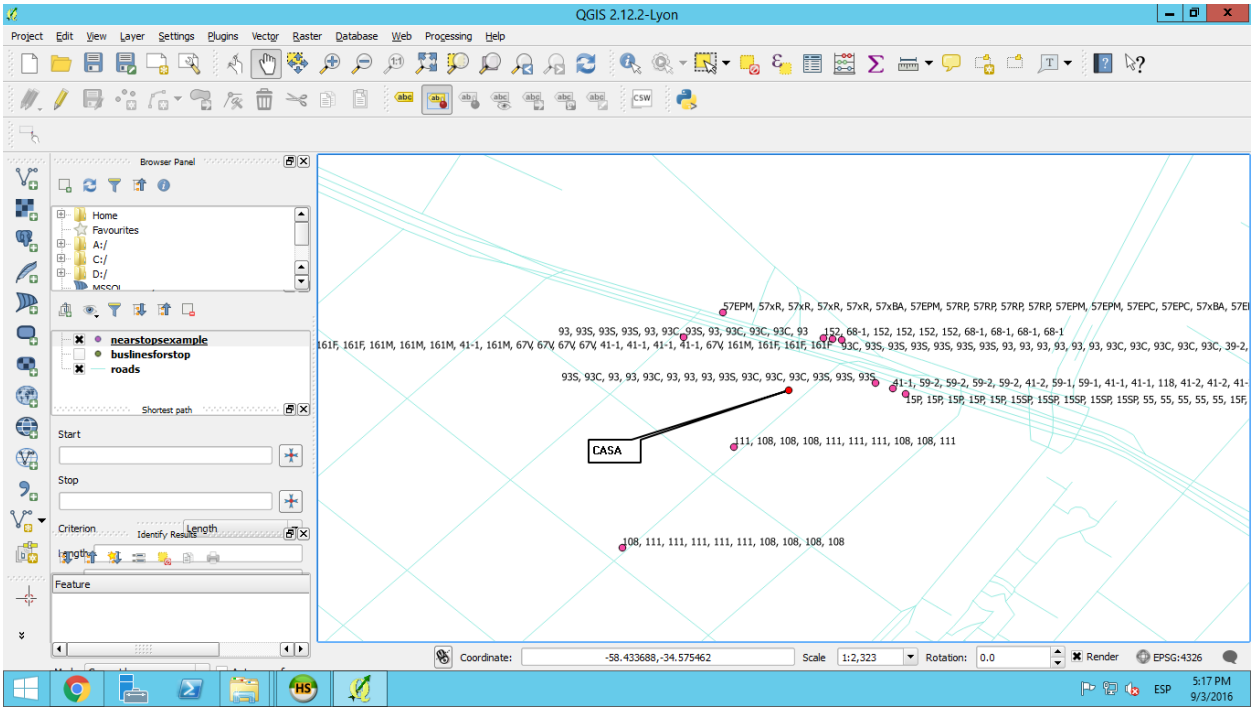
```
SELECT *, ST_Distance(ST_Transform(ST_SetSRID(ST_MakePoint(-58.4297263,-34.5784211),4326),26986),
ST_Transform(geom,26986)) as dist
FROM busLinesForStop
ORDER BY dist
LIMIT 10
```

Donde se puede cambiar la posición de acuerdo al usuario cambiando los parámetros de la función “ST_MakePoint”. Esta función y ST_Distance son de postgis.



Como pueden ver se hizo sobre la vista “busLinesFromStop” con lo cual la columna “bus_lines” tiene en formato texto la agrupación de todas las líneas. El problema (o no!) de OSM es que no

están normalizadas las líneas de colectivo. Con lo cual hay como 40 versiones de la línea 60.



Repo Github: https://github.com/jim884/SPOT