3AA
COURET Nathanaël
TABOULOT Clément

HAGENBOURGER Maxime
CHENAL Vincent

TP1 SIG

1) select COUNT(*) from users;

```
osm=> select COUNT(*) from users;
 count
-------
  4576
(1 row)
```

2)

a) select * from nodes where id=1787038609;
select ST_X(geom), ST_Y(geom) from nodes where id=1787038609;
st_x : 5.7680106
st_y : 45.192893

```
osm=> select ST_X(geom), ST_Y(geom) from nodes where id=1787038609;
   st_x    |   st_y
-----------+-----------
 5.7680106 | 45.192893
(1 row)
```

b) select * from spatial_ref_sys where srid = (select ST_SRID(geom) from nodes where id = 1787038609;)
Le SRS est EPSG 4326, soit WGS 1984.

```
osm=> select * from spatial_ref_sys where srid = (select ST_SRID(geom) from nodes where id = 1787038609);
 srid | auth_name | auth_srid |                                                                                srtext
                                                                                        |              proj4text
------+-----------+-----------+------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------+------------------------------------
 4326 | EPSG      |      4326 | GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORI
TY["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]] | +proj=longlat +datum=WGS84 +no_defs
(1 row)
```

3) select ST_X(ST_Centroid(linestring)), ST_Y(ST_Centroid(linestring)) from ways where tags->'amenity' = 'townhall' and tags->'name' LIKE '%Grenoble%';

```
osm=> select ST_X(ST_Centroid(bbox)), ST_Y(ST_Centroid(bbox)) from ways where tags->'amenity' = 'townhall' and tags->'name' LIKE '%Grenoble%';
    st_x    |    st_y
------------+------------
 5.73644115 | 45.18644215
(1 row)
```

4) select tags->'highway', count(*) as "amount" from ways where exist(tags,'highway') group by tags->'highway' order by amount desc;

5)

a) select tags->'highway', ST_Length(linestring) as "length" from ways where exist(tags,'highway') order by length desc;

```
osm=> select tags->'highway', ST_Length(linestring) as
esc;
         ?column?         |        length
--------------------------+----------------------
 tertiary                 |    0.308131482574593
 tertiary                 |     0.30584280718474
 corridor                 |    0.290625783209972
 primary                  |    0.289584000121336
 secondary                |    0.279611294930457
 secondary                |    0.264392113375508
 secondary                |    0.252342872000858
 primary                  |    0.250827371988434
 secondary                |    0.244342536177091
 secondary                |     0.24160754152937
 secondary                |    0.233450226543954
 motorway                 |    0.231224913631015
 secondary                |    0.230617408347474
 secondary                |     0.22866168362643
 secondary                |    0.224610332663099
```

NB: Tous les résultats ne sont pas affichés.

b) valeurs exprimées en degrés d'après le référentiel spatial WGS 1984.

c) select tags->'highway', ST_Length(ST_GeogFromWKB(ST_AsEWKB(linestring)))/1000 as "length" from ways where exist(tags,'highway') order by length desc;

```
osm=> select tags->'highway', ST_Length(ST_GeogFromWKB(S
ags,'highway') order by length desc;
         ?column?          |        length
---------------------------+----------------------
 tertiary                  |     29.8161800994786
 tertiary                  |     27.1739841662598
 corridor                  |     26.2314453100925
 secondary                 |     25.8188854884837
 primary                   |     25.4719965020426
 secondary                 |     23.6579546730933
 secondary                 |     23.2760853387868
 secondary                 |     22.9089517839896
 primary                   |     22.7476904904507
 secondary                 |     22.5848191817746
 secondary                 |     22.1147891079692
 secondary                 |     21.1474078778018
 secondary                 |     20.6852480898629
 secondary                 |     20.6438066188022
 secondary                 |      20.555165513609
 primary                   |     20.5188608910495
 tertiary                  |     19.9556158515408
 motorway                  |     19.5833361316418
```

NB: Tous les résultats ne sont pas affichés.

6) SELECT SUM(ST_Area(ST_GeogFromWKB(ST_AsEWKB(ST_MakePolygon(linestring))))) AS sqm FROM ways WHERE lower(tags->'name') LIKE lower('%ensimag%');

```
        sqm
-------------------
 3343.02268860344
(1 row)
```

7) select quartier.quartier, count(ways.id) from quartier, ways where ways.tags->'amenity' = 'school' AND ST_Intersects(the_geom, (ST_Transform(linestring,2154))) = TRUE group by quartier.quartier ORDER BY count(ways.id) Desc;

```
       quartier        | count
-----------------------+-------
 BERRIAT ST BRUNO      |    13
 CENTRE VILLLE         |    12
 EXPOSITION-BAJATIERE  |    10
 RONDEAU-LIBERATION    |     7
 ABBAYE-JOUHAUX        |     6
 MALHERBE              |     6
 EAUX CLAIRES          |     6
 ALPINS-ALLIERS        |     5
 JEAN MACE             |     4
 VILLENEUVE2           |     4
 VILLENEUVE1           |     4
 CAPUCHE GR            |     3
 CENTRE GARE           |     2
 VILLAGE-OLYMPIQUE     |     2
 NOTRE DAME            |     1
 TEISSEIRE             |     1
 MISTRAL-DRAC          |     1
 FOCH AIGLE            |     1
 CHAMPIONNET           |     1
 MUTUALITE             |     1
 ILE VERTE             |     1
 POLYGONNE             |     1
 SAINT-LAURENT         |     1
(23 rows)
```

11 et 12)

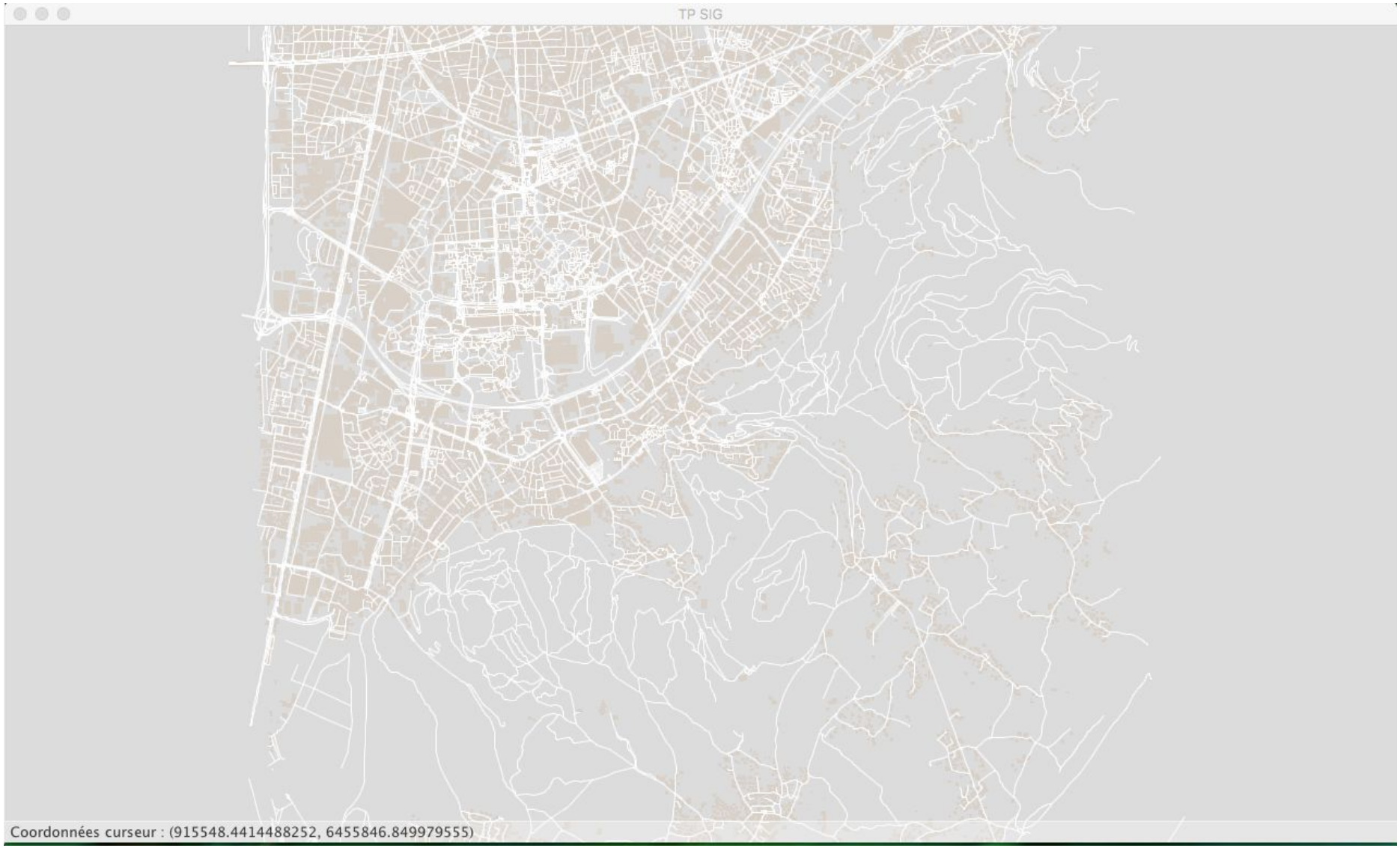
Fonctionnement du programme :

- Il existe une classe Application qui accueille notre méthode Main. On lance donc le programme de cette manière. La classe Application lance ensuite une méthode de la classe Runner en fonction de la présence d'un argument ou non.

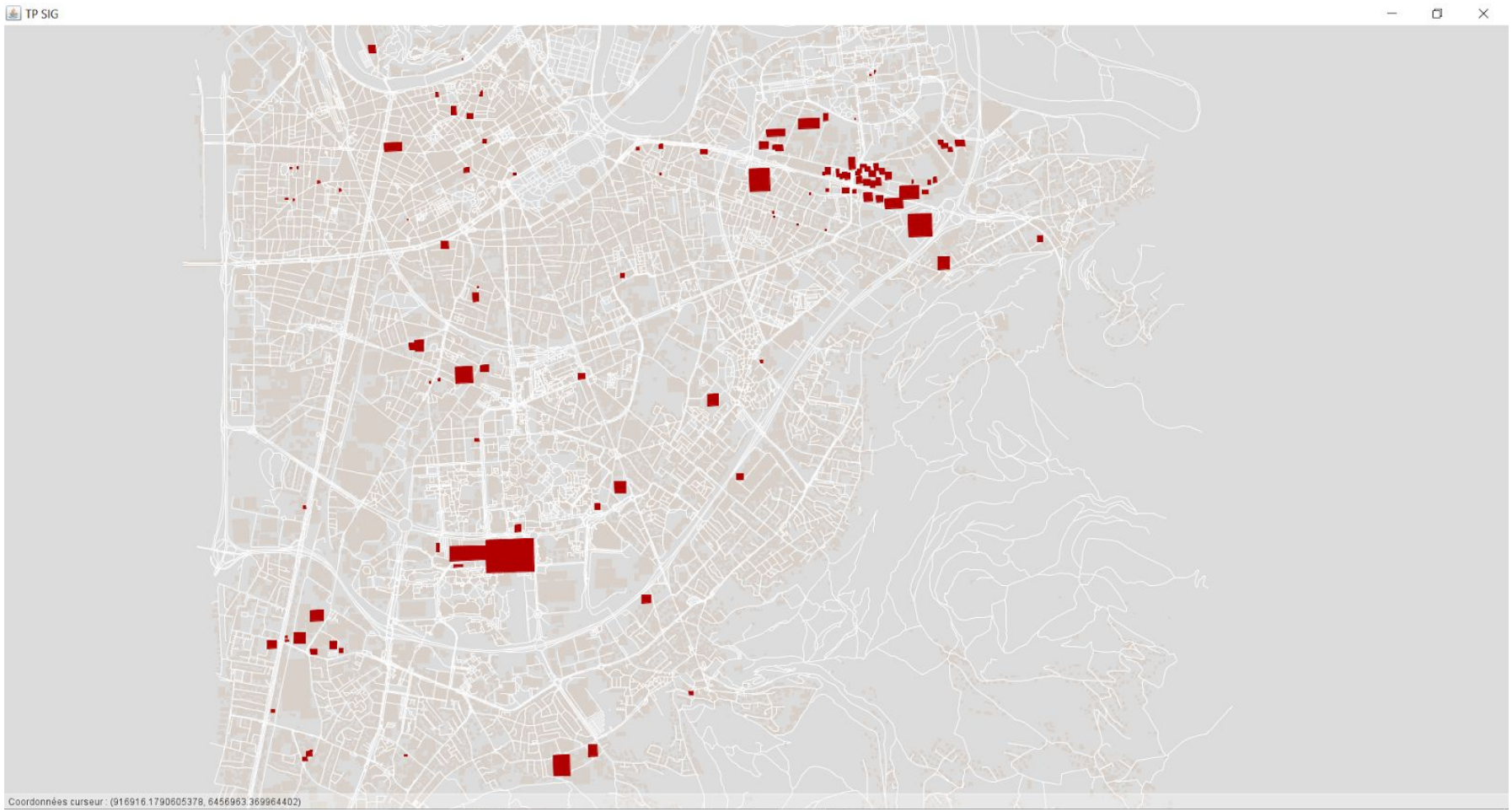
- Dans le cas d'un argument quelconque pour effectuer une recherche, on exécute searchByNameMode() qui va aller chercher les résultats de la recherche correspondant à l'argument fourni. On exécute ensuite la fonction getPositionByName de la classe SigBDD qui en fonction de l'argument fourni retourne le résultat de la recherche. Pour tester notre programme, nous avons décidé de lancer l'application avec l'argument suivant : "%Ensi%". Voici ce que notre application retourne :

```
Nom   | Longitude | Latitude
------+-----------+---------
Ensimag - E | 5.768041 | 45.193516
Ensimag | 5.7683244 | 45.193398
Ensimag - D | 5.768564 | 45.19323
Ensimag - H | 5.7670026 | 45.193092
```

- Pour les cas restants, on va lancer la méthode MapMode de la classe Runner. Elle va ensuite chercher l'ensemble des bâtiments et des routes avec les méthodes drawBuildings() et drawRoads() qui vont communiquer avec la classe SigBDD pour aller chercher les bâtiments et routes correspondantes. Pour la question 12, on obtient la carte de l'ensemble des routes et bâtiments de Grenoble. Voici le résultat :


TP SIG
Coordonnées curseur : (915548.4414488252, 6455846.849979555)

Pour la Q13, on va chercher l'ensemble des magasins de Grenoble et les mettre en évidence. On utilise une constante Q13, déclarée dans le fichier MapMode. Elle est utilisée lors de l'exécution avec le paramètre -q13. Comme pour la question précédente, on va communiquer avec la classe SigBDD qui s'occupe de lancer les requêtes correspondantes grâce aux méthodes drawBuildings, drawRoads et drawStores. Voici le résultat :



Pour la Q14, on va représenter graphiquement la localisation des boulangeries de Grenoble. On utilise encore une constante Q14, déclarée dans le fichier MapMode. Elle est utilisée lors de l'exécution avec le paramètre -q14. Comme pour la question précédente, on va communiquer avec la classe SigBDD qui s'occupe de lancer les requêtes correspondantes grâce aux méthodes drawBuildings, drawRoads et cette fois-ci drawBoulangeries. Voici le résultat :

Requêtes utilisées :

- Récupération des bâtiments :
  SELECT ST_Transform(bbox,2154) FROM ways WHERE ST_X(ST_Centroid(bbox)) BETWEEN <*Long_min*> AND <*Long_max*>AND ST_Y(ST_Centroid(bbox)) BETWEEN <*lat_min*> AND <*lat_max*> AND exist(tags,'building')

- Récupération des routes :
  SELECT ST_Transform(linestring ,2154)FROM ways WHERE ST_X(ST_Centroid(bbox)) BETWEEN <*Long_min*> AND <*Long_max*> AND ST_Y(ST_Centroid(bbox)) BETWEEN <*lat_min*> AND <*lat_max*> AND exist(tags,'highway')

- Récupération des shops :
  select ST_Transform(bbox,2154) from ways where ST_X(ST_Centroid(bbox)) BETWEEN <*Long_min*> *AND* <*Long_max*> AND ST_Y(ST_Centroid(bbox)) BETWEEN <*lat_min*> AND <*lat_max*> AND exist(tags,'shop')

- Récupération de la répartition des boulangerie par quartier :
  select quartier.the_geom, *count*(ways.id) from quartier, ways where ways.tags->'amenity' = 'school' AND ST_Intersects(the_geom, (ST_Transform(linestring,2154))) = TRUE group by quartier.the_geom ORDER BY *count*(ways.id) Desc;

Dans les deux cas, on utilise le centroïde de l'objet concerné pour savoir s'il fait partie de la zone géographique concernée. Les coordonnées retournées sont en Lambert-93 (SRID = 2154). A l'origine les coordonnées sont en WGS84, c'est pour cela qu'il n'y a aucune transformation dans les clauses "WHERE".

Diagramme de classes :

Des diagrammes de classes UML de chacun des packages sont disponible dans le dossier Documentation de la solution.
Pour les visualiser ouvrir le fichier index.html avec votre navigateur préféré.