1. Création de la base de données

- téléchargement de `midi-pyrenees-latest.om.pbf` sur `https://download.geofabrik.de/europe/france/` (On ne s'intéresse qu'au Gers donc Midi-Pyrénées est largement suffisante, je n'ai pas réussi à trouver de carte plus récente (région Occitanie))
- utilisation de osm2pgsql `osm2pgsql -d osm_db -U postgres --create --slim -G --hstore --number-processes 4 midi-pyrenees-latest.osm.pbf`

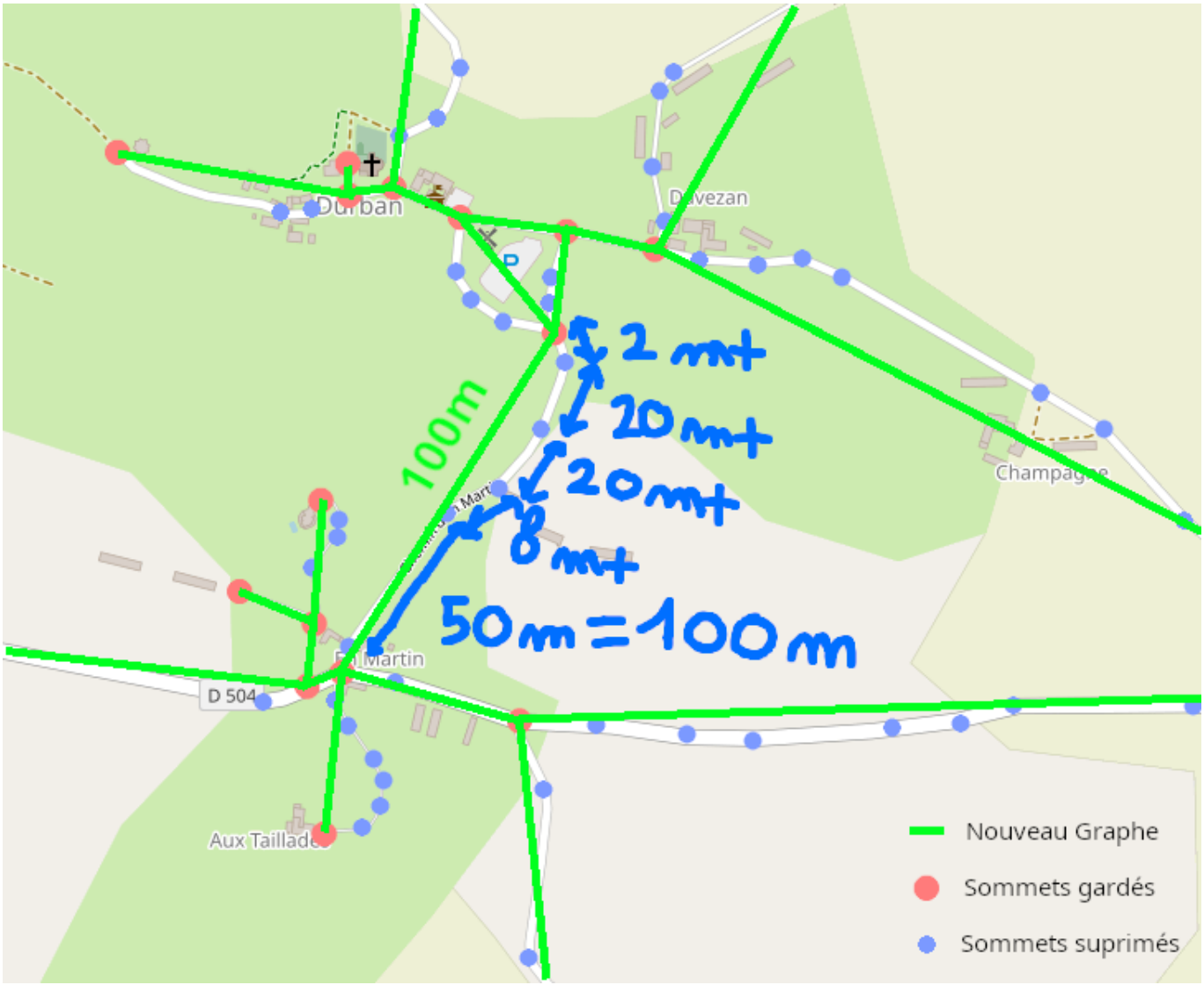
2. Nettoyage de la base de données pour ne garder que les routes praticables en voiture

```
CREATE TABLE filtered_ways AS
SELECT id, nodes, tags
FROM planet_osm_ways
WHERE tags ? 'highway'  -- Vérifie si 'highway' existe
AND tags->>'highway' IN (
    'motorway', 'motorway_link', 'trunk', 'trunk_link',
    'primary', 'primary_link', 'secondary', 'secondary_link',
    'tertiary', 'tertiary_link', 'residential', 'unclassified', 'road'
);
```

3. Réduction du volume de données

Pour réduire le volume de données, je me suis dit qu'il était judicieux que l'application renvoie uniquement des intersections de routes. Cela permet de simplifier la base de données en ne conservant que les intersections de routes (et les culs de sacs) et en combinant les arêtes entre chaque sommet en une seule

arête qui garde les mêmes attributs (vitesse de déplacement, distance, ...)



Pour ce faire, j'ai commencé par créer une table avec les sommets qui ont une seule arête adjacente (cul-de-sac) :

```
CREATE TABLE road_ends AS
WITH node_counts AS (
    SELECT unnest(nodes) AS node_id, COUNT(*) AS degree
    FROM filtered_ways
    GROUP BY node_id
)
SELECT n.id, n.lat, n.lon, n.tags
FROM planet_osm_nodes n
JOIN node_counts nc ON n.id = nc.node_id
WHERE nc.degree = 1;
```

Ainsi qu'une table des sommets qui ont au moins trois arêtes adjacentes (intersections):

```
CREATE TABLE intersections AS
WITH node_counts AS (
    SELECT unnest(nodes) AS node_id, COUNT(*) AS degree
```

```
    FROM filtered_ways
    GROUP BY node_id
)
SELECT n.id, n.lat, n.lon, n.tags
FROM planet_osm_nodes n
JOIN node_counts nc ON n.id = nc.node_id
WHERE nc.degree >= 3;
```

Déjà ici, j'ai un problème, car quand j'affiche le graphe résultant de cette commande, je n'obtiens pas que des intersections :

```
SELECT id, lat * 1e-7 AS lat, lon * 1e-7 AS lon, tags
FROM road_ends
WHERE ST_Intersects(
    ST_SetSRID(ST_MakePoint(lon * 1e-7, lat * 1e-7), 4326),
    ST_GeomFromText(%s, 4326)
);
```

Avec %s remplacé par le polynôme d'isochrones généré par graphHopper (cf. `utils/utils.py/create_graph_from_postgreSQL()`)

En effet, voici le résultat :