Web-SIG permettant de visualiser la comparaison des mobilités en période COVID et hors période COVID à Toulouse et à Lyon

Réalisé par :

- KHLIFI Ali
- FAUCHER Paul
- HADDOUCHE Chihab

Introduction

Dans le cadre de l'UE « Atelier Géomatique », notre équipe constituée de trois étudiants du master 2 SIGMA, a pour objectif d'imaginer et de concevoir un outil d'exploration multidimensionnel de données d'utilisation de vélos en libre-service (VLS). Cette requête concerne l'utilisation de cette alternative à d'autres moyens de transport, en période de COVID et hors période de COVID à Toulouse et à Lyon. Destiné aux chercheurs, cet outil fait partie d'un projet plus vaste mené par une équipe pluridisciplinaire de chercheures toulousains et lyonnais. Ces derniers proposent de comparer l'effet de la pandémie sur l'utilisation de "VELÔToulouse" à Toulouse et des "velo'v" à Lyon. L'objectif général du projet est d'aider les opérateurs et les pouvoirs publics à se saisir rapidement de ces changements d'habitudes d'usage pour répondre au mieux à la demande. Il est aussi important de Comprendre les opportunités actuelles en termes d'usage de vélos pour mettre en place les toutes les conditions nécessaires à une pérennisation des pratiques cyclables.
Ce travail fait suite à une étude de faisabilité, menée par des étudiants du master 1 SIGMA dont l'objectif était de suggérer des solutions techniques pour la création d'un Web SIG vélos. Il s'en est suivi la production d'un état de l'art commenté des visualisations utiles pour ce type d'analyse, ainsi qu'une maquette d'interface et des préconisations pour la gestion des données. La notion d'outil multidimensionnelle est mise en évidence. L'outil doit permettre de représentation spatiale de la donnée, en prenant compte de la dimension temporelle, mais pas que. Une représentation de données d'autres moyens de transport, tel que les trafics routiers et les transports en commun sont aussi demandés à titre comparatif. L'objectif étant d'obtenir l'analyse la plus détaillée des phénomènes observés, ainsi que de trouvé les explications les plus plausibles en intégrant le plus de variables explicatives possible, tels que la météo.

Un travail préalable de traitement et de transformation chronophage d'une donnée « brute » à une donnée exploitable par l'application est donc à faire. Cela dit nos accomplissements n'entrent que dans le cadre d'une base, qui servira à un stage de six mois par la suite.
Nos missions pour ce projet sont donc multiples et vont de la conception d'un modèle conceptuel de données adapté aux usages prévus, à la construction d'une architecture détaillée et un prototype d'application web, ainsi que l'organisation d'un système d'information (stockage, traitement, visualisation).

Organisation du travail

Lorsque le projet nous a été confié, nous avons tout d'abord cherché à comprendre l'entièreté du sujet, ainsi que les attendus des chercheurs à notre égard. Suite à la réunion de démarrage du 24/01/2022, nous avons articulé les missions du projet ainsi que l'objectif attendu. Pour ce faire, nous avons choisi d'organiser notre travail, d'élaborer un planning et de nous y tenir le plus possible.

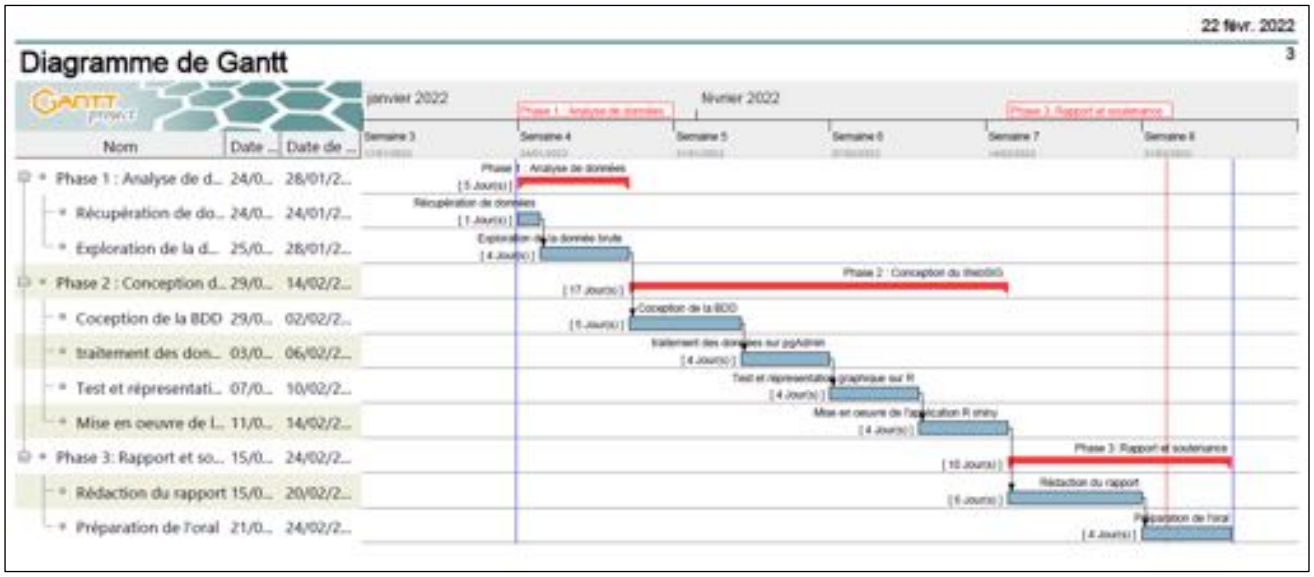


Figure 1 : Diagramme de Gantt pour l'organisation du travail

Nous avons organisé notre travail en trois phases majeures. La première consistait à l'analyse et l'exploration de données brutes mises à disposition par les chercheurs. La seconde visait la conception et le développement de l'outil WebSIG et enfin la troisième phase était consacrée à la rédaction et la préparation des livrables de notre travail.
Nous nous sommes répartis le travail sur trois et chacun s'occupe d'une partie à savoir mobilité vélo, transport en commun et trafic routier, tout en échangeant et en se complétant au besoin. Nous avons gardé cette méthodologie de partage des tâches tout le long des trois phases du travail, avec des réunions en Visioconférence quasi quotidiennes pour voir l'avancement globale et résoudre d'éventuels blocages techniques.

Présentation et exploration des données brutes

Contenu des données de test à notre disposition

Vélo :

Ces données VSL sont des données protégées fournies par JCDecaux.
Ces données brutes nous ont été transmises sous la fome de fichier csv (séparé par point-virgule) de la forme :

```
"Code borne sortie";"Borne sortie";"Date sortie CNIL (sans secondes)";"Code borne retour";"retour";"Date retour CNIL (sans secondes)"
```

6005;"6005 - PLACE EDGAR QUINET";"2019/01/01 00:00:00";2024;"2024 - R…PUBLIQUE / MAUPIN";"2019/01/01 00:20:00"

Cette structure de la donnée brute est la même pour les deux villes. On met en évidence que chaque enregistrement correspond à un flux avec le lieu et la date de sortie et de retour pour un même vélo. La précision de ces enregistrements est à la minute. On a aussi accès avec cette donnée à la relation entre le code de la borne et le nom de la borne, mais nous n'avons pas directement accès à la localisation géographique de ces bornes. Cette information de localisation des bornes est cependant facilement accessible en open source sur les sites data.Toulouse-métropole pour Toulouse et data.grandlyon pour Lyon. On remarque que les identifiants des bornes présents dans les données open sources correspondent exactement à ceux présents dans les enregistrements des trajets VLS fournis par JC Decaux.

Après une étude plus approfondie des données, nous avons remarqué que les codes des bornes de Lyon vont de 201 à 34002 et ceux de Toulouse de 0 à 288. Nous avons ensuite essayé de déterminer si des bornes de Toulouse et de Lyon avait le même 'code borne', ce qui pourrait poser un problème lors d'un éventuel regroupement. C'est le cas uniquement pour le code 201. Ce code correspond toutefois à une station de test dans les données de Lyon. Cela nous a permis de mettre en évidence que certaines stations n'étaient en réalité pas de vraies stations, mais des stations de tests. Ces stations seront donc à supprimer lors de notre traitement de la donnée brute.

Trafic :

Les données de trafic comprennent deux types de fichiers et sont fournies par Autoroutes Trafic. Nous avons eu accès à des fichiers csv contenant les données du trafic routier par tronçon de route qui se présente de la forme suivante :

"deviceID",	"latitude" ,"longitude","speed","heading", "horodate", "LINK_ID"
f9f29a305ca1a2d9a538e0570a7f92b1c2f1abf5,45.78581161,4.872719128,74,-108,2019-01-01T00:00:31,810795386	

La première colonne "deviceID" contient un identifiant unique et anonymisé du véhicule. La deuxième et la troisième colonne correspondent respectivement à la latitude et à la longitude du véhicule dans le système de coordonnées WGS84. La quatrième colonne contient l'information de vitesse du véhicule en km/h. La cinquième correspond au cap du véhicule (angle d'orientation en degré : Nord = 0 et croit dans le sens anti-trigonométrique). La sixième colonne contient l'horodate de la position GPS au format {yyyy-MM-dd"T"HH:mm:ss}. Chaque enregistrement est donc précis à la seconde. Cela donnera lieu à une potentielle agrégation temporelle moins précise et générant ainsi des questions quant aux méthodes D'agrégation des informations de latitude, longitude, vitesse et cap. Finalement, la septienne colonne contient l'identifiant du tronçon routier sur lequel la position GPS a été map-matchée. L'autre source de données correspond à des shapefiles contenant les tronçons de route également identifiés par l'attribut "LINK_ID" pour avoir les informations géographiques associées aux données de trafic. Ces fichiers contiennent aussi entre-autres le nom des tronçons. Les autres champs présents ne sont pas des champs qui seront intéressants à conserver. Nous

avons remarqué que les tronçons présents dans ces shapefiles sont de taille assez faible, cela diminue donc la probabilité d'avoir plusieurs fois le même id_gps sur le même tronçon. C'est une information importante qu'il faudra avoir en tête si nous décidons de compter le nombre de signaux GPS par tronçon pour calculer une densité.

Données transport en commun :

La structure des données brutes est différente pour les deux villes. Elle sera donc la présenter séparément :

Données transport en commun Toulouse :

Ces données de transport en commun sont des données protégées fournies par TISSEO. Nous avons reçu deux types de fichiers brutes. Un premier qui correspond aux données billettiques (nombre de tickets validés) de transport en commun (métro, bus et tram) par station d'entrée et toutes les 15 minutes. Ces fichiers au format csv contiennent les informations formatées de la façon suivantes :

"DATE_VALID";"HEURE_VALID";"NUM_LIEU2";"NBVALID";"NBDEPL"
2019-01-01 00:00:00.000;"0400";100023;24;24

Pour la suite de notre travail, le champ NBDEPL qui correspond au nombre de déplacements n'est pas un champ qui sera à conserver, car nous ne nous interresons qu'au nombre de validations par station (NBVALID), c'est-à-dire l'utilisation en elle-même du transport.
Dans ce fichier les stations sont désignées par un numéro dans le champ NUM_LIEU2. C'est d'ailleurs une information qu'on retrouve dans l'autre type de données brutes reçu, à savoir un fichier qui associe à ce numéro de lieu, le nom du lieu et les coordonnées de ce lieu. Nous avons identifié que ces coordonnées étaient en "Lambert 93 II étendu". Ce fichier "lieu" contient les informations formatées de la façon suivante :

NUM_LIEU;NOM_LIEU;X;Y
10;Les Abattoirs;526578,1;1844787,5

Après une première exploration de ces fichiers, nous nous sommes rendus compte que les champs NUM_LIEU des deux fichiers ne correspondaient pas, rendant alors impossible toute tentative de jointure entre ces données. Après sollicitation de TISSEO, nous avons finalement réussi à obtenir un fichier lieu avec les numéros de lieu qui correspondaient à ceux présents dans les fichiers contenant les données de validations. Nous avons également remarqué dans ce fichier "lieu" que pour un même lieu (NOM_LIEU), il pouvait exister une station de métro et une station de tram ou bus avec bien sûr des NUM_LIEU différents. De plus, nous avons remarqué qu'il pouvait y avoir plusieurs NUM_LIEU pour une même station de métro ou de

bus/tram. TISSEO nous a expliqué que cela est lié au fait qu'un NUM_LIEU est associé à une borne de validation et qu'il peut exister plusieurs bornes de validation pour un même arrêt ou une même station. TISSEO nous a aussi informés que les lieux avec un NUM_LIEU supérieur à 100000 correspondent à la station de métro, le reste étant des stations de bus ou de tram.

Données transport en commun Lyon :

Ces données de transport en commun sont des données protégées fournies par SYTRAL.
Pour les données de transport en commun de Lyon, nous n'avons accès qu'aux données de métro dont l'information est formatée de la façon suivante :

Date jour (CAS tr)	Mois	Année	Code ligne	Station	Jour	Tranche horaire	Heure (CAS tr)	Nb entrées total (CAS tr)	Nb sorties total (CAS tr)
01/01/2019	1	2019	A	Ampère Victor Hugo	2	(vide)	00:00:00	2	4

Comme pour les données de transport en commun toulousain, nous avons l'information du nombre d'entrées (qui est équivalent au nombre de validations) toutes les 15 minutes pour chaque station. Nous disposons en plus de l'information du nombre de sorties, mais étant donné que cette information n'est pas disponible dans les données de Toulouse ne n'allons pas la conserver dans la suite de notre travail. On remarque que le champ jour est en contradiction avec l'information contenue dans le champ 'Date jour (CAS tr)'. À la différence des données brutes disponibles pour les transports toulousains, nous ne disposons pas ici d'identifiant unique pour chaque station et nous n'avons pas non plus l'information de la localisation de ces stations. On obtient cette information en téléchargeant le shapefile des stations de métro sur data.grandlyon.
Lors de l'exploration de ces données nous nous sommes rendus compte que l'agrégation des enregistrements de même date et même station n'avait pas toujours été faite dans le fichier brut. De plus, pour certains enregistrements on trouve la présence de lettres (A, B, C ou D) dans le champ nb_valid.

Données météo :

Les données météorologiques à notre disposition sont fournies par Météo-France. Le fichier brut reçu comprend de nombreuses métriques mesurées toutes les heures de 2019 et 2020 pour deux sites pour chaque ville (TOULOUSE—BLAGNAC, TOULOUSE—FRANCAZAL, LYON—BRON et LYON—ST EXUPERY). Nous avons aussi eu l'information de la localisation de ces différents postes de mesure. Voici un extrait du fichier Excel regroupant ces données :

No jour	No semaine	JOUR	POSTE	DATE	RR1	QRR1
1	1	01/01/2019	31069001	01/01/2019 : 00	0	v
1	1	01/01/2019	31069001	01/01/2019 : 01	0	v
1	1	01/01/2019	31069001	01/01/2019 : 02	0	v
1	1	01/01/2019	31069001	01/01/2019 : 03	0	v
1	1	01/01/2019	31069001	01/01/2019 : 04	0	v

À noter que les valeurs contenues dans le champ 'No jour' vont de 1 à 366 et ne correspondent pas à numéro du jour de la semaine.

Cette exploration initiale de la donnée brute nous a permis de voir les possibilités d'homogénéisation et de potentiel regroupement de la donnée, de voir les possibilités de suppression de champs inutiles pour la suite et également de mettre en évidence des problèmes dans les données brutes. Cette étape a été essentielle pour prendre du recul sur toute la donnée à notre disposition pour pouvoir penser et commencer à concevoir une structure de base de données permettant de stocker l'ensemble de cette information au mieux.

Proposition de structure de la BDD et structure des tables de cette dernière :

Principes de base

En prenant compte la complexité des tables brutes fournies ainsi qu'une perspective de calculs statistiques futurs pas forcément définie dès le début, la création d'une base de données évolutive était préconisée. L'objectif était de structurer une base de données relationnelle afin d'aller chercher n'importe quelle information stockée dans les différentes tables appelée (relations).

Chaque relation possède une clé primaire qui la définie. Cette clé correspond à des champs parmi ceux de la table en question, qui est défini comme étant un identifiant unique. L'appellation clé a été attribuée, car c'est cet identifiant qui permet d'accéder à l'information grâce à des jointures. Le langage SQL permet de faire les requêtes nécessaires afin que toute la relation mise en service soient préalablement jointes et permet également de demander l'information cherchée. Grâce à cette modélisation, il est aussi possible de parcourir l'ensemble de la base de données et d'obtenir de nouvelles tables, contenant des informations relatives a des champs se retrouvant dans des tables différentes.

Deux importants paramètres sont à prendre en considération lors de la construction de la base de données :

1. Éviter la redondance :

Dans une table, en particulier en présence de plusieurs champs, certaines informations sont susceptibles de se répéter. En effet, en fonction des attributs avec lesquels elles sont en relation, des lignes de champs se répètent en fonction des différents cas de figures. C'est ce qui a été remarqué par exemple pour l'attribut "lieux" ou les coordonnées géographiques sont répétés à chaque fois que le lieu est compris dans un trajet en fonction du temps. Dans ce cas, l'information est sortie vers une table apparentiez en l'absence de l'attribut qui induit sa répétition.

Cette approche permet une conception dite évolutive de la base de données. En effet, les changements avenir concernant l'attribut dont la redondance a été évitée, se fera plus facilement et en évitant toute erreur d'oubli ou d'inattention. De plus, l'ajout d'entités a la table se fera plus rapidement et de façon automatique.

1. La notion de clé :

Les clés représentent la base du fonctionnement de la base de données relationnelle. Pour qu'une jointure soit possible entre deux relations, une clé primaire et une clé étrangère doivent impérativement correspondre. Le choix de ces deux clés n'est pas évident, mais doit être

préalablement réfléchie. Dans notre cas, les lieux semblaient être le centre de toutes les informations disponibles. On retrouvait cette information au niveau de toutes les relations, elle pouvait donc représenter des clés primaires et étrangères.

Conception de la base de données

1. Mise en forme des tables :

Un choix de 7 tables différentes a été retenu pour la base de données. Les deux tables “ville” et “lieu” sont au centre des relations, avec les deux champs ‘id_lieu’ et ‘id_ville’, comme clé primaire et étrangère pour la plupart des tables. Cette conception permet de se déplacer perpendiculairement sur la base de données et l’utilisation de ces identifiants donne accès aux informations contenues dans toutes les tables.

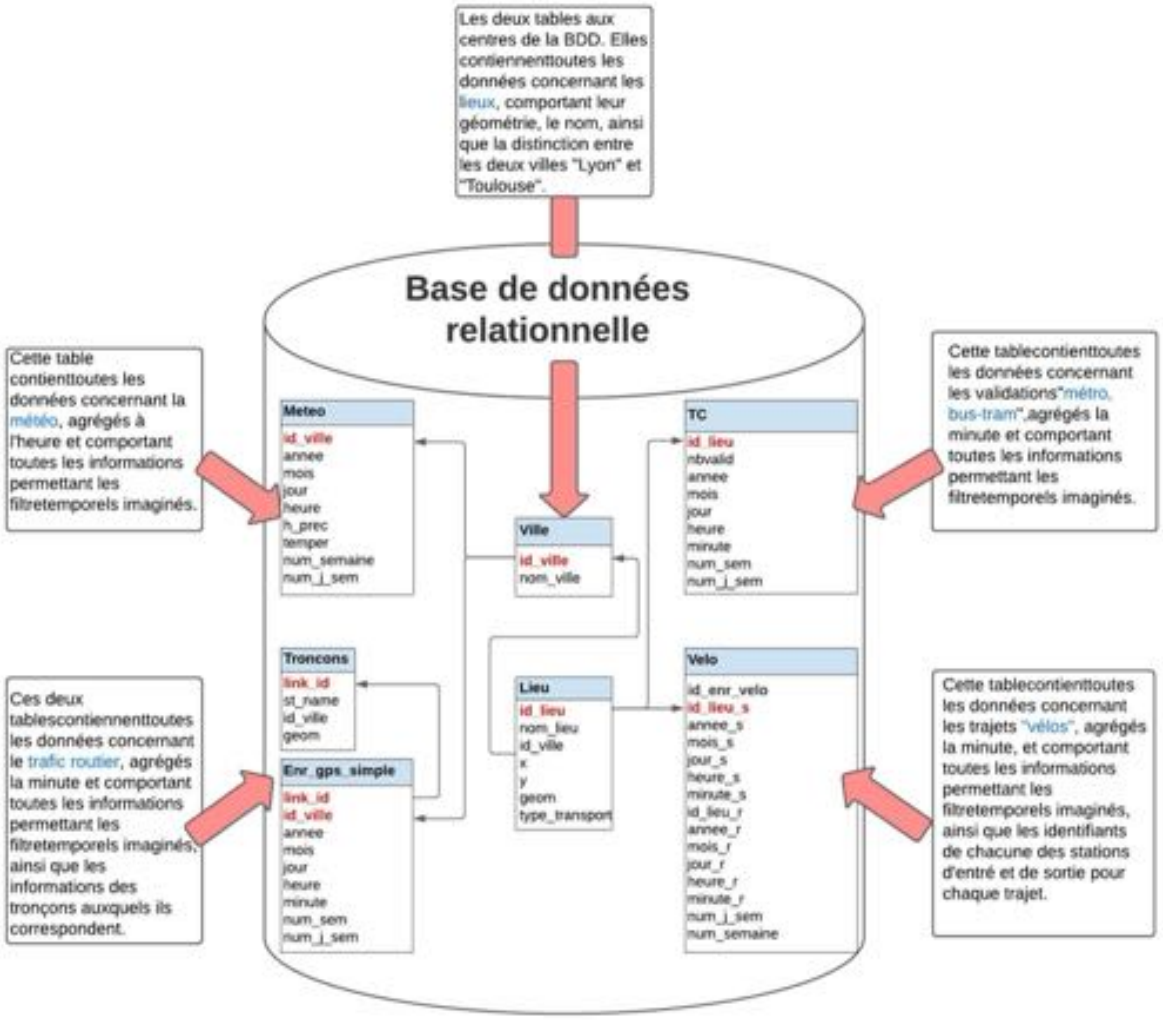


Figure 2: Shéma de la base de données

Pour les trois tables “Transport en commun", “trajet vélo” et “Trafic routier”, le choix a été pris de les garder séparées. Malgré le fait qu’elles contiennent des informations de transport, le format de la donnée reste différent. Les informations de lieux auxquels ils font référence restent tout de même agrégés dans la table “lieu” et permettent de transposer leurs informations si besoin dans la même relation.
Le champ “link id” des données de trafic offre l’opportunité de faciliter l’agrégation spatiale des points, il permet en effet de calculer la densité par tronçon. Les champs de la longitude et de la latitude du véhicule, la vitesse, et le cap ont été supprimés, car ils empêchent l'agrégation des données et la simplification de la table “trafic routier”
Pour les tale “lieu”, ”ville” au centre de la base, la création de l’id lieux a été préconisé. En effet, en créant cet id unique à chaque lieu se trouvant dans les différentes tables, il était possible d’en faire une clé primaire et étrangère facilitant l'accès à toutes les autres relations.

Cependant, lors de l’exploration des données brutes, nous avons remarqué qu’il pouvait avoir un identifiant de lieu qui fait référence à une station de vélo et une station de transport en commun toulousain. Dans le but d'associer à chaque lieu un identifiant unique il a fallu définir des règles pour construire cet identifiant à partir des identifiants déjà présents dans la donnée brute. Nous avons décidé de conserver les identifiants d’origine des transports en commun de Toulouse (qui vont de 10 à 100081). Comme dit précédemment nous n’avons pas détecté de problème de recouvrement entre les identifiants de stations de vélo toulousains et lyonnais si l’on supprime les stations de test. Mais pour éviter les recouvrements avec les identifiants des transports en commun toulousains, nous avons décidé d’ajouter le caractère “v” aux identifiants présent dans la donnée brute des stations de vélo. Pour ce qui est des identifiants des stations de métro lyonnais, étant donné qu’il n’y en avait pas dans la donnée brute nous avons décidé de les indexer à partir de 200000 pour éviter tout recouvrement avec les identifiants des transports en commun toulousains.

Traitements et formatage de la donnée brute :

Dans cette partie, nous allons décrire en détail toutes les étapes de la chaîne de traitement qui permet de passer de la structure de la donnée brute à la structure de chaque table de notre BDD comme imaginé lors de la conception de cette dernière. Ces chaînes de traitement s’achèvent par l’import de ces tables sur la base de données.

La table ‘Ville’ :

La table ville est la table la plus simple de notre base de données car les données que nous avons à notre disposition proviennent uniquement de deux villes à savoir Lyon et Toulouse. On attribut de façon arbitraire un identifiant pour chaque ville. Cette table à été construite à la main étant donné sa simplicité. Voici le résultat obtenu par cette table :

	id_ville [PK] integer	nom_ville character varying (50)
1	1	Lyon
2	2	Toulouse

La table ‘Lieu’ :

Comme expliqué précédemment cette table doit pouvoir contenir les informations de toutes les stations et bornes pour chaque type de mobilité à savoir VLS, métro et bus/tram (autre).
Ces informations sont les suivantes : l’identifiant unique du lieu (id_lieu), le nom de l’endroit où se trouve ce “lieu” (nom_lieu), l’identifiant de la ville où se trouve ce lieu (id_ville), les coordonnées géographiques du ce “lieu” (x pour la longitude, y pour la latitude et un champ ‘geom’ contenant l’information géométrique ponctuelle de ce “lieu”) et enfin le type de transport (type_transport) de ce “lieu” pour savoir s’il correspond à une station de bus/tram, de métro ou bien si c’est une borne de VLS.
La première étape a été de créer des ‘sous-tables lieu’ pour chaque type de mobilité à partir des fichiers de données brutes ou de données disponibles en opendata.
Commençons par détailler comment nous avons obtenu la table lieu pour les bornes de VLS des deux villes (table ‘*lieu_velo’*).

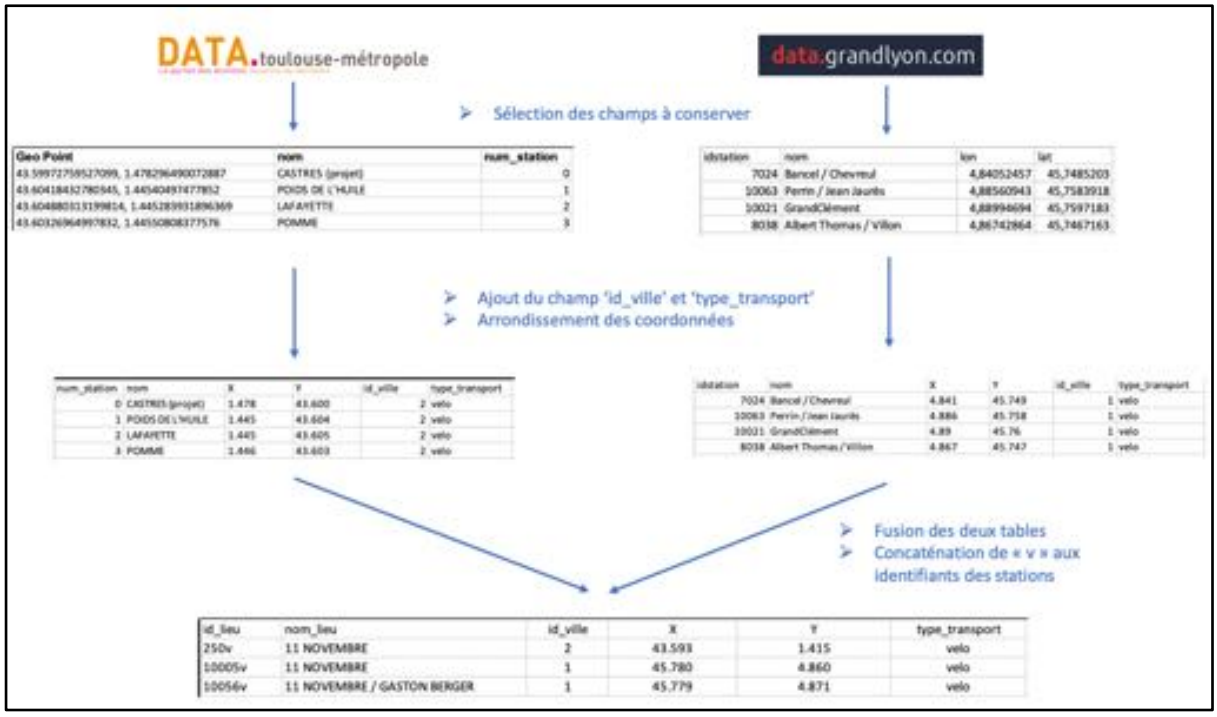


Figure 3: Chaîne de traitements pour obtenir la table ‘lieu_velo’.

La table ‘*lieu_velo*’ a été obtenue à partir de données open source des villes de Toulouse et Lyon. Dans un premier temps, nous avons gardé uniquement les champs d’intérêts, à savoir les identifiants des stations, le nom du lieu et les coordonnées géographiques dans le système La table ‘*lieu_velo*’ a été obtenue à partir de données open source des villes de Toulouse et Lyon. Dans un premier temps, nous avons gardé uniquement les champs d’intérêts, à savoir les identifiants des stations, le nom du lieu et les coordonnées géographiques dans le système.

WMS84. Nous avons ensuite arrondi (fonction ARRONDI() de Excel) et séparé les coordonnées pour obtenir les champs X et Y (sélection de la colonne > Données > Convertir > choix du séparateur). L'ajout des champs 'id_ville' et 'type_transport' a été fait séparément sur les deux tables de façon à pourvoir facilement différencier les bornes situées à Toulouse de celles situées à Lyon. Finalement, nous avons fusionné ces deux tables et ajouter le caractère "v" (fonction CONCAT() de Excel) à l'identifiant des bornes déjà existant pour respecter les règles de définition des 'id_lieu' établies précédemment. Notons que les identifiants des bornes présents dans les données open source correspondent exactement à ceux présents dans les enregistrements des trajets VLS fournis par JC Decaux.

Les processus de construction de la table lieu pour les transports en commun s'est faite différemment pour les deux villes étant donné les différences de mise en forme des données brutes reçues pour les deux villes.
Dans le cas de Toulouse, nous disposions déjà d'un fichier *ExportCNRS-Lieux.csv* dans lequel chaque station de transport en commun avait un numéro de lieu, un nom de lieu et les coordonnées de ce lieu. Un premier traitement de formatage de ce fichier est réalisé sur Excel pour avoir des noms de lieux sans "caractères bizarres" (ouverture du fichier en indiquant comme origine du fichier : Windows (ANSI)) et pour avoir des points au lieu de virgule dans les coordonnées X et Y (rechercher/remplacer). On obtient le fichier *lieux_point.xls* :

NUM_LIEU	NOM_LIEU	X	Y
10	Les Abattoirs	526578.1	1844787.5
11	Les Abattoirs	526581.2	1844756.6
20	Achiary	530268.1	1844710
21	Achiary	530272.7	1844719.7
30	Aérodrome	530070.2	1841882.6

À noter que les coordonnées sont encore en Lambert II étendu.
Nous avons ensuite formaté ce tableau pour qu'il présente les mêmes champs que l'on doit retrouver dans la table 'lieu' de notre BDD. On ajoute alors les champs 'id_ville' et 'type_transport'. D'après l'analyse du contenu du fichier faite précédemment, on considère que les lieux avec un 'NUM_LIEU' supérieur à 100000 correspondent à des stations de métro et le reste est considéré comme autre (bus ou tram). On utilise alors la fonction de Excel SI() pour remplir le champ type_transport en fonction des valeurs contenues dans le champ 'NUM_LIEU'.
On obtient ainsi la table 'lieu' des tc à Toulouse sans agrégation *lieu_TC_toulouse_NA.xlsx* :

NUM_LIEU	NOM_LIEU	id_ville	X	Y	type_transport
10	Les Abattoirs	2	526578.1	1844787.5	autre
11	Les Abattoirs	2	526581.2	1844756.6	autre
20	Achiary	2	530268.1	1844710	autre

La table obtenue précédemment est dite non agrégée, car nous avons remarqué que pour un même 'NOM_LIEU' et un même type de transport, il est possible d'avoir plusieurs

'NUM_LIEU'. Nous avons donc décidé de réaliser un traitement permettant de n'avoir qu'un seul 'NUM_LIEU' pour un même 'NOM_LIEU' et 'type_transport'. Il faut ainsi prendre en considération le champ 'type_transport' en plus de celui du 'NOM_LIEU' pour faire cette agrégation de façon à ne pas agréger des "lieux de métro" avec des "lieux d'autre transport en commun".
Pour réaliser ce type de traitement, nous importons le fichier *lieu_TC_toulouse_NA.xlsx* précedemment obtenu sur QGIS (sans cocher le champ géométrie) pour pouvoir exécuter des requêtes SQL sur cette table via la fonctionalité de gestion de base de donnée de QGIS. Pour pouvoir agréger les enregistrements de cette table par le champ 'NUM_LIEU' pour toutes les stations de métro, on lance la requête ci-dessous :

```
select *
from lieu_TC_toulouse_NA
where type_transport='metro'
group by NOM_LIEU
```

On obtient en exportant le résultat de cette requête lieu_metro_toulouse_A.xls :

id_lieu	nom_lieu	id_ville	X	Y	type_transport
100010	Argoulets	2	530511.8	1847325.3	metro
100001	Arènes	2	525764	1843973	metro
100025	Bagatelle	2	525204	1842413.5	metro
100012	Balma-Gramont	2	530961	1847796	metro
100045	Barrière de Paris	2	527048.7	1847555.6	metro
100014	Basso Cambo	2	523603.7	1841362.3	metro

Pour faire le même traitement mais pour les autres types de transport en commun, on lance la requête suivante pour obtenir *lieu_tc_toulouse_A.xls.*

```
select *
from lieu_TC_toulouse_NA
where type_transport='autre'
group by NOM_LIEU
```

id_lieu	nom_lieu	id_ville	X	Y	type_transport
16270	14ème Rgt d'Infanterie	2	518745.9	1846297.9	autre
20840	1945/05/08	2	530806.7	1838006.7	autre
28230	1962/03/19	2	523116	1851553	autre
870	1er RTP	2	522995.6	1838572.1	autre

Les noms de lieu correspondant à une date ont été modifié à la main.
On fusionne ensuite ces deux tables de lieux via Excel : *lieu_metro_toulouse_A.xls* + *lieu_tc_toulouse_A.xls.*
On obtient le fichier *lieu_mtc_toulouse.xls*.
Finalement, on convertit les coordonnées X et Y dans le système de coordonnées wgs 84. Pour ce faire, on importe le fichier précédemment obtenu sur QGIS en indiquant cette fois-ci que les

champs X et Y sont des champs de géométrie pour que le fichier soit considéré comme une couche. On utilise en ensuite la fonction QGIS 'Reprojeter une couche' pour changer le système de coordonnées de la couche. Puis pour récupérer les champs X et Y convertis dans la table attributaire on utilise la 'calculatrice d'un champ' pour créer les champs 'X' et 'Y' avec les instructions `x($geometry)` et `y($geometry)`.

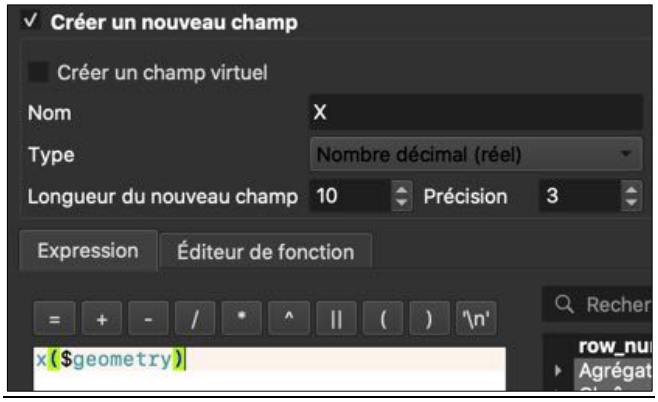


Figure 4 : Paramétrage de la calculatrice d'un champ lors de la création du champ 'X'

On obtient alors la table 'lieu' pour les transports en commun toulousains avec les coordonnées X et Y en wgs 84 (*lieu_mtc_toulouse_wsg.xls)*.

id_lieu	nom_lieu	id_ville	type_transport	X	Y
16270	14ème Rgt d'Infanterie	2	autre	1.331	43.614
20840	08-mai-45	2	autre	1.482	43.541
28230	19-mars-62	2	autre	1.385	43.662

Il ne manque alors plus que la table 'lieu' pour les métros de Lyon pour obtenir la table 'lieu' finale. Pour les métros lyonnais, nous ne disposons pas de l'information de la localisation des stations dans les fichiers bruts. On obtient cette information en téléchargeant le shapefile des stations de métro sur data.grandlyon.
Ensuite, la procédure est semblable à celle effectuée précédemment pour les transports en commun toulousains. On utilise la fonction QGIS 'Reprojeter une couche' pour changer le système de coordonnées de ce shapefile en wgs84. Puis pour récupérer les champs X et Y convertis dans la table attributaire on utilise la 'calculatrice d'un champ' pour créer les champs 'X' et 'Y' avec les instructions `x($geometry)` et `y($geometry)`. À l'aide de l'outil 'gestion de base de données' de QGIS en lance une requête permettant de ne garder que les champs correspondant au nom du lieu et aux coordonnées X et Y et également de ne garder qu'une seule ligne par nom de station en regroupant par le nom du lieu.

```
select nom, X,Y
from metro_reproject
group by nom
```

On obtient la table suivante :

	nom	X	Y
1	Ampère Victor Hugo	4.829	45.753
2	Bellecour	4.834	45.756
3	Brotteaux	4.859	45.767

On exporte ensuite cette table sur Excel pour ajouter les champs 'id_lieu', 'id_ville' et 'type_transport'. Comme prévu précédemment pour éviter les recouvrements au niveau des identifiants de lieu avec les lieux de transport en commun Toulousains ont indèxe les 'id_lieu' de ces stations à partir de 200000.
On obtient alors la table *lieu_metro_lyon* formatée comme la table 'lieu' imaginée lors de la conception de notre base de données :

id_lieu	nom_lieu	id_ville	X	Y	type_transport
200000	Ampère Victor Hugo	1	4.829	45.753	metro
200001	Bellecour	1	4.834	45.756	metro
200002	Brotteaux	1	4.859	45.767	metro

Finalement, les trois tables 'lieu' obtenues (tc_toulouse, metro_lyon et velo) sont fusionnées dans la même table via Excel. Pour mettre tous les noms de lieu du champ 'nom_lieu' sous la même forme, on applique à cette colonne la fonction NOMPROPRE () de Excel. On obtient alors la table lieu comme imaginée lors de la conception de la structure de la base de données (*lieu_final.xlsx*) :

id_lieu	nom_lieu	id_ville	X	Y	type_transport
3670	Jean Jaurès	2	1.449	43.605	autre
100033	Jean Jaurès	2	1.449	43.605	metro
7009v	Jean Jaurès	1	4.838	45.738	velo
7008v	Jean Jaurès / Guillotière	1	4.846	45.753	velo
10063v	Perrin / Jean Jaurès	1	4.886	45.758	velo
200035	Place Jean Jaurès	1	4.838	45.738	metro
18v	Strasbourg Jean Jaures	2	1.448	43.606	velo

Ces données sont ensuite importées sur notre base de données PostgreSQL via PgAdmin. Il faut d'abord créer la table vide sur la base en indiquant le nom des champs et le type de données qu'ils vont contenir avec la requête SQL suivante :

```
create table lieu (id_lieu varchar(50),nom_lieu
varchar(100),id_ville int,X decimal, Y decimal
,type_transport varchar(50),primary key (id_lieu))
```

On peut ensuite importer le fichier *lieu_final.csv* dans cette table.

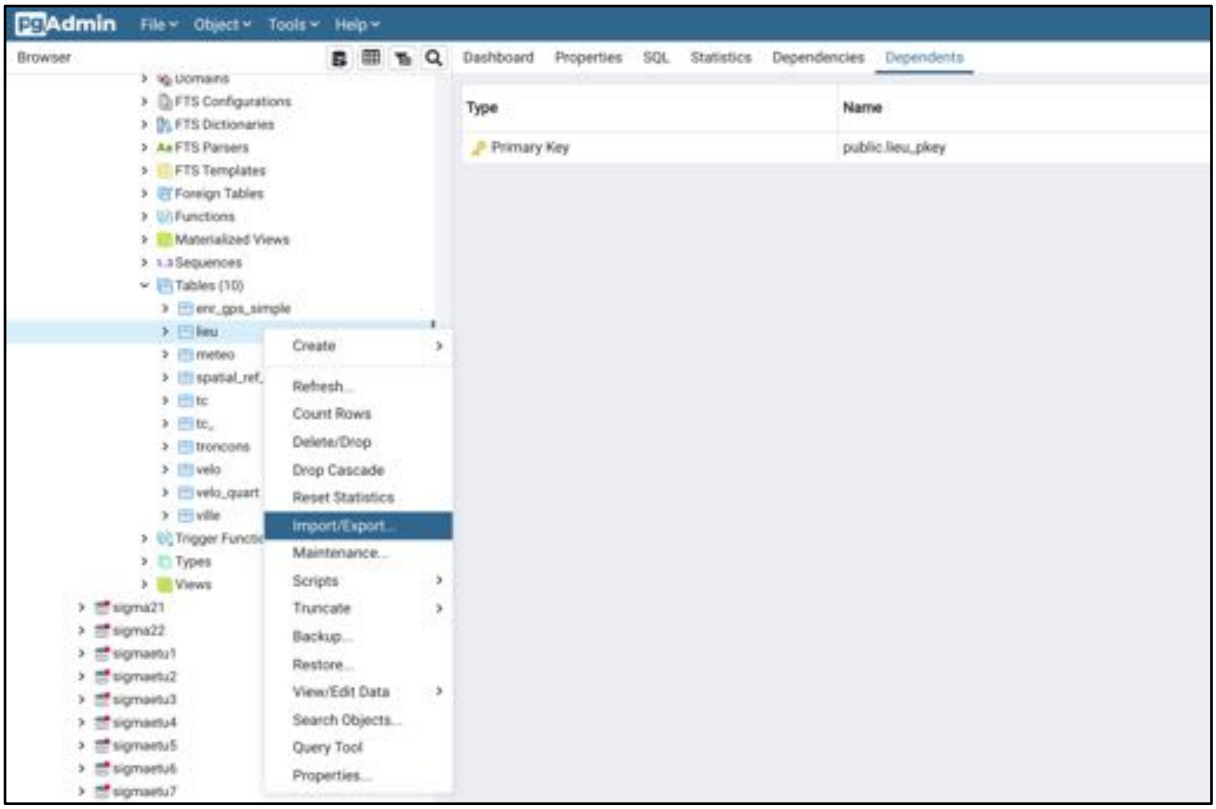


Figure 5 : Capture d'écran pour montrer comment importer des données sur une table d'une base de données via PgAdmin.

Le dernier traitement réalisé sur cette table est l'ajout du champ 'geom' contenant l'information géométrique ponctuelle des "lieux" calculée à partir des champs 'x' et 'y' grâce à la requête suivante :

```
ALTER TABLE lieu ADD COLUMN geom geometry(Point, 4326);
UPDATE lieu SET geom = ST_SetSRID(ST_MakePoint(x, y), 4326);
```

On obtient alors la table 'lieu' :

	id_lieu [PK] character varying (50)	nom_lieu character varying (100)	id_ville integer	x numeric	y numeric	type_transport character varying (50)	geom geometry
1	20840	08-mai-45	2	1.482	43.541	autre	0101000020E6100...
2	28230	19-mars-62	2	1.385	43.662	autre	0101000020E6100...
3	17140	11-nov-18	2	1.229	43.536	autre	0101000020E6100...
4	250v	11 Novembre	2	1.415	43.593	velo	0101000020E6100...

La table 'Météo' :

Comme expliqué précédemment les données brutes de météo contiennent de nombreuses métriques mesurées à deux postes pour chacune des deux villes. Nous avons convenu avec les chercheurs qu'il était pertinent de ne garder que les informations horaires sur la température et la hauteur des précipitations mesurées au niveau du poste situé le plus proche du centre-ville. Nous avons tout d'abord déterminé quel poste était situé le plus proche du centre-ville pour les deux villes. À partir des coordonnées des postes fournis nous avons représenté sur une carte leur position avec le logiciel QGIS. On peut voir les résultats ci-dessous :

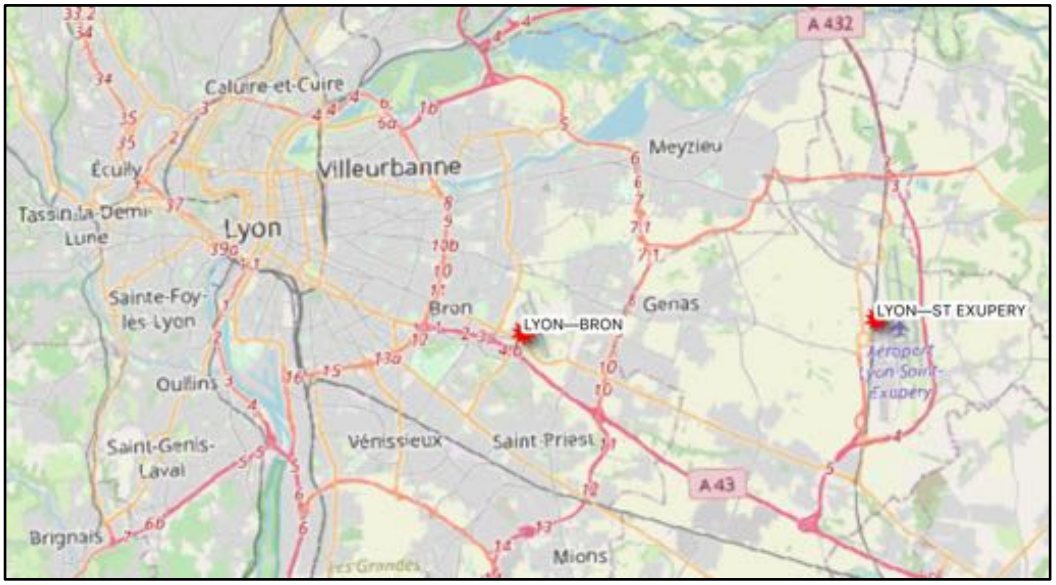


Figure 6 : Localisation des postes de mesures météorologiques de Lyon

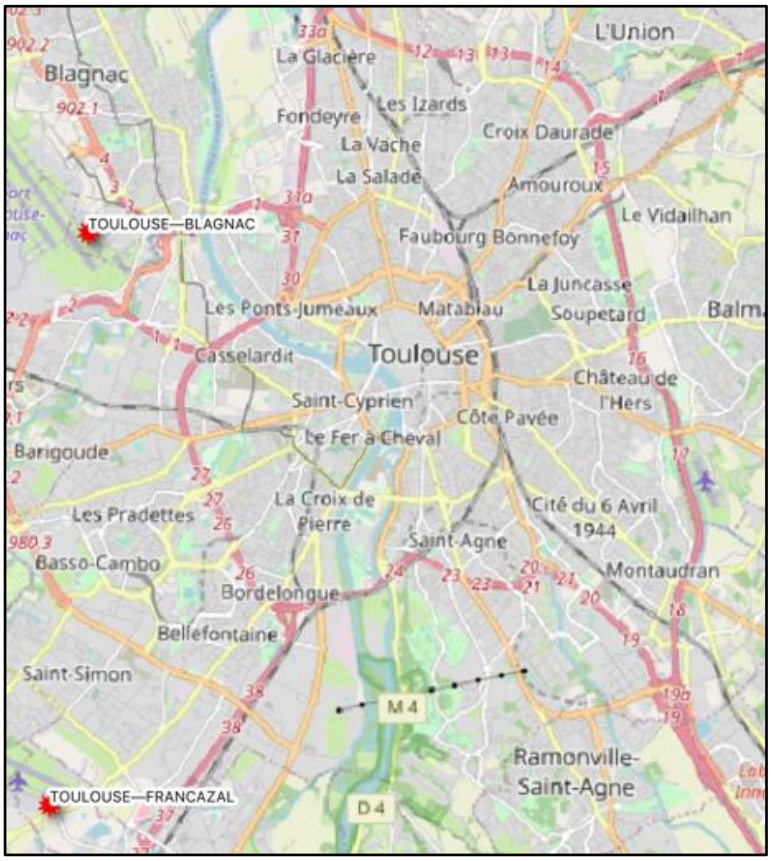


Figure 7 : Localisation des postes de mesures météorologiques de Toulouse.

D'après ces résultats nous allons garder les mesures du poste Lyon-Bron pour Lyon et Toulouse-Blagnac pour Toulouse.

On passe ensuite à la chaîne de traitement pour obtenir la table meteo imaginée lors de la conception de la structure de la base de données. À partir du fichier brut des enregistrements

météo fournis par les chercheurs, on repart des enregistrements horaires sur lesquels on garde uniquement les champs à conserver, c'est-à-dire les champs : 'No jour', 'No semaine', 'POSTE', 'DATE', 'RR1' (hauteur des précipitations horaire) et 'T'(température sous abri horaire).

No jour	No semaine	POSTE	DATE	RR1	T
1	1	31069001	01/01/2019 : 00	0	1,6
1	1	31069001	01/01/2019 : 01	0	1,8
1	1	31069001	01/01/2019 : 02	0	1,7

A partir de ce tableau, nous avons séparé le champ 'DATE' pour obtenir les champs 'heure', 'jour', 'mois' et 'annee'. Nous avons ensuite supprimé les enregistrements des deux postes 'éloignés' du centre-ville (filtrage>supprimer). On remplace alors le champ 'POSTE' par le champ 'id_ville' et on renomme les champs 'RR1' en 'h_prec' et 'T' en 'temp'. On modifie ensuite les valeurs contenues dans le champ 'No jour' pour qu'elles correspondent au numéro du jour de la semaine (allant de 1 à 7) avec les fonctions Excel `TEXTE("date";"jjjj")` qui permet d'extraire le jour de la date choisie suivie de la fonction : `SI.MULTIPLE(P2;"lundi";1;"mardi";2;"mercredi";3;"jeudi";4;"vendredi";5;"samedi";6;"dimanche";7)` pour donner un numéro de jour dans la semaine plutôt que le nom du jour. On fait ensuite en sorte que le séparateur décimal des champs 'h_prec' et 'temp' soit le point plutôt que la virgule (rechercher/remplacer). On obtient ainsi :

no_j_semain	no_semaine	heure	jour	mois	annee	h_prec	temp	id_ville
2	1	0	1	1	2019	0	1.6	2
2	1	1	1	1	2019	0	1.8	2
2	1	2	1	1	2019	0	1.7	2

On importe finalement cette table sur la base de données de la même façon que la table 'lieu'. On obtient alors la table 'météo' sur la base de données :

	id_ville integer	temper numeric (3)	h_prec numeric (3)	annee integer	mois integer	jour integer	heure integer	num_semaine integer	num_j_sem integer
1	2	2	0	2019	1	1	0	1	2
2	2	2	0	2019	1	1	1	1	2
3	2	2	0	2019	1	1	2	1	2

Les tables contenant les informations de trafic ('tronçons' et 'enr_gps_simple') :

Comme prévu dans la structure de la base de données les informations de trafic seront comme pour la donnée brute reçue dans deux tables distinctes.

La table ‘tronçons’ :

Pour cette table la donnée brute se présente sous la forme de deux shapefiles contenant les tronçons routiers respectivement de la ville de Lyon et de Toulouse.

	AR_TAXIS	AR_TRUCKS	LINK_ID	AR_AUTO	DIR_TRAVEL	ST_NAME	PHYS_LANES	SPEED_CAT	AR_BUS	FUNC_CLASS
1	Y	N	58222010	Y	B	CHEMIN DU ...	2	7	Y	5
2	Y	Y	58222028	Y	B	CHEMIN DU ...	2	6	Y	5
3	Y	Y	58222036	Y	B	RUE DE LA B...	2	6	Y	5

Le premier traitement a été de conserver uniquement les champs ‘LINK_ID’, ‘ST_NAME’ de la table attributaire des deux shapefiles d’origine via une requête SQL (selection des champs : ‘link_id’,’ ST_NAME’, ‘geometry’) avec ajout du champ ‘id_ville’ à la table attributaire (avec la valeur 1 pour Lyon et 2 pour Toulouse) via la calculatrice de champ QGIS.
Ces deux shapefiles ont ensuite été fusionnés pour n’en faire qu’un via la fonction QGIS “Fusionner des couches vecteur”.

	LINK_ID	ST_NAME	id_ville	layer	path
1	57717795	D70	2	fusT_sig	/Users/paulfa...
2	57965531	ROUTE DE T...	2	fusT_sig	/Users/paulfa...
3	57965535	D50	2	fusT_sig	/Users/paulfa...

Nous avons ensuite éliminé les champs qui se sont rajoutés après la fusion avec la requête SQL

```
select LINK_ID,ST_NAME, id_ville, geometry
from fusLT
```

On enregistre le résultat de cette requête (*troncon.shp*) dont la table attributaire contient les informations suivantes :

	LINK_ID	ST_NAME	id_ville	geometry
1	57717795	D70	2	MultiLineString ...
2	57965531	ROUTE DE TOULOUSE	2	MultiLineString ...
3	57965535	D50	2	MultiLineString ...
4	57969078	RUE DES ARTS	2	MultiLineString ...
5	57969079	NULL	2	MultiLineString ...

Finalement, nous avons importé ce fichier *troncon.shp* directement sur la base de données via QGIS.
On obtient donc sur la base de données la table ‘troncons’ suivante :

	id [PK] integer	geom geometry	link_id bigint	st_name character varying (254)	id_ville bigint
1	1	0105000020E6100...	57717795	D70	2
2	2	0105000020E6100...	57965531	ROUTE DE TOULOUSE	2
3	3	0105000020E6100...	57965535	D50	2
4	4	0105000020E6100...	57969078	RUE DES ARTS	2
5	5	0105000020E6100...	57969079	[null]	2

La table 'enr_gps_simple' :

Comme expliqué précédemment nous avons fait le choix de travailler sur une table simplifiée des enregistrements de trafic routier, c'est-à-dire sans les champs contenant l'information de la vitesse, du cap, de la longitude et de la latitude du véhicule pour simplifier en première approche les questions que pose l'agrégation à la minute de ces champs. Cependant les transformations à faire à partir des données brutes sont les mêmes que si on avait décidé de garder ces champs.

Les différentes transformations de la donnée brute correspondent :

- à fusionner tous les fichiers en un seul
- à décomposer le champ "date" qui contient les informations sous cette forme : 2019-01-01T00:00:31 en plusieurs champs : année - mois - jour - heure - minute - seconde, ce qui donne avec cet exemple : 2019, 01,01,00,00,31.
- à ajouter le champ 'id_ville' (avec la valeur 1 pour les enregistrements situés à Lyon et la valeur 2 pour les enregistrements situés à Toulouse)

Pour montrer qu'une partie de l'automatisation des traitements de la donnée brute sont possibles nous avons décidé d'automatiser ces premiers traitements via la production d'un script Python (*autom_trafic_test.py* dans le dossier *'livrables'*). Avant d'exécuter ce script : il faut veiller à bien enregistrer ce script au même endroit que les 4 fichiers bruts de la forme : *FCD_trafic_{date}_{ville}_test_S1.txt*.

On obtient en sortie 3 fichiers text dont *fus_all_trafic.txt* qui est le résultat des différents traitements. Ce fichier contient les enregistrements gps non simplifiés avec pour nom de champ dans l'ordre : id_ville, id_gps, lat, long, vitesse, cap, année, mois, jour, heure, minute, seconde.

La prochaine étape est l'agrégation à la minute de ces enregistrements. Et étant donné que nous avons fait le choix de supprimer les champs qui pouvaient faire varier les enregistrements dans la même minute (lat, long, vitesse, cap), il suffit de supprimer le champ des secondes en plus des champs lat, long, vitesse, cap et de regrouper les enregistrements alors identiques. Celà permet aussi de ne compter qu'une seule fois le même véhicule sur un même tronçon dans la même minute. Durant cette étape on supprime aussi le champ 'id_gps' qui n'apporte pas une information utile pour la suite de notre travail.

Cette étape se résume par une requête SQL exécutée sur la table obtenue suite au traitement python. C'est pourquoi nous avons importé ce tableau sur notre base de données de façon à exécuter la requête suivante :

```
select link_id,id_ville,annee,mois,jour,heure,minute
from enr_gps
group by link_id,id_ville,annee,mois,jour,heure,minute
```

On passe alors de la table *enr_gps* :

id_ville character vary	id_gps character varying (5(	lat numeric	long numeric	vitesse integer	cap integer	annee integer	mois integer	jour integer	heure integer	minute integer	seconde integer	link_id integer
2	4ce37c92a577da...	43.64619071	1.464280021	87	102	2019	1	1	0	3	57	1231037557
2	6eabd23fbec0b7...	43.58340808	1.497289411	107	-19	2019	1	1	0	3	58	1231392790
2	a6b4d608ad68ea...	43.61936492	1.393717487	16	-157	2019	1	1	0	4	2	1233975692

à la table *enr_gps_sans_sec* :

	link_id integer	id_ville character varying (50)	annee integer	mois integer	jour integer	heure integer	minute integer
1	1163296415	2	2020	1	1	11	42
2	58226650	1	2019	1	1	14	24
3	58439738	1	2019	1	1	14	18

Il ne reste maintenant plus qu'à ajouter les champs 'num_semaine' et 'num_j_sem'. Ce traitement est réalisé sous Excel. On exporte donc cette table en format csv (*enr_gps_sans_sec.csv)* pour l'ouvrir dans Excel. On ajoute le champ 'num_j_sem' qui contient le numéro du jour de la semaine (allant de 1 à 7) avec les fonctions Excel `TEXTE("date";"jjjj")` qui permet d'extraire le jour de la date choisie suivie de la fonction : `SI.MULTIPLE(P2;"lundi";1;"mardi";2;"mercredi";3;"jeudi";4;"vendredi";5;"samedi";6;"dimanche";7)` pour donner un numéro de jour dans la semaine plutôt que le nom du jour. Pour remplir le champ 'num_semaine' on utilise la fonction `NO.SEMAINE.ISO("date)"` qui permet d'extraire le numéro de la semaine correspondant de la date choisie (avec la date au format "français"). On enregistre ensuite ce fichier au format csv (*enr_gps_simple_s_sem.csv*) pour l'importer sur notre base de données.

On exécute la requête suivante pour créer la table vide :

```
create table enr_gps_simple (
link_id int,
     id_ville int,
     annee int,
     mois int,
     jour int,
     heure int,
     minute int)
```

On importe finalement le fichier csv dans cette table. On obtient alors la table 'enr_gps_simple' :

	link_id integer	id_ville integer	annee integer	mois integer	jour integer	heure integer	minute integer	num_sem integer	num_j_sem integer
1	1163296415	2	2020	1	1	11	42	1	3
2	58226650	1	2019	1	1	14	24	1	2
3	58439738	1	2019	1	1	14	18	1	2
4	62153665	1	2019	1	1	14	25	1	2

La table ‘Velo’ :

La première étape à partir des quatres fichiers bruts a été de les fusionner dans un seul fichier étant donné que les quatres fichiers possèdent les mêmes champs. Cette fusion a été faite via Excel par simple copier/coller après ouverture des fichiers en indiquant “Windows (ANSI))” comme origine du fichier pour avoir la bonne mise en forme du texte. On obtient alors la table ci-dessous :

Code borne sortie	Borne sortie	Date sortie CNIL (sans secondes)	Code borne retour	retour	Date retour CNIL (sans secondes)
6005	6005 - PLACE EDGAR QUINET	01/01/2019 00:00	2024	2024 - RÉPUBLIQUE / MAUPIN	01/01/2019 00:20
10072	10072 - JACQUES BREL	01/01/2019 00:01	4012	4012 - PLACE ADRIEN GODIEN	01/01/2019 00:09
6037	6037 - CITÉ INTERNATIONALE / INTERPOL	01/01/2019 00:02	10025	10025 - TOTEM	01/01/2019 00:32
3012	3012 - PLACE DU CHÂTEAU	01/01/2019 00:06	7002	7002 - UNIVERSITÉS LYON III / LYON II	01/01/2019 00:26

Le traitement suivant a été de faire en sorte de n’avoir que le nom de la borne dans les champs ‘Borne sortie’ et ‘retour’ et de séparer les champs contenant la date (‘Date sortie’ et ‘Date retour’) pour avoir plusieurs champs : année - mois - jour - heure - minute pour la sortie et le retour. Ces traitements ont été faits sur Excel avec la fonctionnalité Données > Convertir. Ensuite, un nouveau champ ‘trajet’ a été ajouté. Ce dernier contient le résultat de la concaténation entre le champ contenant le nom de la station de sortie et le champ contenant le nom de la station de retour (fonction CONCAT() de Excel). On peut ensuite supprimer le champ contenant le nom de la station de sortie et le champ contenant le nom de la station de retour étant donné qu’ils sont déjà dans la table ‘lieu’. De la même façon que pour la construction des tables précédentes on ajoute les champs ‘num_semaine’ et ‘num_j_sem’. Ces champs ont été construits à partir de la date de sortie. On ajoute finalement un dernier champ ‘id_enr_velo’ dont les valeurs s’incrémentent d’une unité à chaque enregistrement. On obtient alors une table sous cette forme :

id_enr_velo	id_borne_s	id_borne_r	annee_s	mois_s	jour_s	heure_s	minute_s	annee_r	mois_r	jour_r	heure_r	minute_r	num_semaine	num_j_sem	trajet
1	2	30	2019	2	19	16	2	2019	2	19	16	6	8	2	LAFAYETTE_PLACE ANATOLE FRANCE
2	30	2	2019	2	19	16	13	2019	2	19	16	17	8	2	PLACE ANATOLE FRANCE_LAFAYETTE
3	2	8	2019	2	19	16	18	2019	2	19	16	21	8	2	LAFAYETTE_ESPACE SAINT GEORGES
4	8	107	2019	2	19	17	22	2019	2	19	17	33	8	2	ESPACE SAINT GEORGES_SEBASTOPOL

Il ne reste maintenant plus qu’à ajouter un champ ‘id_lieu_s’ et ‘id_lieu_r’ pour pouvoir faire le lien avec la clé primaire de la table ‘lieu’. Pour celà, on concatène donc “v” aux ‘id_borne’ et ‘id_borne_v’ et on les supprime. On obtient alors la table velo (*velo.xls*) :

id_enr_velo	id_lieu_s	id_lieu_r	annee_s	mois_s	jour_s	heure_s	minute_s	annee_r	mois_r	jour_r	heure_r	minute_r	num_semaine	num_j_sem	trajet
1	2v	30v	2019	2	19	16	2	2019	2	19	16	6	8	2	LAFAYETTE_PLACE ANATOLE FRANCE
2	30v	2v	2019	2	19	16	13	2019	2	19	16	17	8	2	PLACE ANATOLE FRANCE_LAFAYETTE
3	2v	8v	2019	2	19	16	18	2019	2	19	16	21	8	2	LAFAYETTE_ESPACE SAINT GEORGES
4	8v	107v	2019	2	19	17	22	2019	2	19	17	33	8	2	ESPACE SAINT GEORGES_SEBASTOPOL
5	107v	70v	2019	2	19	18	23	2019	2	19	18	36	8	2	SEBASTOPOL_FER A CHEVAL

On fait ensuite toujours le même processus d’import de la table sur la base de données.

On crée la table ‘velo’ vide :

```
create table Velo (id_enr_velo int, id_lieu_s int, id_lieu_r
int,annee_s int,mois_s int, jour_s int,heure_s int,minute_s
int,annee_r int,mois_r int, jour_r int,heure_r int,minute_r
int,num_semaine int,num_j_sem int,trajet varchar(100),
                        primary key (id_enr_velo))
```

On importe le fichier *velo.csv* dans cette table. On obtient alors la table ‘velo’ sur la base de données :

	id_enr_velo [PK] integer	id_lieu_s character va	id_lieu_r character v	annee_s integer	mois_s integer	jour_s integer	heure_s integer	minute_s integer	annee_r integer	mois_r integer	jour_r integer	heure_r integer	minute_r integer	num_sem integer	num_j_sem integer	trajet character varying (100)
1	795252	1022v	3016v	2020	1	8	10	0	2020	1	8	10	7	2	3	PLACE TOLOZAN_LIBERTâ€¦ /…
2	795253	10002v	10028v	2020	1	8	10	0	2020	1	8	10	7	2	3	INSA_ECOLE DE MUSIQUE
3	795254	8009v	3101v	2020	1	8	10	0	2020	1	8	10	32	2	3	JEAN XXIII / MERMOZ_PATIN…

La table ‘Transport en commun’ (TC) :

La construction de cette table transport en commun (TC) s’est faite en plusieurs étapes indépendantes. En effet, on a construit d’un côté la table des enregistrements des validations pour le métro lyonnais (*enr_metro_lyon*) et d’un autre coté la table des enregistrements des validations pour tous les transports en commun de Toulouse. On a ensuite fusionné ces deux tables pour obtenir la table finale (*tc.csv*).

Construction de la table des données de métro de Lyon

(enr_metro_lyon) :
Cette table sera à ajouter à la table globale ‘TC’, elle doit donc contenir seulement les champs : id_enr_tc, id_lieu ,nb_valid, annee, mois, jour, heure, minute, num_sem, num_j_sem.
La chaîne de traitement commence à partir des fichiers textes brut de 2019 et 2020 dont la structure est la suivante une fois ouvert dans Excel (UTF-8, séparateur : tab) :

Date jour (CAS tr)	Mois	Année	Code ligne	Station	Jour	Tranche horaire	Heure (CAS tr)	Nb entrées total (CAS tr)	Nb sorties total (CAS tr)
01/01/2019	1	2019	A	Ampère Victor Hugo	2	(vide)	00:00:00	2	4
01/01/2019	1	2019	A	Ampère Victor Hugo	2	(vide)	00:15:00	3	2
01/01/2019	1	2019	A	Ampère Victor Hugo	2	(vide)	00:30:00	3	7

On commence par regrouper le fichier de 2019 avec celui de 2020 (copier/coller Excel). On supprime le champ ‘Jour’ car en contradiction avec le jour du champ ‘Date jour (CAS tr)’. On sépare les champs ‘date’ et ‘heure’ de la même façon que les autres tables pour obtenir les champs ‘jour’, ‘mois’, ‘annee’, ‘heure’, ‘minute’. On peut alors supprimer les champs qui ne nous seront pas ou plus utile : ‘Date jour (CAS tr)’, ‘Mois’, ‘Année’, ‘Code ligne’, ‘Jour’, ‘Tranche horaire’ , ‘Heure (CAS tr)’ et ‘Nb sorties total (CAS tr)’. On garde le champ ‘Nb_entrées tota’l dont on change le nom par ‘nb_valid’ car est équivalent au champ ‘NBVALID’ des données brutes issues des transports en commun toulousains. La raison pour laquelle nous avons supprimé le champ ‘Nb sorties total’ est que nous ne disposons pas de cette

information dans les données brutes issues des transports en commun toulousains. On remplace ensuite le champ 'Station' par l'identifiant ('id_lieu') attribué pour chaque station lors de la création de la table 'lieu_metro_lyon'. Pour ce faire, on réalise une jointure sur le champ 'station' avec la table 'lieu_metro_lyon' pour récupérer l' 'id_lieu' unique grâce au nom de la station avec la fonction RECHERCHEV de Excel. On supprime finalement les enregistrements où on retrouve la présence de lettres (A, B, C ou D) dans le champ 'nb_valid'.
On obtient alors la table *enr_metro_lyon.xls* :

id_lieu	jour	mois	annee	heure	minute	nb_valid
200000	1	1	2019	0	0	2
200000	1	1	2019	0	15	3
200000	1	1	2019	0	30	3
200000	1	1	2019	0	45	1
200000	1	1	2019	1	0	0
200000	1	1	2019	1	15	0

À cette étape, il faut encore regrouper les enregistrements de même date et même station en faisant la somme du nb_valid car nous avons remarqué que cela n'avait pas été fait dans le fichier brut. Il faut également ajouter les champs 'num_sem' et 'num_j_sem'.
Ces traitements se feront une fois que ce fichier aura été fusionné à celui comportant les enregistrements des transports en commun de Toulouse.

Construction de la table des données de transport en commun de Toulouse

(tc_15j_2019_A.csv) :
Pour faciliter les traitements et les imports, nous avons fait le choix de ne travailler que sur les données des 15 premiers jours de 2019 en ce qui concerne les données de transport en commun de Toulouse.

On part du fichier test *ExportCNRS-2019S1.csv* :

ExportCNRS-2019S1

DATE_VALID	HEURE_VALID	NUM_LIEU2	NBVALID	NBDEPL
2019-01-01 00:00:00.000	400	100023	24	24
2019-01-01 00:00:00.000	400	100024	9	9
2019-01-01 00:00:00.000	400	100025	7	7
2019-01-01 00:00:00.000	400	100026	4	4

On ouvre ce fichier brut sur Excel et on garde uniquement les enregistrements des 15 premiers jours de 2019. De la même façon que pour les autres tables, on sépare le champ 'DATE_VALID' pour obtenir les champs 'année', 'mois' et 'jour'. On supprime le champ 'NBDEPL' car nous ne nous intéressons qu'au nombre de validations par station ('NBVALID'). On extrait ensuite les champs 'heure' et 'minute' du champ 'HEURE_VALID' en le divisant par 100 avec la fonction QUOTIENT de Excel pour obtenir le résultat de la

division sans le reste, ce qui correspond à l'heure et en récupérant le reste de cette division avec la fonction MOD de Excel qui correspond à la minute.
On obtient alors le fichier *Tc_15j_2019_simple.xls* :

jour	mois	annee	heure	minute	NUM_LIEU2	NBVALID
1	1	2019	4	0	100023	24
1	1	2019	4	0	100024	9
1	1	2019	4	0	100025	7
1	1	2019	4	0	100026	4

La problématique est maintenant de pouvoir regrouper les enregistrements provenant du même lieu et pour un même type de transport en commun à la même date et à la même heure. En effet, on sait que pour un même lieu il peut exister une station de métro et une station de tram ou bus. De plus, il peut y avoir plusieurs 'NUM_LIEU2' pour une même station de métro ou de bus/tram. Il va donc être important, dans un premier temps, de pouvoir séparer les 'NUM_LIEU2' correspondant à des stations de métro et ceux correspondant à des stations de bus/tram. Et une fois cette séparation faite, on pourra additionner le nombre de validations 'NBVALID' en fonction du nom du lieu.

Séparation par type de transport et agrégation au 'nom_lieu' du fichier

Tc_15j_2019_simple.xls :
Dans un premier temps, nous allons joindre l'information du nom du lieu contenue dans la table *lieu_TC_toulouse_NA* (la table 'lieu' des transports en communs à Toulouse sans agrégation) à partir du champ 'NUM_LIEU2' de la table *Tc_15j_2019_simple*.

Rappel de la structure de la table *lieu_TC_toulouse_NA* :

NUM_LIEU	NOM_LIEU	id_ville	X	Y	type_transport
10	Les Abattoirs	2	526578.1	1844787.5	autre
11	Les Abattoirs	2	526581.2	1844756.6	autre
20	Achiary	2	530268.1	1844710	autre

Ces traitements vont se faire avec des requête SQL, c'est pourquoi nous importons ces deux tables sur une base de donnée (BDD : sigma22 , schéma : velo_covid). Sur cette base la table '*lieu_na'* contient les informations du fichier *lieu_TC_toulouse_NA* et la table '*tc_simple'* contient les informations du fichier *Tc_15j_2019_simple.*
On réalise la jointure avec la requête suivante (le résultat est stocké dans une vue appelée '*jointure'*) :

```
create view "velo_covid"."jointure" as
select
jour,mois,annee,heure,minute,"velo_covid"."tc_simple".num_lie
u, nbvalid, nom_lieu
from "velo_covid"."lieu_na", "velo_covid"."tc_simple"
where "velo_covid"."lieu_na".num_lieu =
```

```
"velo_covid"."tc_simple".num_lieu
```

On obtient alors la vue 'jointure' :

	jour integer	mois integer	annee integer	heure integer	minute integer	num_lieu integer	nbvalid integer	nom_lieu character varying (50)
1	1	1	2019	4	0	100023	24	Arènes
2	1	1	2019	4	0	100024	9	Argoulets
3	1	1	2019	4	0	100025	7	Bagatelle
4	1	1	2019	4	0	100026	4	Bellefontaine

On agrège ensuite au jour, mois, année, heure, minute, nom_lieu pour les métro uniquement (num_lieu > 100000) avec la requête suivante :
Le résultat est stocké dans une vue temporaire appelée '*metro_nom_lieu'* :

```
create view "velo_covid"."metro_nom_lieu" as
select jour,mois,annee,heure,minute, sum(nbvalid) as
s_nbvalid, nom_lieu
from "velo_covid"."jointure"
where num_lieu > 100000
group by jour,mois,annee,heure,minute,nom_lieu
```

	jour integer	mois integer	annee integer	heure integer	minute integer	s_nbvalid bigint	nom_lieu character varying (50)
1	1	1	2019	4	0	24	Arènes
2	1	1	2019	4	0	9	Argoulets
3	1	1	2019	4	0	7	Bagatelle
4	1	1	2019	4	0	62	Balma-Gramont
5	1	1	2019	4	0	16	Barrière de Paris

Il reste à joindre l' 'id_lieu' correspondant au 'nom_lieu' pour les métros via une jointure avec le fichier *lieu_mtc_toulouse.xls* (importé sous le nom '*lieu_mtc'* sur base de données). Cet 'id_lieu' va remplacer le champ 'nom_lieu'. On le fait avec la requête suivante :

```
select id_lieu, s_nbvalid as nbvalid,
annee,mois,jour,heure,minute
from "velo_covid"."lieu_mtc", "velo_covid"."metro_nom_lieu"
where "velo_covid"."lieu_mtc".nom_lieu =
"velo_covid"."metro_nom_lieu".nom_lieu
and id_lieu > 100000
```

On obtient alors la table contenant les enregistrements des métros avec un seul 'id_lieu' par station :

	id_lieu integer	nbvalid integer	annee integer	mois integer	jour integer	heure integer	minute integer
1	100001	24	2019	1	1	4	0
2	100010	9	2019	1	1	4	0
3	100025	7	2019	1	1	4	0

On exporte ce tableau *tc_15j_2019_metro_A.csv*.

On fait la même procédure pour les transports en commun autres que métro. On agrège ensuite au jour, mois, année, heure, minute, nom_lieu pour les autres transports en commun uniquement (num_lieu < 100000) avec la requête suivante :
Le résultat est stocké dans une vue temporaire appelée '*tc_nom_lieu'* :

```
create view "velo_covid"."tc_nom_lieu" as
select jour,mois,annee,heure,minute, sum(nbvalid) as
s_nbvalid, nom_lieu
from "velo_covid"."jointure"
where num_lieu < 100000
group by jour,mois,annee,heure,minute,nom_lieu
```

Il reste à joindre l' 'id_lieu' correspondant au 'nom_lieu' pour les autres transports en commun via une jointure avec le fichier *lieu_mtc_toulouse.xls* (importé sous le nom '*lieu_mtc'* sur la base de données). Cet 'id_lieu' va remplacer le champ 'nom_lieu'. On le fait avec la requête suivante :

```
select id_lieu, cast (s_nbvalid  as int) as nbvalid,
annee,mois,jour,heure,minute
from "velo_covid"."lieu_mtc", "velo_covid"."tc_nom_lieu"
where "velo_covid"."lieu_mtc".nom_lieu =
"velo_covid"."tc_nom_lieu".nom_lieu
and id_lieu < 100000
```

On obtient ce tableau qu'on exporte ensuite sous le nom *tc_15j_2019_tc_A.csv*.

	id_lieu integer	nbvalid integer	annee integer	mois integer	jour integer	heure integer	minute integer
1	16410	1	2019	1	1	4	45
2	60160	2	2019	1	1	4	45
3	36001	4	2019	1	1	4	45

On fusionne ensuite les deux tables que l'on vient d'obtenir pour avoir les informations des métros et autres transports en commun dans la même table : *tc_15j_2019_A.csv (= tc_15j_2019_metro_A.csv + tc_15j_2019_metro_A.csv)*
Cette fusion se fait en ouvrant ces deux fichiers dans un éditeur de texte et en copiant les données de l'un que l'on colle à la suite de l'autre. Cette table contient alors tous les enregistrements de validations des transports en commun toulousains.

Fusion des tables d'enregistrement des transports en commun de Toulouse et Lyon :

On fusionne les deux tables obtenues précédemment (*enr_metro_lyon.xlsx + tc_15j_2019_A.csv)* pour avoir une table qui contient tous les enregistrements de transports en commun *(tc_final.xlsx)*.
Comme pour toutes les autres tables d'enregistrement de notre base de données, on ajoute de la même façon les champs correspondant au jour et numéro de semaine.

On importe sur la base de données cette table que l'on appelle 'tc'.

```
create table tc (id_lieu varchar(50), nb_valid int, annee
int, mois int, jour int, heure int,minute int, num_sem
int,num_j_sem int)
```

Comme indiqué précédemment le dernier traitement consiste à regrouper les enregistrements de même date et même station car on a remarqué que ça n'avait pas été fait dans le fichier brut des données de métro de Lyon.
On crée la nouvelle table 'tc_' dans laquelle le regroupement est effectué :

```
create table tc_ as
select id_lieu, sum(nb_valid) as nbvalid, annee, mois, jour,
heure ,minute, num_sem,num_j_sem
from tc
group by id_lieu, annee, mois, jour, heure ,minute,
num_sem,num_j_sem
```

On obtient alors la table transport en commun finale (appelée 'tc_' sur notre base de données)

	id_lieu character varying (50)	nbvalid bigint	annee integer	mois integer	jour integer	heure integer	minute integer	num_sem integer	num_j_sem integer
1	10	1	2019	1	2	6	30	1	3
2	10	1	2019	1	2	7	0	1	3
3	10	3	2019	1	2	7	30	1	3
4	10	2	2019	1	2	8	0	1	3

Cette table a été exportée pour l'avoir en fichier csv (*tc.csv*).

Tests de requêtage sur notre BDD

(En ayant en tête la maquette de l'application proposée par les M1) :
Après la conception et l'implémentation de la base de données expliquée ci-dessus, nous voulions tester la fonctionnalité de cette base. Nous avons interrogé notre base de données via différentes requêtes SQL afin de valider si elle permettait d'accéder rapidement et facilement à l'information désirée. Ce test de requêtes s'est fait via le client PgAdmin. Nous avons évidemment testé des requêtes qui permettent de renvoyer de l'information structurée de façon à pouvoir être utilisable en vue de représentation graphique ou cartographique qui pourrait se retrouver sur l'application de WebSIG. De plus, ces requêtes ont été pensées de façon à pouvoir mettre en paramètres certains éléments de ces dernières (en gras dans les requêtes). Celà est important pour la conception future de l'application.

Test de requêtage sur les données de mobilité vélo :

Dans un premier temps, on a cherché l'activité totale des stations VLS dans une ville donnée et à une période donnée. Nous avons imaginé une requête permettant d'obtenir chaque station avec ces coordonnées géographiques ainsi que le nombre de sorties, le nombre de retours et l'activité totale de la station (somme des retours et sorties par station).
La requête ci-dessous permet, par exemple, d'obtenir l'activité de mobilité de VLS par station à Toulouse pour les 10 premiers jours de janvier et de février :

```
select s.nom_lieu ,s.x,s.y, nb_s, nb_r, nb_s + nb_r as
act_tot
from (select lieu.nom_lieu, lieu.x,lieu.y, COALESCE(nb_s,0)
as nb_s
from lieu
full join (select nom_lieu ,x,y, count(nom_lieu) as nb_s
from velo_quart, lieu, ville
where nom_ville = 'Toulouse'
              and annee_s = 2019
              and mois_s >= 1 and mois_s <= 2
              and jour_s >= 1 and jour_s <= 10
              and heure_s >= 0 and heure_s < 24
              and minute_s >= 0 and minute_s <= 60
and id_lieu_s = id_lieu and ville.id_ville = lieu.id_ville
group by nom_lieu,x,y) as t
on t.nom_lieu = lieu.nom_lieu
where type_transport = 'velo' and id_ville = 2
) as s,
(select lieu.nom_lieu, lieu.x,lieu.y, COALESCE(nb_r,0) as
nb_r
from lieu
full join (select nom_lieu ,x,y, count(nom_lieu) as nb_r
from velo, lieu, ville
where nom_ville = 'Toulouse'
              and annee_r = 2019
              and mois_r >= 1 and mois_r <= 2
              and jour_r >= 1 and jour_r <= 10
              and heure_r >= 0 and heure_r < 24
              and minute_r >= 0 and minute_r <= 60
and id_lieu_r = id_lieu and ville.id_ville = lieu.id_ville
group by nom_lieu,x,y) as t
on t.nom_lieu = lieu.nom_lieu
where type_transport = 'velo' and id_ville = 2
) as r
where s.nom_lieu = r.nom_lieu
```

	nom_lieu character varying (100)	x numeric	y numeric	nb_s bigint	nb_r bigint	act_tot bigint
1	11 Novembre	1.415	43.593	43	33	76
2	Albi / Jaffary	1.463	43.633	41	54	95
3	Amouroux /Ste Augustine	1.462	43.628	18	15	33
4	Amouroux/ St Tropez	1.467	43.630	28	32	60

L'implémentation de la commande FULL JOIN dans chacune des deux sous requêtes permet de faire une jointure entre la table de sortie (sous requête 1) et d'entrée (sous requête 2) et

d'afficher ainsi toutes les stations existantes même s'il n'y a pas de sorties et/ou retours. La table "lieu" de la base de données nous a permis de calculer la fréquentation des trajets vélo pour un nom_lieu donné et une période choisie.
Le résultat d'une requête de ce type doit pouvoir permettre une représentation sous forme de cercles proportionnels en fonction du champs 'nb_s', 'nb_r' ou 'act_tot' au choix.
L'autre information importante consernant les données de mobilité VLS est le trajet de chaque utilisateur. Nous avons donc imaginé une requête qui permet d'accéder à l'importance de fréquentation de tous les trajets à partir d'une station donnée à une période donnée avec la géométrie (ligne) de chaque trajet avec la possibilité de filtrer en fonction du nombre de trajet. La requête ci-dessous permet, par exemple, d'obtenir l'importance de fréquentation de tous les trajets à partir de la station lyonnaise 'Bourse Du Travail' le 01/01/2019 sans filtrage avec une importance minimum de fréquentation des trajets.

```
select trajet, st_makeline(ls.geom, lr.geom) as geometry,
nb_trajet,lr.x as x_r,lr.y as y_r
from
(select id_lieu_s,id_lieu_r, trajet, count (trajet) as
nb_trajet
from velo, lieu, ville
where nom_ville = 'Lyon' and ville.id_ville = lieu.id_ville
     and nom_lieu = 'Bourse Du Travail' and velo.id_lieu_s =
lieu.id_lieu
              and annee_s = 2019
              and mois_s >= 1 and mois_s <= 1
              and jour_s >= 1 and jour_s <= 1
              and heure_s >= 0 and heure_s <= 24
              and minute_s >= 0 and minute_s <= 60
 group by trajet,id_lieu_s,id_lieu_r) as t
inner join lieu as ls on t.id_lieu_s = ls.id_lieu
inner join lieu as lr on t.id_lieu_r = lr.id_lieu
where nb_trajet >= 1
```

	trajet character varying (100)	geometry geometry	nb_trajet bigint	x_r numeric	y_r numeric
1	BOURSE DU TRAVAIL_AIRIE D...	0102000020E6100...	2	4.849	45.768
2	BOURSE DU TRAVAIL_BOURS...	0102000020E6100...	4	4.848	45.760
3	BOURSE DU TRAVAIL_CALUIR...	0102000020E6100...	1	4.848	45.785
4	BOURSE DU TRAVAIL_CHARP...	0102000020E6100...	1	4.863	45.770

En plus de la géométrie du trajet dans le champ 'geometry', on a la présence des champs 'x_r' et 'y_r' qui contiennent les coordonnées géographiques de chaque station de retour pour chaque trajet. Cette requête doit pouvoir permettre une représentation des trajets sous forme de flux (lignes) "proportionnels" au nombre de trajets. De plus, les champs 'x_r' et 'y_r' doivent permettre de dessiner les points des stations de destination. Un problème potentiel pour une représentation des flux avec cette requête est que les trajets avec la même borne de

sortie et de retour ne sont pas représentés, car la géométrie obtenue ne correspond pas à une ligne.

Test de requêtage sur les données de Transports en commun

Pour le transport en commun, on cherche à comprendre l'effet mobilité des moyens de transport (métro, tram/bus) sur la mobilité vélo. Pour cela, on a testé des requêtes afin d'accéder au nombre de validations pour une station donnée de métro ou de tram/bus soit toutes les heures, soit à un jour donné pour une date choisie :
La requête ci-dessous permet, par exemple, d'accéder au nombre de validations pour toutes les stations de métro de Toulouse le 01/11/2019 :

```
select nom_lieu, geom, type_transport, sum(nb_valid) as
nb_valid
from lieu, tc, ville
where lieu.id_lieu = tc.id_lieu
and lieu.id_ville = ville.id_ville and nom_ville = 'Toulouse'
          and type_transport ='metro'
          and annee = 2019
          and mois >= 1 and mois <= 1
          and jour >= 11 and jour <= 11
          and heure >= 0 and heure < 24
          and minute >= 0 and minute <= 60
group by  nom_lieu, geom, type_transport
```

	nom_lieu character varying (100)	geom geometry	type_transport character varying (50)	nb_valid bigint
1	Arènes	0101000020E6100...	metro	27977
2	Argoulets	0101000020E6100...	metro	7080
3	Bagatelle	0101000020E6100...	metro	7678
4	Balma-Gramont	0101000020E6100...	metro	20705
5	Barrière De Paris	0101000020E6100...	metro	7964

Le résultat d'une requête de ce type doit pouvoir permettre une représentation sous forme de cercles proportionnels en fonction du champs 'nb_valid'.

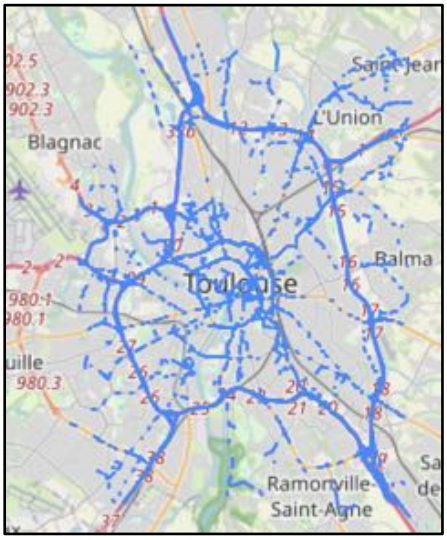
Test de requêtage sur les données de Trafic routier

En termes de trafic routier, nous avons imaginé une requête qui permet d'accéder à la densité du trafic routier en comptant le nombre de signaux gps émis sur chaque tronçon pour une ville et une période donnée. La requête ci-dessous permet, par exemple, d'accéder à cette information le 01/01/2019 à Toulouse entre 8h et 9h.

```
select troncons.link_id, st_name, geom,
count(enr_gps_simple.link_id) as dens
from enr_gps_simple, troncons, ville
where enr_gps_simple.link_id = troncons.link_id
and enr_gps_simple.id_ville = ville.id_ville
and nom_ville ='Toulouse'
and annee = 2019
and mois >= 1 and mois <= 1
and jour >= 1 and jour <= 1
and heure >= 8 and heure <= 9
and minute >= 0 and minute <= 60
group by st_name, geom, troncons.link_id
```

	link_id bigint	st_name character varying (254)	geom geometry	dens bigint
1	57984032	1	0105000020E6100...	3
2	57973958	1	0105000020E6100...	1
3	708956082	12	0105000020E6100...	1

Le champs ‘dens’ indique le nombre de signaux gps émis sur chaque tronçon. En requêtant l’information de géométrie des tronçons on peut les visualiser sur PgAdmin. Celà nous a conforté dans l’idée qu’une représentation cartographique de ces tronçons était possible.



Le temps de réponse des requêtes testées est de 1 à 3 secondes sur PgAdmin pour les trois dimensions de la base de données (vélos, trafic routier et transport en commun). Le rajout d’autres données ou de nouveaux tableaux pourrait sûrement alourdir la base de données et par conséquent faire augmenter le temps de réponse des requêtes. Dans le cadre du futur développement du WebSIG, la partie back end (base de données) nécessitera sans doute des améliorations d’optimisation de performances. Cette amélioration ciblera deux principaux paramètres à savoir :

1. L’optimisation des requêtes

Gardez à l'esprit que plus le nombre d'éléments est faible, plus le traitement est rapide. A facteurs identiques, le traitement d'une instruction SQL simple prend moins de temps que le traitement d'une instruction SQL complexe. De même, à facteurs identiques, les requêtes demandant davantage de données prennent plus de temps que les requêtes demandant moins de données. Afin d’améliorer la performance d'exécution il faudrait éviter, par exemple, les expressions de jointure et de filtres complexes et limiter les expressions de type de données explicites (utilisation des fonctions de calcul) ou implicites (opérations qui exécutent sur des colonnes avec des types de données différents).

2. Amélioration des index

Outre les requêtes, l'autre élément essentiel de la base de données est l'index
D'abord c'est quoi un index ?
Un index, dans une base de données se base sur le même principe qu'un index dans un livre. Avec un index placé sur une ou plusieurs colonnes le système d'une base de données peut rechercher les données d'abord sur l'index et s'il trouve ce qu'il cherche il saura plus rapidement où se trouve les enregistrements concernés.
Lorsque l'indexation est exécutée correctement, on pourrait améliorer les performances de la base de données et accélérer la durée d'exécution des requêtes. En effet, l'indexation permet de créer une structure de données qui organise toutes les colonnes des tables et facilite la recherche des informations demandées.
On pourrait créer des index de la même manière que la syntaxe générale de création d'index SQL:

```
CREATE INDEX <nom_index>
        ON <nom_table> (<list_colonne>)
```

Proposition/ amélioration de la maquette de l'application proposée par les M1 :

Dans cette perspective, le but était d'imaginer une interface interactive permettant d'afficher un certain nombre d'informations simultanément. L'objectif étant d'avoir en main un outil permettant de comparer les trois différentes dimensions d'information contenues dans la base de données, à savoir :

- Trajet des vélos
- Trafic routier
- Transport en commun

Ainsi que des calculs et statistiques pouvant être intéressant à visualiser pour une analyse future.

Contraintes :
L'une des contraintes était de représenter plusieurs informations sur une même vue, sans que cela soit illisible. Représenter toute l'information cartographique ment de façon simultanée, mènerait à avoir une carte ou les différences ne sont pas mises en évidence, notamment à cause du grand volume de données. Un travail est donc à faire sur le plan de la "data-visualisation", spécifiquement celle du flux de transport des vélos entre les stations. L'objectif est en conséquence de séparer au mieux l'information afin d'éviter au maximum la condensation tout en visualisant des données sur les deux villes "Toulouse" et "Lyon" en même temps.

Solutions :
En premier lieu, il a été décidé de concevoir un tableau de bord, contenant différentes représentations de ces trois dimensions. De cette façon, l'information de chacun des paramètres "Météo", "Trajet vélos", "trafic routier" et "transports en commun" sont représentés sur la

même page. Les comparaisons peuvent être faites, non seulement entre les villes, mais entre les différents moyens de transport, afin d'avoir la vue la plus critique possible. Cette proposition a été déjà faite par les étudiants de "master 1 SIGMA", et la maquette éditée par le groupe de travail nous a servis comme base.

Certains éléments se voient être les points forts de cette interface, en lien étroit avec les exigences des commanditaires. Le choix du "jour" grâce à un calendrier en haut à gauche a été retenu, il projette la dimension temporelle indispensable à l'analyse des chercheurs, et auquel une attention particulière a été portée. Deux cartes sont aussi représentées côte à côte, elles permettent en effet la visualisation des statistiques cartographiques pour les deux villes ou pour une même ville avec des paramètres de visualisation différents simultanément afin de pouvoir faire des comparaisons. Nous avons donc conservé le fait de pouvoir sélectionner des paramètres différents via deux formulaires un pour chaque carte. Grâce à des courbes statistiques caractérisant le trafic routier, l'information dimensionnelle est décomposée et mieux mise en évidence, de façon à ce que l'information soit plus lisible. Mais les deux notions de "filtrage thématique" et de "choix des couches" restent les plus intéressantes. Bien qu'évident à prendre en considération lors de la conception d'un SIG complet, il est tout de même intéressant de souligner leur importance, ainsi que la nature de leur contribution à la bonne visualisation de la donnée.

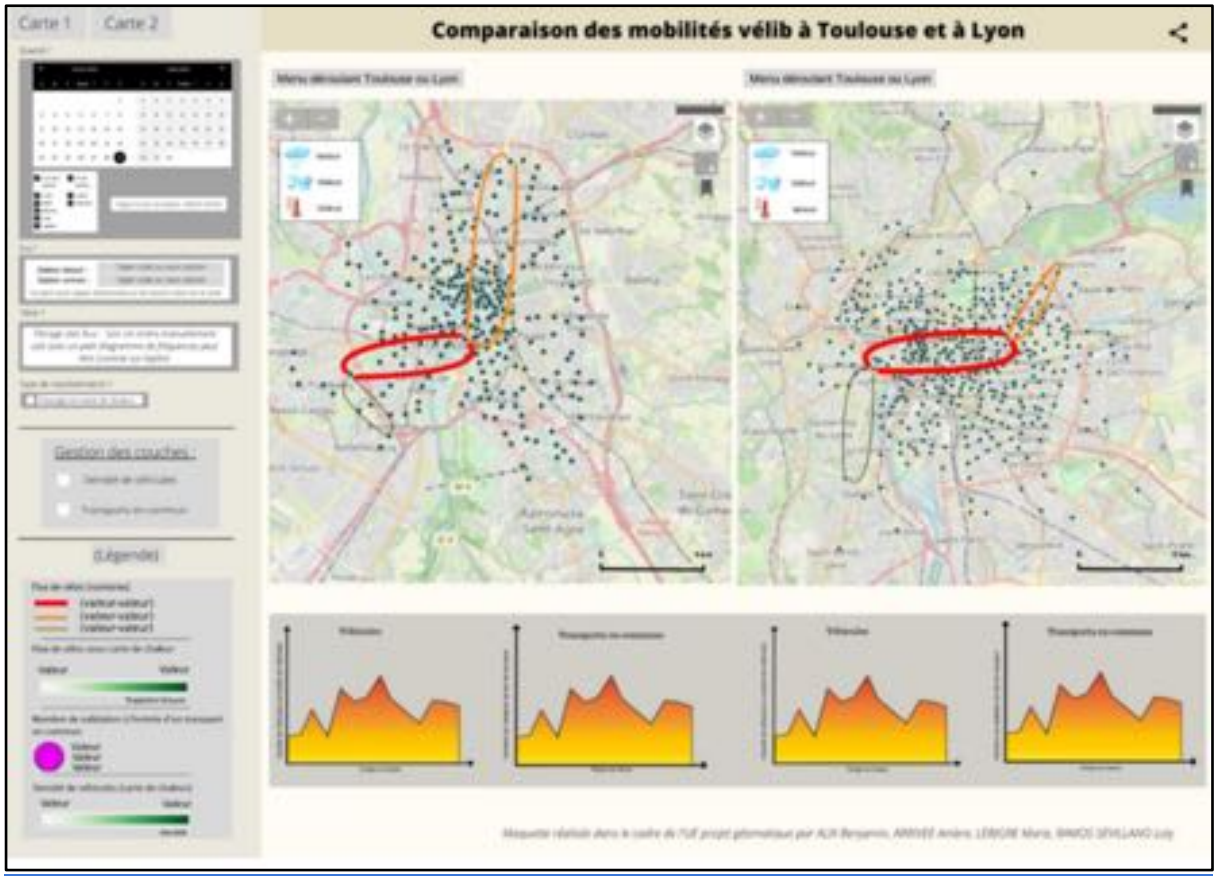


Figure 8 : Maquette initiale proposée par les M1 SIGMA

Évolution de la maquette :
Cela dit, certaines modifications étaient à faire. En premier lieu la représentation du flux qui ne paraît pas très explicite. C'est là où la notion de filtre intervient. La représentation de cette entité étant particulièrement délicate du fait du volume important de données, il a été nécessaire de mettre en place des filtres sur une partie de l'information. Ces filtres peuvent être de nature spatiale ; par exemple les plus au centre-ville ou les plus en périphérie, statistique ; les stations les plus utilisées ou celle avec le plus grand nombre de sorties, ajoutées au filtre temporel.
Pour ces derniers, les exigences des commanditaires ainsi que le séminaire auxquels nous avons participé le 17/02/2022 autour du sujet nous a fait prendre conscience d'un élément à ajouter. En effet, la connaissance du jour de la semaine et semaine de l'année étaient des paramètres à prendre en considération pour beaucoup d'analyses, elles ont donc étés ajoutés comme option de saisie.

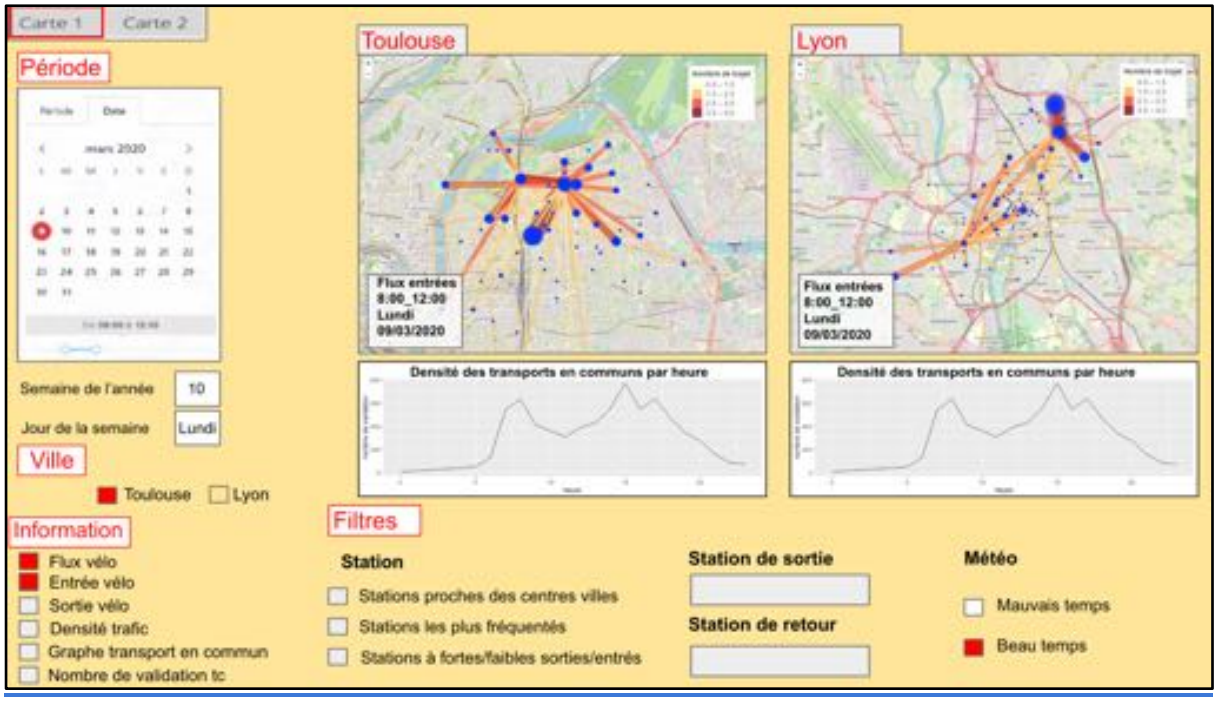


Figure 9 : Maquette améliorée repondant aux attentes des chercheurs

Les informations choisies lors de la sélection sont affichées en bas à gauche de la carte en plus des barres latérales de sélection. Les noms des villes s'affichent également en haut. L'objectif étant d'avoir la vue la plus explicite possible de l'analyse en cours.

Tests requêtage et représentation sur R :

Suite aux tests de requêtage concluants via le client PgAdmin nous avons cherché s'il était possible de réaliser le même type de requête mais directement via le logiciel R Studio de façon à pourvoir exploiter l'information issue des requêtes directement avec les fonctionnalités offertes par R.

Nous avons fait ce choix en se basant sur l'idée de choisir R_shiny comme plateforme de conception de l'application. Ce choix s'explique par deux points :

1. Le fait que les analyses en perspectives à appliquer sur la donnée sont d'ordre statistiques, le logiciel R étant le plus performant dans ce domaine d'application. En effet théoriquement, si les tests sont concluants sur R, cela serait pratique de les transposer directement sur une application produite aussi par ce dernier
2. Les champs d'application de la plupart des chercheures étant les mathématiques appliquées au statistiques, R serai un environnement de travail plus adapté à d'éventuels développements à venir, et d'évolution des analyses sur la donnée

Nous avons trouvé qu'il était possible de récupérer le résultat d'une requête SQL sur une base de données PostgreSQL directement via R Studio grâce à la bibliothèque `RPostgreSQL`. Grâce à cette bibliothèque, il est possible de récupérer le résultat d'un requête SQL sous la forme d'un dataframe de R. Voici ci-dessous un exemple simple :

```
# Chargement du package
library(RPostgreSQL)

# Connexion à la base de données avec le mot de passe et le nom
d'utilisateur
con <- dbConnect("PostgreSQL", dbname = "sigma22",
                 user = "sigma22", password = "6tA8ZBAzjE4GJT44",
host="193.55.175.126" ,port="2002")


# Requête
rs <- dbSendQuery(con, "select * from \"pfaucher\".\"test_trafic\" limit
10")

# Résultat de la requête sous la forme d'un df
df <- fetch(rs, n = -1)

# Déconnexion
dbDisconnect(con)
```

L'avantage d'utiliser une base de données "postgrSQL" est que les tables de cette base de données peuvent contenir des données de type "geometry". Il est donc possible de récupérer suite à une requête SQL des entités avec une géométrie. Cela est intéressant étant donné que la bibliothèque "leaflet" de R sert à cartographier des objets ayant une géométrie. Cependant, les "dataframe" issu du processus expliqué précédemment ne sont pas considéré par "leaflet" comme des objets géométriques bien qu'ils contiennent un champ de type "geometry". Pour pouvoir cartographier ces entités avec "leaflet" par exemple en utilisant leur champ de "geometry" avec il faut convertir le "dataframe" issu de la requête sous la forme d'un objet géométrique avec la fonction `read_sf()` de la librairie "sf". Cette fonction permet de faire cette conversion en même temps que l'application des requêtes de la façon suivante :

```
# Chargement des librairies
library(RPostgreSQL)
library(sf)
```

```
# Connexion à la base de données avec le mot de passe et le nom
d'utilisateur
connexion <- dbConnect("PostgreSQL", dbname = "projet_velos",
                       user = "projet_velos", password =
"ab*rERp#Wz6m", host="193.55.175.126" ,port="2002")

# Requêtage directement avec la fonction read_sf
sf <- read_sf(connexion, query = "requête avec un champ de
geometry ")
```

Il se peut, cependant, pour une raison que nous n'avons pas réussi à déterminer, que R Studio plante (R session aborted) lorsque le résultat de la requête contient un champ de type "géometrie". Nous avons cette erreur sur un de nos trois ordinateurs.
Le fait de pouvoir obtenir le résultat des requêtes SQL directement sous la forme d'un "dataframe" permet d'exploiter ces données avec n'importe quelle fonctionnalité que propose R dont "leaflet" ou bien "R Shiny".
Nous avons ainsi pu tester les requêtes déjà testées via PgAdmin sur R Studio et tester les différents types de représentation envisageables à partir des données recueillies.

Test de représentation cartographique des données de trafic stockées sur notre base de données :

Pour les données de trafic, comme expliqué précédemment, nous avons une requête qui permet de d'accéder au nombre de signaux GPS émis sur chaque tronçon sur une période donnée. Nous avons ainsi testé deux types de représentation avec le "dataframe" issu de cette requête. Une première qui permet de jouer sur l'épaisseur du tronçon en fonction du nombre de signaux GPS émis sur ce dernier et une deuxième qui permet de jouer sur la couleur du tronçon en fonction du nombre de signaux GPS émis sur ce dernier.
Le code permettant la représentation en jouant sur l'épaisseur :

```
library(RPostgreSQL)
connexion <- dbConnect("PostgreSQL", dbname = "projet_velos",
                       user = "projet_velos", password =
"ab*rERp#Wz6m", host="193.55.175.126" ,port="2002")
library(sf)

sf <- read_sf(connexion, query = "select troncons.link_id,
st_name, geom, count(enr_gps_simple.link_id) as dens
from enr_gps_simple, troncons, ville
where enr_gps_simple.link_id = troncons.link_id and
enr_gps_simple.id_ville = ville.id_ville
and nom_ville ='Toulouse'
and annee = 2019
and mois >= 1 and mois <= 1
and jour >= 1 and jour <= 1
and heure >= 8 and heure <= 9
```

```
and minute >= 0 and minute <= 60
group by st_name, geom, troncons.link_id
 ")


library(leaflet)

leaflet(sf) %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  addPolylines(weight = sf$dens/2,smoothFactor = 0.5,
               opacity = 0.8, fillOpacity = 1, label=sf$dens)
```

On obtient la représentation suivante :

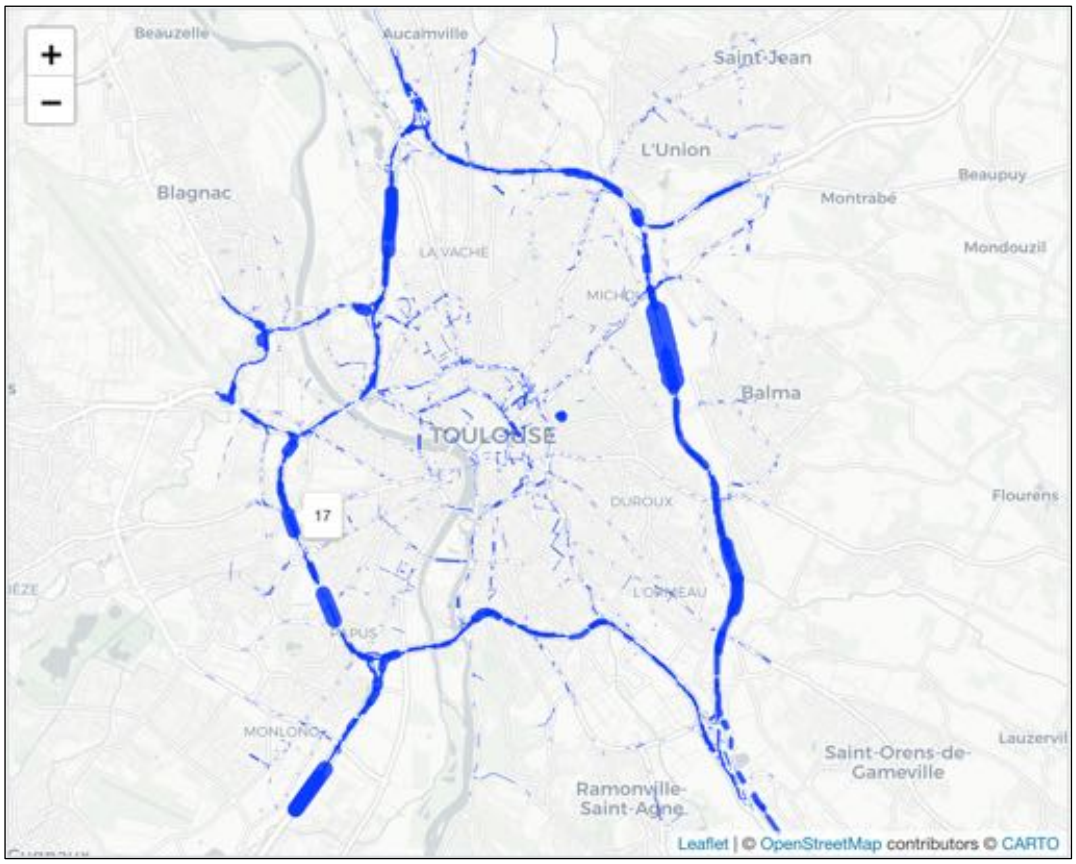


Figure 10 : Représentation graphique du trafic routier en variant l'épaisseur des lignes

Pour cette représentation, on utilise la librairie "leaflet" de R qui permet ici de représenter les tronçons avec la fonction "`addPolylines`". On peut facilement modifier les arguments de cette fonction pour modifier l'aspect visuel notamment pour ajuster la largeur des lignes en fonction de la valeur contenue dans le champ du "dataframe" contenant le nombre de signaux "GPS" émis par tronçon. L'argument 'label' permet d'accéder à l'information du nombre de signaux "GPS" émis sur un tronçon lors du passage de la souris sur ce dernier. Nous n'avons cependant pas réussi à afficher une légende pour faire le lien entre la largeur du tronçon et le nombre de signaux émis et afficher l'échelle sur la carte. On choisit d'utiliser le fond de carte

'Positron' car il permet de bien faire ressortir l'information à représenter tout en gardant des informations pour permettre de la localiser géographiquement.
Le code permettant la représentation en jouant sur la couleur :

```
library(RPostgreSQL)
connexion <- dbConnect("PostgreSQL", dbname = "projet_velos",
                       user = "projet_velos", password =
"ab*rERp#Wz6m", host="193.55.175.126" ,port="2002")
library(sf)

sf <- read_sf(connexion, query = "select troncons.link_id,
st_name, geom, count(enr_gps_simple.link_id) as dens
from enr_gps_simple, troncons, ville
where enr_gps_simple.link_id = troncons.link_id and
enr_gps_simple.id_ville = ville.id_ville
and nom_ville ='Toulouse'
and annee = 2019
and mois >= 1 and mois <= 1
and jour >= 1 and jour <= 1
and heure >= 8 and heure <= 9
and minute >= 0 and minute <= 60
group by st_name, geom, troncons.link_id
 ")


library(RColorBrewer)

# Choix de la couleur de la palette
pall = brewer.pal(n= 9,name = 'YlOrRd')

# choix méthode de discrétisation + calcul des breaks
library(classInt)
ci=classIntervals(sf$dens, 4, style = "jenks")

# fonction qui permet d'attribuer la couleur
library(leaflet)
pal <- colorBin(
  palette = pall,
  domain = sf$dens,
  reverse = FALSE,
  bins=ci$brks
)

# on applique la couleur sur le champ dens
col=pal(sf$dens)

# paramétrage de la carte
leaflet(sf) %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
```

```
  addPolylines(color = col, weight = 2,smoothFactor = 0.5,
               opacity = 0.8, fillOpacity = 1, label =
sf$dens) %>%
  addLegend(
    title = "Nb signal gps/tronçon",
    pal = pal, values = sf$dens, opacity = 0.8)
```

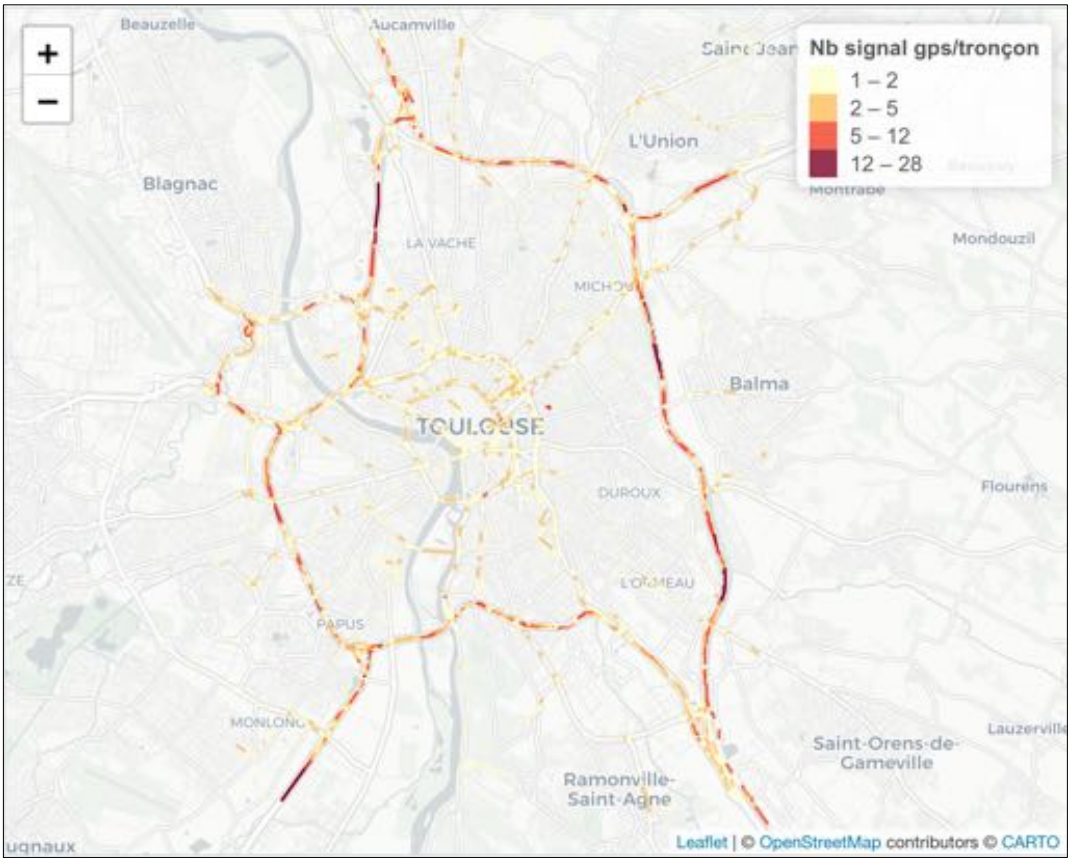


Figure 11 : Représentations du trafic routier en faisant varier la couleur

Pour cette représentation, la logique est la même que précédemment en ajoutant des lignes permettant de configurer la représentation en couleur et la légende associée. Des librairies supplémentaires ont donc été utilisées, à savoir `RColorBrewer` pour pouvoir utiliser des palettes de couleurs pré-existantes et `classInt` pour calculer automatiquement les bornes de chaque classe avec une méthode discrétisation donnée (Méthode de Jenks dans notre exemple) en fonction des valeurs à représenter.

Test de représentation cartographique des données de transport en commun stockées sur notre base de données :

Pour les données de transport en commun, l'objectif est d'observer le nombre de validations pour chaque station sur une période donnée. Une requête SQL nous a permis d'accéder au nombre de validation par station et donc à la fréquence de voyageurs. Nous avons ainsi testé deux types de représentations avec la data frame issu de cette requête. Une première sous format d'un graphique illustrant le nombre de validation en fonction des heures d'un jour donné

et une deuxième, des cercles proportionnels, permet de jouer sur la couleur et le volume des cercles en fonction du nombre de validation pour le type de transport choisi, à savoir métro ou tram/bus.
Code permettant la présentation de nombre de validation en fonction des heures d'une date donnée:

```
library(RPostgreSQL)
con <- dbConnect("PostgreSQL", dbname = "projet_velos",
                 user = "projet_velos", password =
"ab*rERp#Wz6m", host="193.55.175.126" ,port="2002")
requ3 <- dbSendQuery(con, "select nom_lieu, type_transport,
annee, mois,jour,heure, sum(nbvalid) as sum_valid
from lieu, tc_, ville
where lieu.id_lieu = tc_.id_lieu and nom_lieu = 'Empalot'
and lieu.id_ville = ville.id_ville and nom_ville = 'Toulouse'
          and type_transport ='metro'
          and annee = 2019
          and mois >= 1 and mois <= 1
          and jour >= 11 and jour <= 11
          and heure >= 0 and heure < 24
          and minute >= 0 and minute <= 60
group by  nom_lieu, type_transport,annee, mois,jour,heure
")

# Résultat de la requête sous la forme d'un df
df3 <- fetch(requ3, n = -1)   # extract all rows


library(ggplot2)
graph1 <- ggplot(df3) + aes(x =heure)
graph1 + geom_line(aes(y = sum_valid)) +
  labs(title = "", y="nombre de validation")
```

On obtient la représentation suivante :

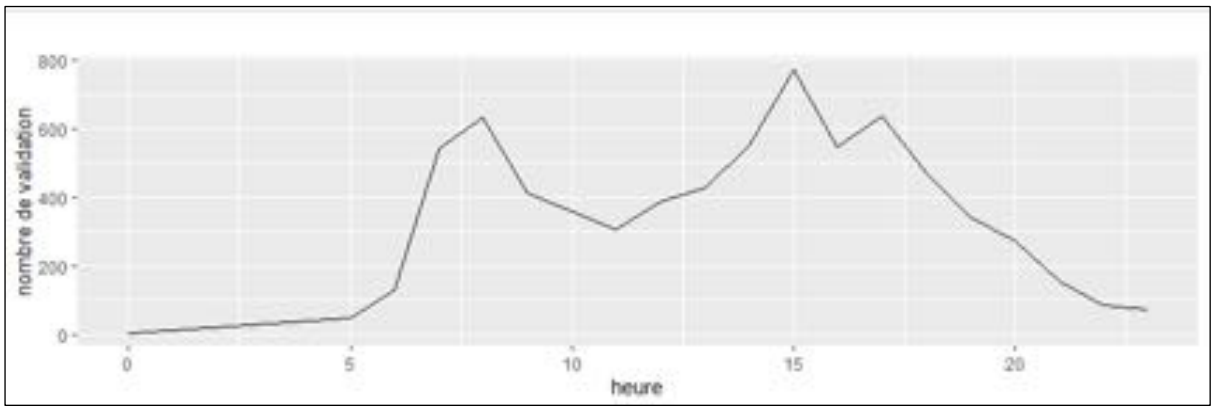
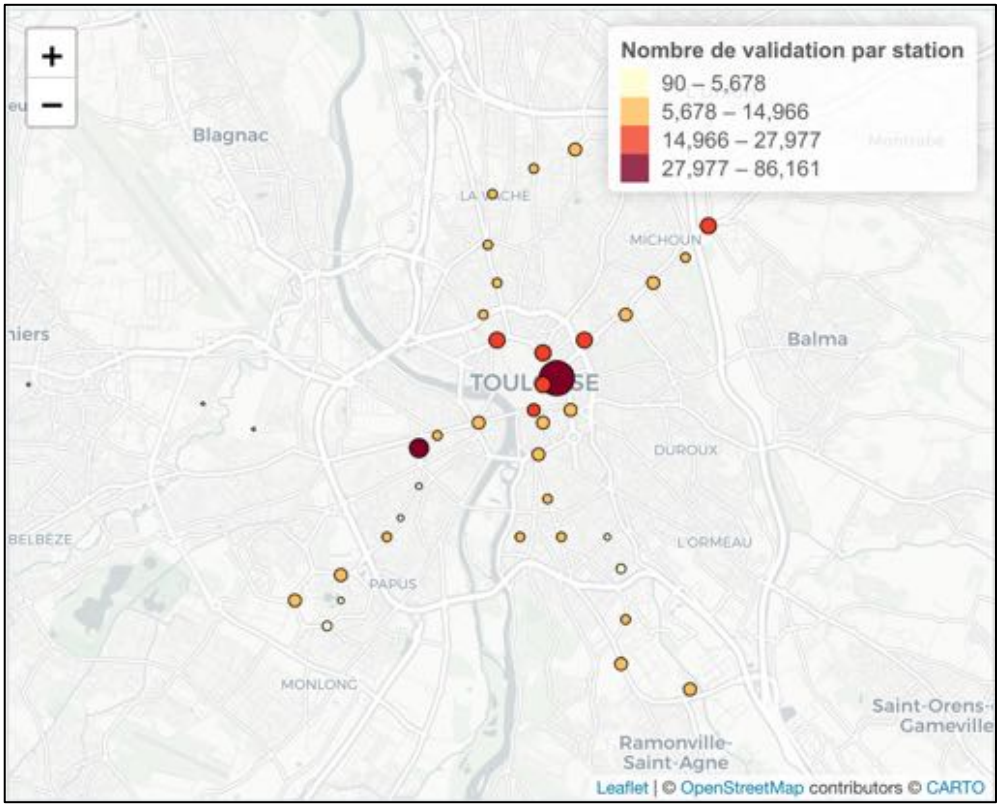


Figure 12 : Graphe représentant le nombre de validation "transports en communs" par heure

Grâce à la fonction "ggplot" de la librairie "ggplot2" de "R", on peut dessiner des graphiques semblables à celui au-dessus en jouant sur les arguments de cette fonction ; le "dataframe" cible et les deux axes (x, y). En local, l'affichage de cette représentation suite à l'exécution du code ci-dessus est quasi instantanée.
Le code permettant la représentation avec des cercles proportionnels et colorés.

```
library(RPostgreSQL)
con <- dbConnect("PostgreSQL", dbname = "projet_velos",
                         user = "projet_velos", password =
"ab*rERp#Wz6m", host="193.55.175.126" ,port="2002")
library(sf)

requ4 <- dbSendQuery(con, "select nom_lieu,x, y,
sum(nb_valid) as nb_valid
from lieu, tc, ville
where lieu.id_lieu = tc.id_lieu
and lieu.id_ville = ville.id_ville and nom_ville = 'Toulouse'
          and type_transport ='metro'
          and annee = 2019
          and mois >= 1 and mois <= 1
          and jour >= 11 and jour <= 11
          and heure >= 0 and heure < 24
          and minute >= 0 and minute <= 60
group by  nom_lieu, x, y
order by nb_valid desc")


# Résultat de la requête sous la forme d'un df
df4 <- fetch(requ4, n = -1)   # extract all rows

# Symbologie couleur
library(RColorBrewer)
# Choix de la couleur de la palette
pall = brewer.pal(n= 9,name = 'YlOrRd')
# choix méthode de discrétisation + calcul des breaks
library(classInt)
ci=classIntervals(df4$nb_valid, 4, style = "jenks")

library(leaflet)
# fonction qui permet d'attribuer la couleur
pal <- colorBin(
  palette = pall,
  domain = df4$nb_valid,
  reverse = FALSE,
  bins=ci$brks
)

# on applique la couleur sur le champ dens
col=pal(df4$nb_valid)
```

```
# paramétrage de la carte


leaflet(df4) %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  addCircles(lng = ~x, lat = ~y, weight = 1, radius =
sqrt(df4$nb_valid), label = df4$nb_valid,color="#000",
             fillColor = col, opacity = 0.8, fillOpacity =
1)%>%
  addLegend(
    title = "Nombre de validation par station",
    pal = pal, values = df4$nb_valid, opacity=0.8)
```



Pour cette représentation, on utilise la librairie "leaflet" de R qui permet ici de représenter les stations avec la fonction "`addCircles`". On peut facilement modifier les arguments de cette fonction pour modifier l'aspect visuel notamment pour ajuster le diamètre du cercle fonction de la valeur contenue dans le champ du "dataframe" contenant le nombre de validations pour chaque station. Pour cet exemple, nous avons d'ailleurs décidé d'attribuer comme valeur de rayon des cercles la racine du nombre de validations pour une meilleure visualisation. Le fait d'ordonner les résultats de la requête avec "`order by nb_valid desc`" permet d'afficher au premier plan les cercles avec le plus petit rayon de façon à ce qu'il soit toujours visible même à proximité des cercles avec un rayon plus important.

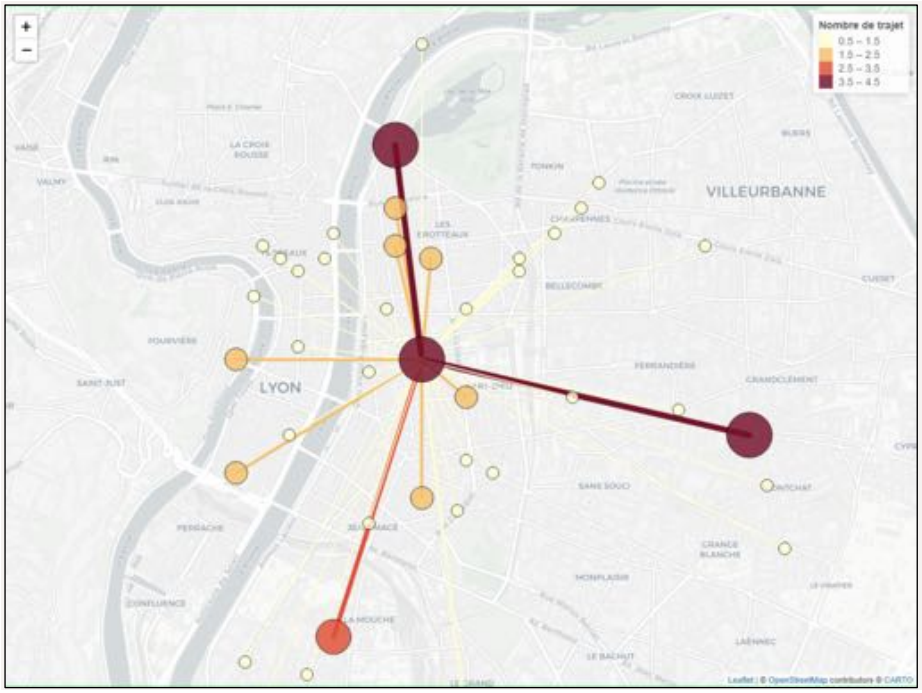
Pour la configuration de la couleur et la légende associée, la logique est la même que celle de la représentation du trafic routier explicité ci-dessus. En local, l'affichage de cette représentation suite à l'exécution du code ci-dessus est quasi instantanée (3 secondes).

Après avec avoir fait des tests de représentation sur R de données de trafic et de transports en commun, la suite logique était de tenter de représenter des données issues de la table 'velo'. Pour tester un type de représentation autre que des cercles proportionnels nous avons décidé de représenter les trajets/ flux de VLS entre stations. Pour une meilleure visibilité sur la représentation cartographique, nous avons tenté de représenter tous les trajets à partir d'une station donnée pour une période temporelle donnée avec la possibilité de filtrage pour ne garder que les flux les plus importants. Voici le code que nous proposons :

```
library(sf)
library(RPostgreSQL)
library(RColorBrewer)
library(classInt)
library(leaflet)
library(glue)

con <- dbConnect("PostgreSQL", dbname = "projet_velos",
                 user = "projet_velos", password =
"ab*rERp#Wz6m", host="193.55.175.126" ,port="2002")

requ4 <- read_sf(con, query = "select trajet,
st_makeline(ls.geom, lr.geom) as geometry, nb_trajet, lr.x as
x_r,lr.y as y_r
from
(select id_lieu_s,id_lieu_r, trajet, count (trajet) as
nb_trajet
from velo, lieu, ville
where nom_ville = 'Lyon' and ville.id_ville = lieu.id_ville
     and nom_lieu = 'Bourse Du Travail' and velo.id_lieu_s =
lieu.id_lieu
          and annee_s = 2019
          and mois_s >= 1 and mois_s <= 1
          and jour_s >= 1 and jour_s <= 1
          and heure_s >= 0 and heure_s <= 24
          and minute_s >= 0 and minute_s <= 60
 group by trajet,id_lieu_s,id_lieu_r) as t
inner join lieu as ls on t.id_lieu_s = ls.id_lieu
inner join lieu as lr on t.id_lieu_r = lr.id_lieu
where nb_trajet >= 1
                                  ")


# Disconnect from the DB
dbDisconnect(con)

# Convert to data.frame
#data.frame(requ4)
```

```
# Symbologie couleur
library(RColorBrewer)
# Choix de la couleur de la palette
pall = brewer.pal(n= 9,name = 'YlOrRd')
# choix méthode de discrétisation + calcul des breaks
library(classInt)
ci=classIntervals(requ4$nb_trajet, 4, style = "jenks")
# fonction qui permet d'attribuer la couleur
pal <- colorBin(
  palette = pall,
  domain = requ4$nb_trajet,
  reverse = FALSE,
  bins=ci$brks
)#qui permet de definir la couleur
col=pal(requ4$nb_trajet)

# Render map

map<- leaflet(requ4) %>%
    addProviderTiles(providers$CartoDB.Positron)%>%
    addPolylines (data = requ4$geometry, weight =
requ4$nb_trajet*2,
                  opacity = 1, color=col)
    addCircles(map= map, lng = ~requ4$x_r, lat = ~requ4$y_r,
radius= requ4$nb_trajet*50, fillOpacity = 0.8 ,fillColor=
col, color="#000", opacity = 0.8, weight = 1)%>%
    addLegend(
      title = "Nombre de trajet",
      pal = pal, values = requ4$nb_trajet, opacity=0.8)
```

On obtient comme représentation :



Ce résultat montre qu'avec notre requête nous pouvons représenter sur R avec la bibliothèque Leaflet à la fois des cercles proportionnels avec des lignes proportionnelles. Cependant, le code précédant est adapté pour représenter de manière correcte les informations issues d'une requête précise. Cela signifie qu'il faut trouver un code permettant de représenter de manière correcte n'importe quelle information qui peut varier par rapport à celle de l'exemple testé en qualité et en quantité.
En matière de temps de réponse, les requêtes testés sur R sont quasiment instantanées sur nos locaux, car l'application consomme les ressources nécessaires sur la machine depuis laquelle elle est exécutée. Néanmoins, nous n'avons pas tester l'application sur un serveur web pour évaluer sa performance.

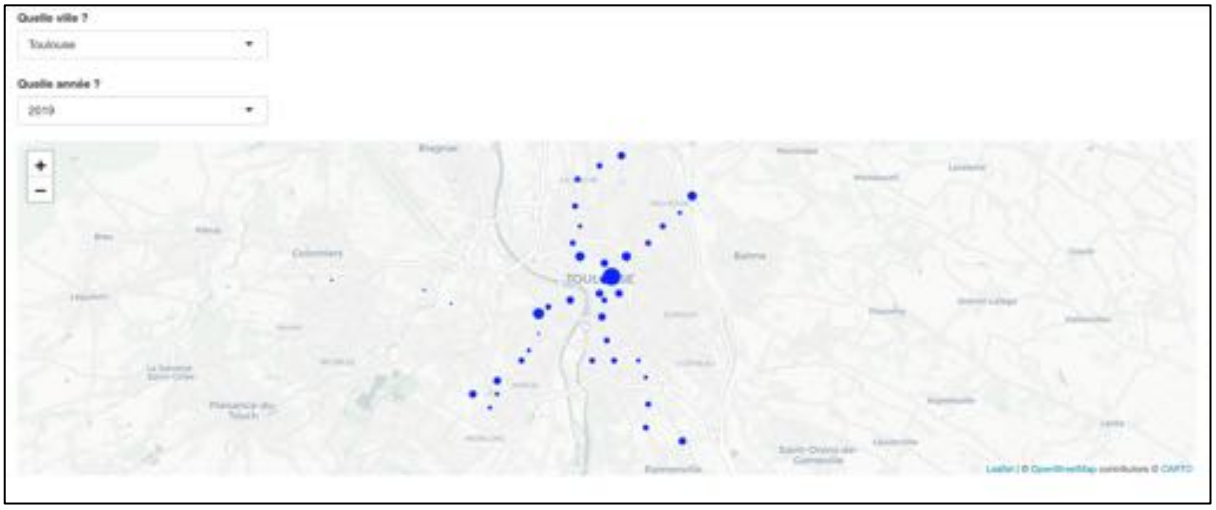
Transposition des résultats en interface R_Shiny

Nous avons vu précédemment qu'il était possible de représenter la donnée issue de requête SQL depuis R. Ces requêtes sont écrites en dur dans le script. Nous avons donc voulu tester s'il était possible de faire en sorte de pouvoir paramétrer ces requêtes avec des paramètres qui dépendent des choix d'un utilisateur via une interface pour ensuite représenter l'information issu de ces requêtes. Nous avons donc cherché à faire le lien entre Rshiny et les fonctions pour faire du requêtage fournies par `RPostgreSQL`. Nous avons remarqué qu'il était possible de réaliser cela simplement avec R Shiny en local. Voici un exemple simple qui permet de l'illustrer :

```
library(RPostgreSQL)
library(leaflet)
library(glue)
```

```
# Définition de la partie interface utilisateur de l'application
ui <- fluidPage(
  selectInput("ville", label = "Quelle ville ?", choices =
c("Toulouse","Lyon")),
  selectInput("annee", label = "Quelle année ?", choices = c(2019,2020)),
  leafletOutput("map"))

# Définition de la partie serveur de l'application
server <- function(input, output) {
  # définition de la fonction data() qui permet de récupérer/renvoyer le
dataframe                                                    issu du
résultat de la requête
  data <- reactive({
    con <- dbConnect("PostgreSQL", dbname = "projet_velos",
                     user = "projet_velos", password = "ab*rERp#Wz6m",
host="193.55.175.126" ,port="2002")

    requ4 <- dbGetQuery(con, glue("select nom_lieu,x, y, sum(nb_valid) as
nb_valid
  from lieu, tc, ville
  where lieu.id_lieu = tc.id_lieu
  and lieu.id_ville = ville.id_ville and nom_ville = '{input$ville}'
          and type_transport ='metro'
          and annee = {input$annee}
          and mois >= 1 and mois <= 1
          and jour >= 7 and jour <= 7
          and heure >= 0 and heure <= 24
          and minute >= 0 and minute <= 60

  group by  nom_lieu, x, y"))


    # Déconnexion de la BDD
    dbDisconnect(con)

    # Convertion du résultat de la requête en dataframe
    data.frame(requ4)
  })

  # On associe à l'output "map" la fonction permettant d'afficher la
représentation cartographique
  output$map <- renderLeaflet({
    leaflet(data=data()) %>%
      addProviderTiles(providers$CartoDB.Positron)%>%
      addCircles(lng = ~x, lat = ~y, weight = 1, radius =
sqrt(data()$nb_valid), label = data()$nom_lieu,
                 opacity = 5, fillOpacity = 1)
  })
}
```

```
shinyApp(ui = ui, server = server)
```



Dans cet exemple l'utilisateur doit sélectionner via une liste déroulante une ville et une année. Les valeurs sélectionnées par l'utilisateur sont mises en paramètre dans la requête avec cette nomenclature `{input$ville}`. Une nouvelle requête est envoyée à chaque fois que l'utilisateur sélectionne une autre valeur, car les fonctions des requêtes de "`RPostgreSQL`" se trouve dans une fonction "`reactive`" spécifique à "R_Shiny". On reprend ensuite le script R permettent d'afficher la représentation cartographique qu'on associe à l'output correspondant à la carte avec le 'render' spécifique à "leaflet" dans "R_Shiny". Pour ce qui est de l'interactivité de cette application simple, on remarque que le changement de ville provoque un affichage de la nouvelle carte quasi instantané. Le changement de l'année (passage de 2019 à 2020) provoque un affichage plus long (5 sec) alors que passage 2020 à 2019 est quasi instantané.

Notre requête étant prévue pour filtrer en fonction d'une période temporelle donnée et d'un type de transport en commun donné, on peut faire une interface qui permet à l'utilisateur de choisir tous ces paramètres. Voici le code permettant de sélectionner une période donnée avec l'ajout de la symbologie couleur sur la représentation cartographique :

```
library(RPostgreSQL)
library(RColorBrewer)
library(classInt)
library(leaflet)
library(glue)

ui <- fluidPage(sidebarLayout(
  sidebarPanel(
  selectInput("ville", label = "Quelle ville ?", choices = c("Toulouse","Lyon")),
  selectInput("tc", label = "Quelle type de transport en commun ?", choices =
c("metro","autre")),
  selectInput("annee", label = "Quelle année ?", choices = c(2019,2020)),
  sliderInput("mois", label= "Intervalle pour le mois", min=1, max=12,
value=c(1,1)),
  sliderInput("jour", label= "Intervalle pour le jour", min=1, max=31,
value=c(11,11)),
  sliderInput("heure", label= "Intervalle pour l'heure", min=0, max=23,
```

```
value=c(0,24)),
  sliderInput("minute", label= "Intervalle pour les minutes", min=0, max=60,
value=c(0,60))),
  mainPanel(
  leafletOutput("map", height = 600))))

server <- function(input, output) {
  # définition de la fonction data() qui permet de récupérer/renvoyer le df issu
du résultat de la requête
  data <- reactive({
    con <- dbConnect("PostgreSQL", dbname = "projet_velos",
                     user = "projet_velos", password = "ab*rERp#Wz6m",
host="193.55.175.126" ,port="2002")

  requ4 <- dbGetQuery(con, glue("select nom_lieu,x, y, sum(nb_valid) as nb_valid
  from lieu, tc, ville
  where lieu.id_lieu = tc.id_lieu
  and lieu.id_ville = ville.id_ville and nom_ville = '{input$ville}'
             and type_transport ='{input$tc}'
             and annee = {input$annee}
             and mois >= {input$mois[1]} and mois <= {input$mois[2]}
             and jour >= {input$jour[1]} and jour <= {input$jour[2]}
             and heure >= {input$heure[1]} and heure <= {input$heure[2]}
             and minute >= {input$minute[1]} and minute <= {input$minute[2]}

  group by  nom_lieu, x, y
  order by nb_valid desc"))


  # Disconnect from the DB
  dbDisconnect(con)

  # Convert to data.frame
  data.frame(requ4)

})

  pal <- reactive({

  # fonction qui permet d'attribuer la couleur
  colorBin(
    palette = brewer.pal(n= 9,name = 'YlOrRd'),
    domain = data()$nb_valid,
    reverse = FALSE,
    bins=classIntervals(data()$nb_valid, 4, style = "jenks")$brks
  )
  })

  col <- reactive({
    pal_ <- colorBin(
      palette = brewer.pal(n= 9,name = 'YlOrRd'),
      domain = data()$nb_valid,
      reverse = FALSE,
      bins=classIntervals(data()$nb_valid, 4, style = "jenks")$brks
    )
    pal_(data()$nb_valid)
  })

  # Render map
  output$map <- renderLeaflet({
leaflet(data=data()) %>%
  addProviderTiles(providers$CartoDB.Positron)%>%
  addCircles(lng = ~x, lat = ~y, weight = 1, radius = sqrt(data()$nb_valid),
label = data()$nb_valid, color="#000",
             fillColor = col(),opacity = 0.8, fillOpacity = 1) %>%
      addLegend(
        title = "Nombre de validation par station",
```

```
        pal = pal(), values = data()$nb_valid, opacity=0.8)
})
}

shinyApp(ui = ui, server = server)
```

On obtient alors une interface de ce type :

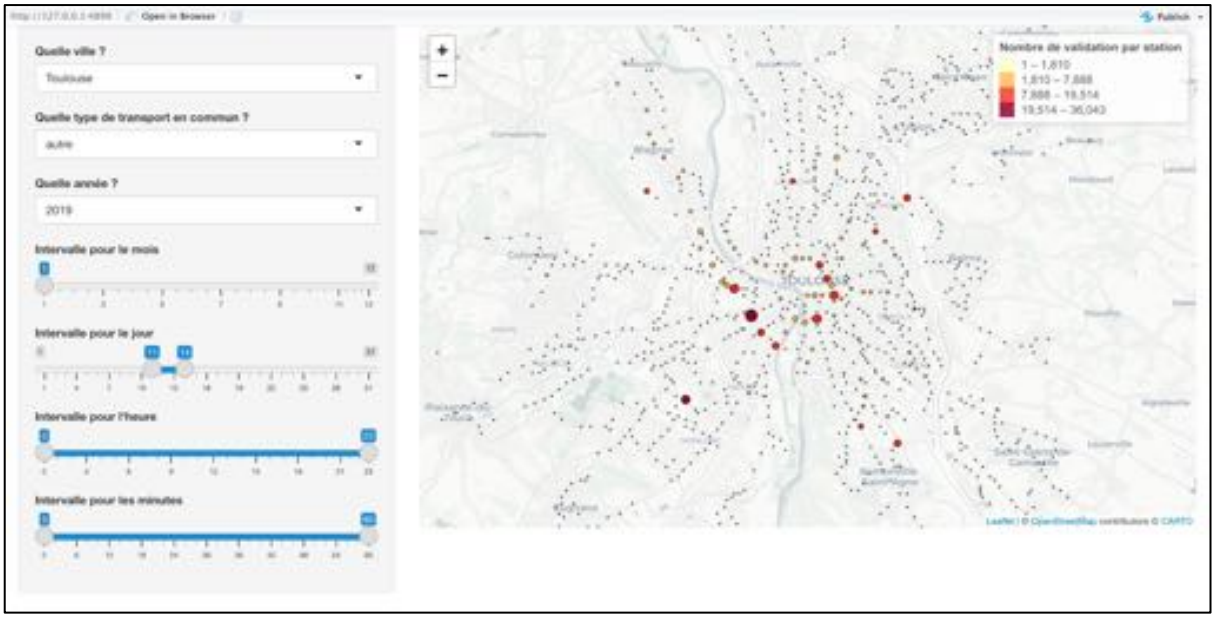


Figure 13 : Interface R_shiny avec sliders de choix de paramétrage pour les données de Transports en communs

Par rapport à l'application précédemment présentée celle-ci est conçue de façon à avoir un panneau contenant les sélecteurs avec à côté un autre panneau contenant la carte. Celà est obtenu avec l'instruction pour l'interface utilisateur de la forme suivante : `sidebarLayout(sidebarPanel(`*...inputs...*`), mainPanel(`*...output de la carte...*`)`.
Comme imaginé dans la maquette de l'interface, notre première idée était de mettre un calendrier pour sélectionner la période mais les valeurs sélectionnées via l'input de type calendrier de R Shiny renvoie des dates alors que qu'il faut des entiers pour notre requête. Cela explique pourquoi nous avons choisi ces sélecteurs en "double curseur" sur cet exemple. Un problème avec ce type de sélecteur est qu'une requête est lancée pour chaque valeur sur le déplacement du curseur. On doit donc attendre un certain temps pour obtenir le résultat sur la période désirée après déplacement d'un curseur. Pour ce qui est de la réactivité de l'outil sur cet exemple, on remarque que lorsque la période sélectionnée par l'utilisateur est assez large tout changement resulte sur un nouvel affichage quasi instantané. Cependant, lorsque la période sélectionnée est très précise (voir *figure 14*) l'affichage dure beaucoup plus longtemps (10 sec).
De plus, des erreurs comme
"les indices négatifs ne peuvent être mélangés qu'à des 0" ou "*'type' (list) de l'argument incorrect"* apparaissent lorsqu'aucun enregistrement ne correspond à la sélection.

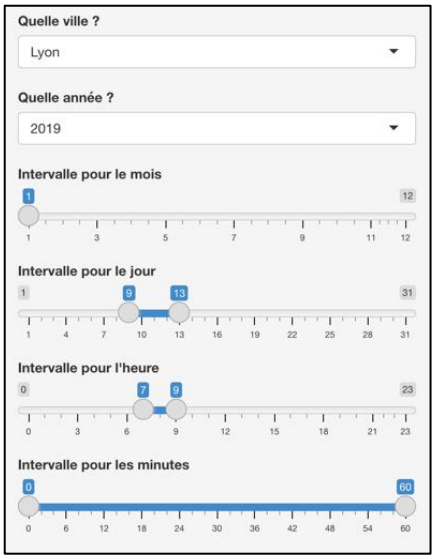


Figure 14 : Exemple de sélection dite « précise »

Nous avons ensuite voulu tester une représentation autre que ponctuelle via R Shiny. Pour cela nous avons fait le choix de représenter des trajets de VLS. Pour une meilleure visibilité sur la représentation cartographique, nous avons eu l'idée de faire une interface permettant de la même façon que pour les autres applications de choisir la ville et la période temporelle mais avec en plus la possibilité d'afficher uniquement des données d'une station de VLS. En effet, l'exemple d'application suivant permet de représenter les flux pour une station donnée avec possibilité de filtrage pour garder les flux les plus importants. Le code de cette application est le suivant :

```
library(sf)
library(RPostgreSQL)
library(RColorBrewer)
library(classInt)
library(leaflet)
library(glue)

ui <- fluidPage(sidebarLayout(
  sidebarPanel(
  selectInput("ville", label = "Quelle ville ?", choices = c("Lyon","Toulouse")),
  selectInput("annee", label = "Quelle annne ?", choices = c(2019,2020)),
  textInput("station", label= "Quelle station ?", value = 'Bourse Du Travail'),
  sliderInput("filtre", label= "Afficher des trajets avec une frequentation d'au
moins :", min=1, max=20, value=1),
  sliderInput("mois", label= "Intervalle pour le mois", min=1, max=12,
value=c(1,1)),
  sliderInput("jour", label= "Intervalle pour le jour", min=1, max=31,
value=c(1,1)),
  sliderInput("heure", label= "Intervalle pour l'heure", min=0, max=23,
value=c(0,23)),
  sliderInput("minute", label= "Intervalle pour les minutes", min=0, max=60,
value=c(0,60))),
  mainPanel(
  leafletOutput("map", height = 600))))
```

```
server <- function(input, output) {
  # définition de la fonction data() qui permet de récupérer/renvoyer le df issu
du résultat de la requête
  data <- reactive({
       con <- dbConnect("PostgreSQL", dbname = "projet_velos",
                        user = "projet_velos", password = "ab*rERp#Wz6m",
host="193.55.175.126" ,port="2002")

  requ4 <- read_sf(con, query = glue("select trajet, st_makeline(ls.geom,
lr.geom) as geometry, nb_trajet, lr.x as x_r,lr.y as y_r
from
(select id_lieu_s,id_lieu_r, trajet, count (trajet) as nb_trajet
from velo, lieu, ville
where nom_ville = '{input$ville}' and ville.id_ville = lieu.id_ville
    and nom_lieu = '{input$station}' and velo.id_lieu_s = lieu.id_lieu
        and annee_s = {input$annee}
        and mois_s >= {input$mois[1]} and mois_s <= {input$mois[2]}
        and jour_s >= {input$jour[1]} and jour_s <= {input$jour[2]}
        and heure_s >= {input$heure[1]} and heure_s <= {input$heure[2]}
        and minute_s >= {input$minute[1]} and minute_s <= {input$minute[2]}
 group by trajet,id_lieu_s,id_lieu_r) as t
inner join lieu as ls on t.id_lieu_s = ls.id_lieu
inner join lieu as lr on t.id_lieu_r = lr.id_lieu
where nb_trajet >= {input$filtre}
"))
return (requ4)

  # Disconnect from the DB
  dbDisconnect(con)

  # Convert to data.frame
  #data.frame(requ4)
})
pal <- reactive({

  # fonction qui permet d'attribuer la couleur
  colorBin(
       palette = brewer.pal(n= 9,name = 'YlOrRd'),
       domain = data()$nb_valid,
       reverse = FALSE,
       bins=classIntervals(data()$nb_trajet, 4, style = "jenks")$brks
       )
  })

  col <- reactive({
       pal_ <- colorBin(
       palette = brewer.pal(n= 9,name = 'YlOrRd'),
       domain = data()$nb_trajet,
       reverse = FALSE,
       bins=classIntervals(data()$nb_trajet, 4, style = "jenks")$brks
       )
pal_(data()$nb_trajet)
})


  # Render map
  output$map <- renderLeaflet({
       map<- leaflet(data()) %>%
```

```
        addProviderTiles(providers$CartoDB.Positron)%>%
        addPolylines (data = data()$geometry, weight = data()$nb_trajet*2,
                      opacity = 1, color=col())
        addCircles(map= map, lng = ~data()$x_r, lat = ~data()$y_r, radius=
data()$nb_trajet*50, fillOpacity = 0.8 ,fillColor= col(), color="#000", opacity =
0.8, weight = 1)%>%
        addLegend(
        title = "Nombre de trajet",
        pal= pal(), values = data()$nb_trajet, opacity=0.8)

})
}

shinyApp(ui = ui, server = server)
```

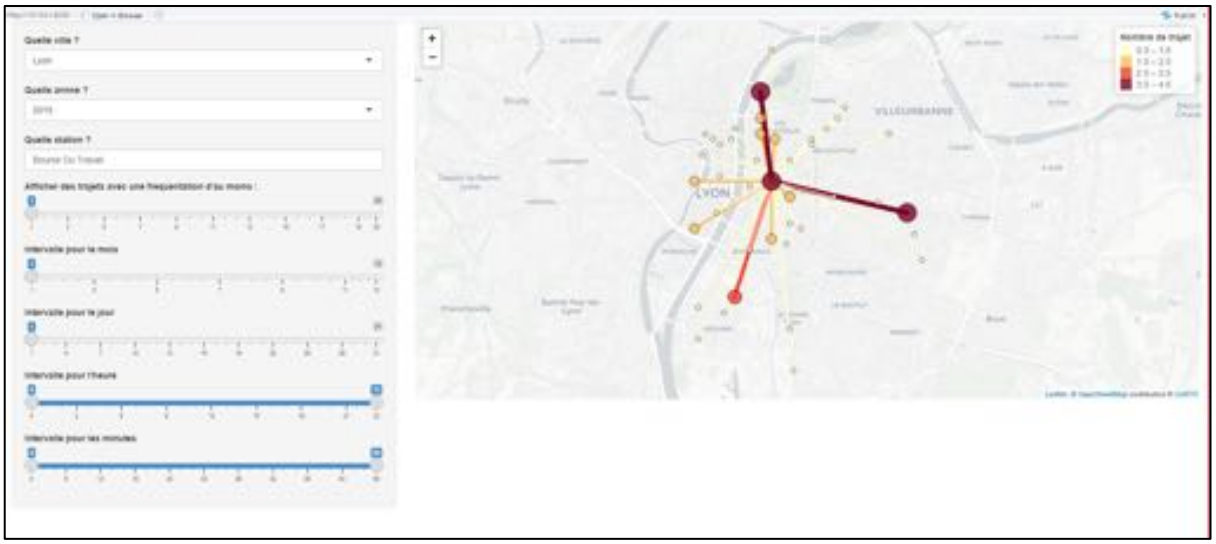


Figure 15 : Interface R_shiny avec sliders de choix de paramétrage pour les données de flux de trajets vélos

À la différence des applications précédente, il faut convertir la géométrie des entités issues de la requête. C'est pourquoi, on utilise la fonction "`read_sf`" plutôt que "`dbGetQuery`" lors de l'application des requêtes. Il ne faut pas oublier de renvoyer le résultat de la fonction "`read_sf` déjà convertit sous forme "dataframe" avec l'instruction "`return`" `(requ4)` dans notre exemple. Pour la réactivité et l'apparition d'erreur, nous avons avec cette application les mêmes problèmes que pour les autres. Nous n'avons pas optimisé l'input où l'utilisateur saisit le nom de la station VLS recherchée. En effet, une requête est envoyée à chaque fois que la chaîne de caractères saisie varie d'un caractère lors de la suppression par exemple, ce qui peut générer des erreurs. Il en résulte que cette application est encore très perfectible, mais peut servir de base pour approfondir l'affichage d'entités à géométrie linéaire via "R_Shiny".

Conclusion

Le travail réalisé durant ce projet est un travail exploratoire qui a permis de faciliter une future conception de l'application de "WebSIG" désirée par les chercheurs. Nous avons par-ailleurs pu dialoguer avec les fournisseurs de la donnée pour ajuster les anomalies détectées sur la donnée. Nous avons ensuite donné des préconisations pour la réalisation de la chaîne de traitement jusqu'à l'import sur la base de données. Il reste cependant à automatiser tous ces traitements. Le choix du logiciel R_shiny pour le développement de l'application a été pris à un stade avancé de la réalisation, en liens avec le domaine d'application des équipes de recherches ainsi que le type d'analyses souhaitées sur les données. Grace aux différents tests « SQL » que l'équipe a pu réaliser sur « PG_Admin » et le logiciel "R", il a été montré que la base de données mise en place est bien fonctionnelle et caractère évolutif, de façon à pouvoir chercher n'importe quelle information sur n'importe quelle table de la base, et d'ajouter de nouveaux champs, de nouvelles tables ou de nouveaux enregistrements si besoin. Cela dit par manque de temps, le développement sur "R_shiny" n'a pas pu atteindre un stade avancé, mais la base nécessaire a un travail futur en continuité de celui qui a été réalisé sont bien mises. Par ces bases, nous faisant référence a :

1. Une base de données complète et fonctionnelle.

2. Une méthodologie bien détaillée sur "R" et "R_shiny".

3. Une maquette de l'interface répondant aux attentes des chercheurs.

4. Des perspectives de développement avenir.

Parmi ces perspectives, il serait intéressant de poursuivre le travail d'agrégation temporelle des données de trafic en prenant compte des informations de position, cap et vitesse. Il pourrait aussi être intéressant de travailler sur une typologie des stations (proche du centre-ville, très/peu fréquentée, ect) comme imaginé sur l'interface de l'application et de l'implémenter directement sur la base de données. De la même façon, un travail de typologie sur la météo (beau/mauvais temps, ect) à réfléchir en fonction de l'agrégation choisie par l'utilisateur pourrait être nécessaire pour l'application finale.

Sources/ données externes :

- Localisation des stations de VLS (Data Toulouse Metropole, 2021)
https://data.toulouse-metropole.fr/explore/dataset/station-velo-toulouse/export/

- Localisation Stations Vélo'v de la Métropole de Lyon (Data Grand Lyon)
https://data.grandlyon.com/jeux-de-donnees/stations-velo-v-metropole-lyon/telechargements

- Localisation des stations de métro de Lyon
https://data.grandlyon.com/jeux-de-donnees/entrees-sorties-stations-metro-reseau-transports-commun-lyonnais/telechargements

Annexes et autres livrables :

En plus de ce rapport, nous fournissons d'autres livrables listés ci-dessous :

- le fichier *pb_data_test_metro_lyon.docx* qui recense les différents problèmes identifiés lors de l'exploration de la donnée brute de test des métros lyonnais
- les fichiers de données brutes en opendata utilisés (localisation des stations VLS de Toulouse et Lyon ainsi que la localisation des stations de métro de Lyon)
- le script python *autom_trafic_test.py* permettant de fusionner et formatter les fichiers bruts de test conportant les données de trafic routier
- les fichiers des tables finales importées sur BDD avec en plus les tables intermédiares des grandes étapes pour obtenir les tables finales.
- la maquette de l'interface au format pptx
- fichiers .R (représentation/ requêtage + applications shiny)

Table des matières

Table des Figures

ANNEXE 2 : grille d'évaluation

Ceci est un document présentant notre grille d'évaluation. Afin de faciliter la lecture de cette dernière, nous avons représenté chaque WebSIG de façon individuelle avec le nom et le lien hyper texte pour chacun d'entre eux.

	Pas un WebSIG mais une méthode de représentation graphique qui peut être intéressante sur l'utilisation de la voiture/camion avant et après covid https://dataviz.cerema.fr/trafic-routier/
Présentation générale de la carte	
Fond de carte	Pas de fond de carte
Légende	
Fonctionnalités	– Curseur d'échelle temporelle qui permet d'augmenter et diminuer la période représentée sur le graphique. – Passer la souris sur le graphique permet de voir les statistiques pour chaque jour. – Possibilité de mettre en place une sélection pour choisir uniquement les jours de la semaine ou bien les samedis ou les dimanches. – Les données peuvent aussi être exprimées sur différentes entités géographiques : la France, les régions, les grandes agglomérations, les frontières.
Analyse des données	
Les variables mises en avant	– On possède des données sur la circulation des voitures et les camions.
Données brutes ou calculées	– Ici on accède à aucune donnée brute mais à des variables calculées. – Un trafic de référence = moyenne du trafic journalier sur une période de référence avant le covid. – Le trafic quotidien est ensuite présenté sous forme d'un indice simple (Trafic du jour - trafic de référence) / trafic de référence. – Ceci permet une représentation par rapport à 0, moyenne du trafic journalier "avant crise" avec ensuite des variations négatives ou positives de l'indice. – Enfin le graphique propose une moyenne glissante qui met en valeur les tendances sur une temporalité de 7 jours.
Date des données	Les données sont régulièrement actualisées puisqu'étant actuellement le 27 octobre 2021 lors de la consultation, on constate que les plus récentes datent du 25.
Visualisation des données	
Type de représentation – (Respect de la sémiologie ?)	Pas de carte
Représentations graphiques	Comme expliqué précédemment, on a un graphique qui exprime un indice par rapport à une moyenne de référence. En ordonnée, on retrouve la part en pourcentage par rapport à cette moyenne et en abscisse les différents mois qui ont suivi cette moyenne temporelle jusqu'à aujourd'hui.
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	La page HTML semble contenir des bibliothèques JavaScript, le tout fonctionnant grâce à l'application Rshiny.

Logiciels serveurs	Serveur sous R Très performant pour la création de graphiques interactifs, mais nécessite une bonne connaissance et maîtrise du codage sous R
Logiciels de base de données	Absence d'informations.
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles…	Ici comme dit plus haut il ne s'agit pas d'un WebSIG puisqu'il n'y a pas d'information spatialisée. Néanmoins, ce site est digne d'intérêt puisqu'il propose, comme va le chercher le futur WebSIG, une analyse temporelle avant et après covid. Ici on met en avant l'évolution du trafic routier qui pourrait être finalement remplacée par l'évolution du trafic de vélos en libre-service (VLS). De plus, ce site est également un moyen de démontrer la puissance de l'application Rshiny dans la création de graphique interactif avec l'utilisateur comme nous avons pu le mettre en valeur dans le listing des fonctionnalités. La proposition d'une dimension temporelle ici se fait par une barre temporelle modulable avec une agrégation des données par journée. Comme nous l'avons montré également dans la décomposition de nombreux indices, des calculs intéressants ont été réalisés (indice à partir d'une moyenne sur une situation post-covid, moyenne glissante...) et peuvent pourquoi pas être reportés dans l'analyse des données vélos. Pour conclure, cette application peut venir, pourquoi pas, donner des idées dans l'analyse des données aux chercheurs. Elle peut également, se révéler inspirante dans l'association sur le WebSIG d'une représentation visuelle et d'une ou plusieurs représentations graphiques. A ce titre, l'application Rshiny, développé par RStudio, est un outil intéressant permettant le développement de ce type de graphique mais aussi de carte.

	WebSIG créé pour la visualisation des flux des vélos en libre-service à Lyon en 2011 (*pas d'accès à l'application mais à un article mappemonde*) http://mappemonde-archive.mgm.fr/num43/fig14/fig14301.html
Présentation générale de la carte	
Fond de carte	Fond de carte discret, pas trop chargé d'information : réseau routier, hydraulique, pas de toponymie, ni de couleurs vives. On retrouve enfin les différents points de retrait de VLS.
Légende	Une légende claire apparaît pour chaque carte en cours de visualisation (après une requête). Elle est simple et accessible.
Fonctionnalités	Peu de fonctionnalité : – Simple zoom et dézoom. – Panneau de requête (les sélections temporelles se font depuis celui-ci) Les requêtes se basent sur plusieurs critères : **Qui** ? : fourchette d'âge, code postal, genre. **Quand** ? A quelles dates, horaires, jours de semaine. **Où** ? On sélectionne les stations départs/arrivées (on peut aussi directement cliquer sur la carte). **Sélection** ? Permet de filtrer les flux selon leur importance, leur longueur ou leur durée → c'est un moyen de rendre la carte plus visible mais aussi une solution au problème de lisibilité des cartes de flux. – Fenêtre de légende. – Fenêtre qui résume la demande en cours. – Graphiques de distributions statistiques qui résument les composantes (a-spatiales) des flux sélectionnés.
Analyse des données	
Les variables mises en avant	Les données qui sont cartographiées sont brutes. Il s'agit de l'heure exacte de la prise en main d'un vélo à une station de départ puis à l'heure exacte du dépôt dans une station d'arrivée. A cette donnée s'ajoute l'ensemble des données personnelles sur les adhérents (dans cette étude, seuls les flux des adhérents ont été pris en compte : genre, âge, adresse).
Données brutes ou calculées	Les données au sein de la base de données sont donc brutes, néanmoins à partir de celle-ci lors de requête, le système du WebSIG est capable (plutôt la base de données elle-même) de réaliser des calculs simples pour des variables secondaires : durée de déplacement, distance à vol d'oiseaux.
Date des données	Les données couvrent l'année 2011, elles ne semblent pas avoir été mises à jour.
Visualisation des données	
Type de représentation – (Respect de la sémiologie ?)	Ici les données ont été représentées par des lignes de flux classiques entre un point A de départ et un point B d'arrivée. Pour pallier au problème classique de cette représentation (lisibilité). Les flux sont donc représentés de manière agrégée sur le principe d'une courbe de Bézier "orientée" pour représenter un mouvement origine-destination et une représentation alliant couleur, épaisseur et plan de représentation (Exemple : flux de forte intensité en rouge, épais et premier plan, inverse pour de faible intensité)
Représentations graphiques	La force de cette application est qu'elle intègre au viewer de la carte des représentations, des informations spatiales avec des infos a-spatiales sous forme de graphique de distribution.On a par exemple pour un groupe de flux sélectionnés, d'après les critères de requêtes des représentations avec toujours en ordonnée le nombre de trajet et une variable en abscisse comme le jour de la semaine, les mois, l'heure, l'âge.
Caractéristiques techniques du WebSIG	

(avantages/ inconvénients)	
Logiciels clients	Client constitué de JavaScript, jQuery, D3.JS, Leaflet. Ici c'est le serveur qui est dynamique : la bibliothèque D3.JS qui réceptionne les données requêtées en format JSON et qui trace les courbes de Béziers, représentatives du mouvement origine-destination avec les règles de sémiologie dynamique (hiérarchie présentée auparavant). C'est le client qui représente aussi les diagrammes de distributions.
Logiciels serveurs	Pas de serveur cartographique, mais un serveur internet apache qui reçoit la requête via AJAX vers une tâche PHP. A partir de là, Apache interroge la base de données en Structured Query Language (SQL).
Logiciels de base de données	Base de données PostgreSQL /PostGIS
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles…	Ce WebSIG est très intéressant car il se base sur un jeu de données, du point de vue des vélos, similaire à celui du projet actuel. Leur traitement, la conception, les outils utilisés pour construire le WebSIG peuvent donc largement servir d'inspiration pour le futur du WebSIG. Néanmoins il faut se méfier car ce travail ayant été réalisé il y a déjà 10 ans, les méthodes de représentations, les outils ont pu évoluer. Ici la gestion temporelle se fait par requête en tapant directement dans le filtre dédié. L'agrégation minimal temporelle se fait à l'heure. La représentation des données se fait par une représentation agrégée bipoint avec une largeur de ligne proportionnelle à la quantité. Ce choix associé à la possibilité pour l'utilisateur dans le requêtage de sélectionner les déplacements selon des critères spatio-temporels ou de représentativité, permet de pallier aux problèmes de visualisation que peut entraîner ce type de représentation. La méthode utilisée pour l'agrégation est la courbe de Bézier, avec une illustration par la taille, la couleur et le plan. A ce point positif s'ajoute le complément de l'information spatiale par des diagrammes de distributions. Ces diagrammes semblent avoir été réalisés par le client. En conclusion cette application est assez représentative de l'attendu des chercheurs pour notre projet, elle permet une visualisation facile des données avec une architecture informatique reproductible. Néanmoins, le type des représentations ont pu évoluer et aujourd'hui l'utilisation de carte de chaleur pour représenter les flux est de plus en plus en mise en avant.

	Application en WebSIG de représentation des vélos à Helsinki en Finlande https://towardsdatascience.com/visualization-of-bike-sharing-system-movements-in-helsinki-with-an-interactive-flow-map-451d897104fe
Présentation générale de la carte	
Fond de carte	Fond de carte classique issue d'OpenStreetMap avec les principaux axes généralisés, les masses d'eau, les terres immergées dans des teintes très foncées (bleu nuit) pour venir contraster avec les flux qui sont très colorés. On retrouve également quelques toponymes de localisation : villes, grands parcs, mers, dont la visibilité est directement reliée à l'échelle d'étude.
Légende	La légende ici est extrêmement simple, même si certains points posent question : pourquoi les stations de vélos qui sont représentées par un symbole ponctuel sont-elles dans la légende sous forme d'un aplat de couleur ?
Fonctionnalités	– On peut zoomer et dézoomer avec la molette (on observe d'ailleurs lorsque nous sommes totalement dézoomer qu'il s'agit d'une projection Mercator). – On peut afficher/désafficher la légende, celle-ci est fixe. – On peut changer la langue. – On peut passer dans un mode 3D qui offre un autre point de vue sur la visualisation des données. – On a enfin à notre disposition un outil de sélection par dessin d'un polygone ou d'un rectangle qui va sélectionner les mouvements à l'intérieur de celui-ci, ou bien les mouvements qui l'intersectent. – Quand on clique sur un flux on a des informations de la base de données qui s'affichent : l'identifiant de l'agrégation, le nombre déplacements qui ont été agrégés sur cette route, les coordonnées géographiques des stations d'origines et de départs.
Analyse des données	
Les variables mises en avant	Les flux agrégés des déplacements entre une station de départ et d'arrivée durant une période d'étude. Ici on a aucune comparaison possible durant cette période on visualise simplement l'ensemble des flux.
Données brutes ou calculées	Les données brutes sont en parties similaires au projet vélo : un tableau avec les stations de départ, les stations d'arrivée, identifiant station, nom, durée du trajet temps et mètres (à vol d'oiseau). On a également un autre jeu de données brutes avec les localisations de chaque station de vélo. Les deux jeux de données ont été fusionnés pour obtenir une matrice complète avec chaque flux individuel mais cela laisse place à des confusions lors de la visualisation. Les trajets entre deux mêmes stations ont donc été associés à un identifiant commun puis agrégés. Les données visualisées ne sont donc pas brutes. L'ensemble de ces opérations ont été réalisées grâce au langage de programmation Python.
Date des données	Il s'agit des flux du mois de juillet 2019.
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	On a une représentation par ligne de flux agrégés pour une meilleure visibilité. Cette agrégation n'est pas représentée par une augmentation de la largeur mais par un changement de couleur. Les règles de sémiologie n'ont pas été respectées puisqu'au lieu de traduire l'ordre, on traduit ici la différence avec des couleurs changeantes pour chaque classe.
Représentations graphiques	Aucune représentation graphique possible avec cet outil.

Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Nous ne savons pas s'il est possible de considérer cela comme un client mais on utilise ici Kepler.gl qui est un outil simple de prise en main. Il permet la création d'applications web interactives de visualisation de données géolocalisées. Positif : très facile à prendre en main et donne des résultats visuels assez esthétiques et gratuits Négatif : peut être limité avec des données assez simples et pas de possibilité de requêtage sur la base de données.
Logiciels serveurs	Pas de serveur ?
Logiciels de base de données	Pas de logiciel, avec kepler.gl il suffit de lui donner un fichier .csv, json, ou geoson
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles…	Kepler.gl n'est pas vraiment un outil de WebSIG mais plutôt un outil de visualisation de données, avec notamment des données de flux. Ce type d'outil est donc assez limité, pas de possibilité de requêtages, pas de filtres, un simple outil de sélection s'offre à nous. Du point de vue de la représentation, on peut choisir une agrégation des flux pour plus de visibilité. En conclusion Kepler.gl est un outil de visualisation assez intéressant lors de la phase d'exploration des données comme ont pu nous indiquer les chercheurs. Dans ce même esprit on peut citer flowmap.blue ou encore Arabesque qui sont dans un esprit similaire.

	Création d'une application en WebSIG pour étudier la congestion routière à Grenoble file:///C:/Users/benjamin/AppData/Local/Temp/VALLET-Baptiste_2019_RapportHA.pdf
Présentation générale de la carte	
Fond de carte	Fond de carte classique OpenStreetMap
Légende	La légende apparaît dans un petit onglet : une progression du jaune vers le rouge pour les plus fortes valeurs, la légende manque peut-être d'indications sur les valeurs représentées.
Fonctionnalités	– Un bouton Zoom/ Dézoom – Une fenêtre classique avec des filtres qui permettent des sélections temporelles : heures, jours, vacances, mois, années → et un bouton pour l'actualisation de ces filtres. – Une fois les filtres appliqués et une représentation visuelle affichée, on peut cliquer sur une commune : on obtient le temps moyen qu'il faut pour aller de cette commune à la commune de Grenoble. – Enfin fonctionnalité très intéressante, on peut switcher sur quatre onglets : le premier c'est la représentation classique des variables de temps de parcours ainsi que leur écart de temps. Un deuxième onglet se spécialise dans l'évolution des temps de parcours. Un troisième permet la visualisation des données toujours grâce au même filtre mais par des courbes et des diagrammes de distributions.
Analyse des données	
Les variables mises en avant	Dans ce travail on essaye de représenter la congestion routière dans l'agglomération de Grenoble. Pour cela on utilise les Floating Car Data (FCD) avec des informations sur : la vitesse des voitures, le temps de parcours d'un trajet, la localisation des voitures. Ces données permettent d'étudier les temps de parcours sur divers itinéraires, les zones de congestions et leurs fréquences mais elles ne permettent pas de modéliser des flux, d'évaluer des charges de trafic et d'expliquer les phénomènes de ralentissement.
Données brutes ou calculées	Les données représentées sont toutes calculées : des moyennes de temps de parcours, des écarts par rapport à un écoulement libre en valeur relative et absolue, des évolutions de temps de parcours en valeur absolue et relative.
Date des données	2018-2019, les données depuis ont sans doute été mises à jour par l'agence d'urbanisme.
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Les données représentées ne sont pas de flux.
Représentation graphiques	Oui des représentations graphiques : des courbes et des diagrammes de distributions dans un onglet dédié. Ces derniers sont sélectionnables et paramétrables à l'aide de la même fenêtre de filtre que pour la visualisation cartographique. Dans cet exemple, on représente pour une date donnée les écarts entre le temps de référence et réel en valeur absolue (minutes) et relative (pourcentages). On retrouve enfin des courbes qui représentent les évolutions des écarts absolus et relatifs moyens entre le temps de référence et réel.
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	

Logiciels clients	Le client est ici l'application Rshiny. Utilise des fonctions comme Leaflet (librairie JavaScript)
Logiciels serveurs	Pas de serveur cartographique. Mais serveur sous R. Il faut savoir coder sous RStudio. L'environnement est entièrement gratuit et flexible et permet la création facile de diverses représentations statistiques.
Logiciels de base de données	PostgreSQL. Dans cet exemple, c'est un peu plus complexe. C'est un script R qui va faire des requêtes API dans un fichier .csv des données de TomTom, qui va ensuite leur faire subir un certain nombre de traitements pour aboutir aux variables qui serviront ensuite à la représentation visuelle. Ces données sont alors intégrées dans une base de données PostgreSQL
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Ce WebSIG n'effectue pas une représentation de type, données de flux. Toutefois il détient plusieurs points forts qui, nous pensons, peuvent être intéressants pour la création du futur WebSIG. D'abord car celui-ci comme la plupart des WebSIG fonctionne à l'aide d'une fenêtre filtre assez poussée avec ici le paramètre temporel. Les données exprimées peuvent être consultées à un pas de temps totalement modifiable par l'utilisateur, puisqu'il a le choix de l'année, le mois, le jour (avec dans ces deux cas une sélection multiple possible) et de l'horaire (les données ayant été agrégées au minimum toutes 30 minutes). La sélection temporelle est donc assez puissante et pourrait inspirer le futur WebSIG. La force de l'application réside également dans sa capacité à générer divers graphiques (courbes, diagrammes de distributions) répondant également aux requêtes temporelles. L'outil permettant ce type de production est Rshiny développé sous R, de plus étant issus d'un rapport de stage la méthode de production est décrite ce qui est un plus pour l'inspiration lors du futur développement. En conclusion, bien que ne représentant pas des données de flux ce WebSIG peut très largement être inspirant dans sa capacité à exprimer des requêtes temporelles, par sa génération de graphiques et par l'utilisation d'outils comme Rshiny dont la méthode est détaillée.

	Mobiliscope (Lyon et Toulouse) https://mobiliscope.cnrs.fr/fr/geoviz/idf
Présentation générale de la carte	
Fond de carte	Fond OpenStreetMap
Légende	Affichage sur la carte en bas à gauche
Fonctionnalités	– Détails à côté de la réglette de temps sur l'utilisation du site. – Possibilité de lancer "l'animation" pour voir les changements au fil des heures. – Zoom et dézoom – Possibilité de télécharger les données. – Changement du fond de cartes avec la possibilité de voir les quartiers prioritaires (cohérent pour l'analyse de la population). – Menu avec "partie" méthodes, un « à propos » et le fonctionnement de l'outil – Possibilité d'affichage des variables selon trois modes : flux, nombre et pourcentage.
Analyse des données	
Les variables mises en avant	La partie du WebSIG sur les transports est axée sur l'utilisation de trois types : les mobilités douces (le problème ici c'est que l'on peut supposer qu'ils ont rassemblé la marche à pied et le vélo), véhicules motorisés privés et transports en communs.
Données brutes ou calculées	Données calculées. Ils ont très certainement fait la somme de plusieurs modes de transport pour chaque catégorie. Le problème étant que le traitement des données qui a été fait n'est pas expliqué. Lors du téléchargement des données (pour Toulouse), il y a seulement la feuille avec les résultats finaux. Pas d'explications des calculs utilisés pour le traitement des données issus de l'Enquête Ménages déplacements.
Date des données	Toulouse : 2013, puisqu'ils traitent les données issues de l'Enquête Ménages déplacements, donc assez vieux quand même et pas actualisées depuis Lyon : les données datent de 2015.
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Possibilités d'obtenir un résultat avec des flux. Deux secteurs sont reliés entre eux, celui qui est sélectionné correspond au nombre de personnes qui y sont présentes, relié à leur lieu de résidence. La donnée est superposée avec l'utilisation des mobilités douces, transports en communs ou véhicules motorisés privés → exprime le nombre de personnes et le mode de transport qu'ils ont utilisés pour se rendre dans un secteur hors de leur lieu de résidence .
Représentations graphiques	Une des forces du Mobiliscope ce sont ses représentations graphiques, avec la proposition d'un graphique avec les trois modes représentés de manière "empilé" ou un seul mode. En abscisse on a les heures avec un pas de 2 et ont ordonné le nombre ou la proportion de personnes ayant utilisé une des trois catégories de transport → c'est intéressant puisque c'est une manière de faire une synthèse sur les 24h, en ayant une vision plus large.

Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Leaflet/ bibliothèque graphique D3.js/ mesure de statistiques web: Matomo alternative à google analytics et CARTO. "Des scripts R permettent de traiter les données et de produire des fichiers CSv et GeoJSON, qui sont ensuite utilisés pour l'application avec une application cliente JavaScript qui utilise la bibliothèque D3JS pour produire les graphes et les cartes."
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	C'est un WebSIG qui n'est pas centré sur les transports mais sur l'étude de la population. Points positifs: réglette de temps au centre de la page avec possibilité d'avoir une animation (évolution sur la journée, observation sur 24h)/ Synthèse par secteur des modes de transport utilisés avec les graphiques/ possibilité de choisir de quelle manière les données sont représentées. Points négatifs : la comparaison entre les différentes catégories des modes de transport n'est possible qu'au travers du graphique. De plus, leur représentation par flux ne permet pas une visualisation rapide, la lecture est plutôt complexe, sachant que cette représentation n'est pas faite pour chaque secteur.

	Bike Share Map (Toulouse et Lyon) https://bikesharemap.com/lyon/#/12.235385362277597/4.8671/45.7873/
Présentation générale de la carte	
Fond de carte	Fond OpenStreetMap
Légende	Situé en haut à droite, sous forme d'une réglette (station vide à remplie) en dégradé de couleurs avec de part et d'autre deux rectangles avec les bordures plus foncées. Il manque des informations, certains cercles sur la carte n'ont pas de légende qui correspond. Cercles proportionnels mais pas de légende. Possibilité de changer le dégradé de couleurs.
Fonctionnalités	Cadran à gauche: – Onglet information avec le nom des villes avec un système similaire, la météo et la source sur les données. – Onglet nombre avec nombre de vélos disponibles au total, stations et analyse du territoire (densité…) – Onglet « graphiques » – Onglet "news" mais il n'y a rien – Menu déroulant pour changer de ville – Vidéo pour avoir sur les derniers 24h accessible l'évolution du parc de vélo – Blog "A propos" sur la carte et son fonctionnement – Partage
Analyse des données	
Les variables mises en avant	Pour chaque station, on a le nombre de bornettes, vélos + espaces libres, le nombre de vélos et le nombre de places libres. Ils ont utilisé les données JCDecaux
Données brutes ou calculées	On peut supposer que ce sont des données brutes qui n'ont pas eu de traitement particulier.
Date des données	Sur les dernières 24h
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Pas de données de flux mais des cercles proportionnels colorés
Représentations graphiques	Représentation graphique lorsque l'on clique sur les cercles → abscisse : heures minute par minutes (sur 24h) et ordonné nombre de vélo. Une courbe sur la donnée vélo + places libres & vélo disponible par stations.
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Bibliothèque de données jQuery, JavaScript.
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Ce WebSIG est basé sur l'étude des vélos. Ici, contrairement à la majorité des autres sites, il affiche la météo. Le site propose une analyse de base plutôt axée sur la disponibilité des vélos. Comme le WebSIG précédent, il est possible grâce à une animation de voir l'évolution des données sur 24h avec également la présence d'un graphique qui permet de synthétiser la donnée. L'essentiel est présent pour l'analyse de la disponibilité des vélos en temps réel sur chaque ville. La légende à l'image du site reste sommaire, pour quelques données on se retrouve avec une absence d'information sur la représentation de ces dernières.

	Web SIG ARE Cartes de la Suisse - Confédération suisse - https://map.geo.admin.ch/?lang=fr&topic=are&bgLayer=ch.swisstopo.pixelkarte-grau&catalogNodes=954,959,965
Présentation générale de la carte	
Fond de carte	Carte topographique (intéressant à superposer avec les données transports) affichée en premier mais possibilité d'avoir un fond blanc, noir & blanc, colorés ou aérien.
Légende	Accès par l'icône informations à côté de chaque variable → "l'infobox" on a alors accès à la légende. Positif : Explication pour le calcul des variables et leurs sources Négatif : pas pratique de devoir aller chercher la légende à chaque fois.
Fonctionnalités	– Possibilité de zommer et dézommer – Vue en 3D en changeant l'orientation → (l'actualisation ou non de ces données est notifiée – Acquisition de géolocalisation (ajoute de l'interactivité pour l'appli). – Partage avec Facebook, twitter, mail, QR code. – Impression où l'on peut changer d'échelle. – Dessin et mesure. – Import des données et comparaison entre deux jeux de données → ouvert à un public plus professionnel. – Barre de recherche
Analyse des données	
Les variables mises en avant	Données brutes issus du Modèle National du Trafic Voyageurs (MNTP).
Données brutes ou calculées	– Données calculées. Charge sur le trafic ferroviaire et routier avec l'utilisation du modèle national du trafic voyageurs (MNTP) → nombre de passagers transportés sur le réseau ferroviaire et routier : valeurs quotidiennes pour le Trafic Journalier Moyens des jours Ouvrables (TJMO), Trafic Journalier Moyen (TJM), valeurs des heures de pointe du matin et soirée (chaque segment à ces trois données), pour le trafic routier on a juste de TJMO. – Qualité des dessertes des TP → "à partir des données de l'horaire électronique des entreprises suisses de transports" (zone tampon) – Temps de parcours vers les centres ou vers les agglomérations par la route ou avec les transports en communs. – Accessibilité en transports en communs ou par la route.
Date des données	Certaines données datent de 2017 pour les plus anciennes et les plus récentes de cette année (mars-juin)
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Cela ressemble à des flux mais tous relié entre eux comme une « toile d'araignée »
Représentation graphiques	Pas de représentation graphique
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	HTML et JavaScript. "Le géoportail fédéral Suisse utilise la bibliothèque JS cliente OpenLayers pour la partie carto. et le framework (ensemble de bibliothèques) Angular.JS

Logiciels serveurs Logiciels de base de données	pour construire l'interface utilisateur. Les données vectorielles sont fournies en geoJSON à OpenLayers, les données tuilées du fond de carte proviennent d'un serveur WMTS de SwissTopo (IGN Suisse) : Ce serveur doit tourner sous GeoServer."
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Les données misent en avant dans ce WebSIG sont intéressantes (temps de parcours et trafic), leur mode de représentation aussi par carte de chaleur ou par « flux liés » qui varie selon le flux de voyageurs. Cependant, la légende n'est accessible qu'en cliquant sur l'icône « informations », ce qui n'est pas très ergonomique, on ne peut la consulter d'un seul coup d'œil. La fonctionnalité qu'il est important de relever c'est le fait que le calcul et la description de la donnée soit inscrits, cela permet à l'utilisateur de mieux comprendre ce qui est exprimé. L'ajout de toutes les informations concernant la donnée lorsque l'on clique sur un segment de route est un plus également. La visualisation par vue 3D ajoute une plus-value permettant de tenir compte de la topographie de la région.

ATTENTION : Les deux prochains WebSIG d'Étienne Côme ont exactement la même description puisqu'il s'agit du même type de WebSIG.

	Une carte de flux des vélib' la nuit à Paris, Londres et New-York https://www.comeetie.fr/galerie/atNight/?london
Présentation générale de la carte	
Fond de carte	Fond de carte OpenStreetMap
Légende	La légende est fixe, il s'agit d'un dégradé de noir au bleu
Fonctionnalités	Les fonctionnalités ici sont assez simples : – Bouton zoom/ dézoom → échelle qui s'adapte en fonction des changements. – Une fenêtre avec des filtres : choix de la zone d'étude, du mode : origine du vélo, destination, origine et destination
Analyse des données	
Les variables mises en avant	La densité des vélos par leurs mouvements au niveau des stations de départ ou d'arrivée (ou de la somme des deux).
Données brutes ou calculées	La variable ici est calculée mais on a aucune indication sur son mode de calcul. Après l'aide de Laurent Jégou, il s'agit de la densité moyenne en nombre de trajets par km2 de la zone sur 10h. La méthode de calcul se fait par une grille de carreaux d'une certaine taille qui sert à l'interpolation des valeurs (lissage), et à calculer la densité
Date des données	Les données sont étalées sur 6 mois. L'application ne laisse pas apparaître d'année.
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Ici la représentation se fait par une carte de chaleur : densité moyenne d'activité via mouvement des vélos au niveau des stations
Représentations graphiques	Aucune information graphique
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	En client on a des bibliothèques de fonctions JavaScript comme Leaflet mais aussi du D3.js
Logiciels serveurs	Les dalles du fond de carte sont de simples dalles en png que le serveur envoie au client, c'est Leaflet qui vient chercher ces dalles sur le serveur. Il s'agit donc d'un serveur bête ? (Il est écrit que du R a été utilisé sur le serveur du coup ?)
Logiciels de base de données	Un simple fichier csv ?
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Il s'agit d'un des WebSIG issue des travaux d'Étienne COME. Son apport pour les chercheurs n'est ici pas dans ses fonctionnalités. En effet, l'application propose une interface plutôt simple et clairement pauvre dans sa capacité de requêtage. Malgré cela, elle propose sur la même thématique des vélos une représentation des flux assez différente et unique avec une représentation par carte de chaleur. Celle-ci représente une densité d'activité moyenne par km2 sur 10h sur tout l'espace (via les mouvements de vélos au niveau des stations). La méthode de calcul se fait via une grille de carreaux qui sert à l'interpolation des valeurs. En conclusion, ce WebSIG apporte aux chercheurs une autre méthode de représentation de flux par un calcul de densité représenté sous forme d'une carte de chaleur.

	Carte exploration de matrice départ/arrivée https://www.comeetie.fr/galerie/odn/
Présentation générale de la carte	
Fond de carte	Pas de fond de carte
Légende	La légende est fixe, il s'agit d'un dégradé de jaune au noir
Fonctionnalités	Fonctionnalités simples : – Zoom, dézoom – Une fenêtre de légende fixe – Une fenêtre de filtre : une tranche horaire parmi 6 au choix, un onglet de sélection de la représentation : le départ ou l'arrivée, le choix de la ville. On peut ajuster le rayon qui illustre la densité des flux autour des stations. Enfin on a une barre d'échelle qui s'ajuste aux données de densité. Plus elle est faible, plus la densité visible est forte. Plus on l'augmente plus la densité maximum exprimée est faible. Enfin on peut entrer une adresse ou bouger un marqueur dans l'espace pour montrer les densités autour de ceux-ci
Analyse des données	
Les variables mises en avant	Ici, c'est la densité des vélos via leurs mouvements au niveau des stations de départ ou d'arrivée
Données brutes ou calculées	La variable ici est calculée mais on a aucune indication sur son mode de calcul. Après aide de Laurent Jégou, il s'agit de la densité moyenne en nombre de trajets par km2 de la zone sur 10h. La méthode de calcul se fait par une grille de carreaux d'une certaine taille qui sert à l'interpolation des valeurs (lissage), et à calculer la densité
Date des données	Les données sont étalées sur 6 mois L'application ne laisse pas apparaître d'année.
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Ici la représentation se fait par une carte de chaleur : densité moyenne d'activité via mouvement des vélos au niveau des stations
Représentations graphiques	Pas de représentation graphique
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	En client on a des bibliothèques de fonctions javascript comme Leaflet, du D3.js
Logiciels serveurs	Les dalles du fond de carte sont de simples dalles en png que le serveur envoie au client, c'est Leaflet qui vient chercher ces dalles sur le serveur. Il s'agit donc d'un serveur bête ?
Logiciels de base de données	Un simple fichier csv ?
Synthèse du WebSIG	
Synthèse sous forme du WebSIG en question : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Il s'agit d'un des WebSIG issue des travaux d'Étienne COME. Son apport pour les chercheurs n'est ici pas dans ses fonctionnalités. En effet l'application propose une interface plutôt simple et clairement pauvre dans sa capacité de requêtage (même si ici on peut jouer sur l'échelle de la donnée, la possibilité de bouger un marqueur directement sur la carte est également intéressante). Malgré cela, elle propose sur la même thématique des vélos une représentation des flux assez différente et unique avec une représentation par carte de chaleur. Celle-ci représente une densité d'activité moyenne par km2 sur 10h sur tout l'espace (via les mouvements de vélos au niveau des stations). La méthode de calcul se fait via une grille de carreaux qui sert à l'interpolation des valeurs. Enfin cette appli propose cette fois une possibilité de requêtage temporelle avec des données regroupées sous forme de tranches horaires (et une distinction semaine/week-end). Ce type de représentation temporelle est assez simple mais ne permet pas une visualisation dans le détail ce qui peut être un problème dans le cadre d'un travail de recherche. En conclusion ce WebSIG apporte aux chercheurs une autre méthode de représentation de flux par un calcul de densité représenté sous forme d'une carte de chaleur.

	Un mois de Vélib'. https://www.comeetie.fr/galerie/velib/
Présentation générale de la carte	
Fond de carte	Fond de carte vectorielle avec réseau hydrographique, voies ferrées et espaces verts
Légende	Sur la carte en bas à gauche
Fonctionnalités	– Description pour avoir plus d'informations sur la conception de la carte – Choix pour l'utilisateur, de l'heure, d'une observation selon la semaine ou le week-end, les départs et les arrivées, et enfin quelles stations on souhaite visualiser – Cadran synthétique à gauche avec le weekend, la semaine et les heures
Analyse des données	
Les variables mises en avant	Etude par semaine et/ ou weekend, heure par heure du nombre de départs et d'arrivées des stations de vélib' à Paris. Départs par heure et déplacements par heure
Données brutes ou calculées	Données calculées, réalisées à partir d'un algorithme conçu par Etienne Côme qui permet d'avoir une vision "en survol" de l'usage des stations de vélib'. Chaque couleur correspond à un cluster par la similarité entre stations.
Date des données	2013
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Données de flux avec le nombre de déplacement par heure.
Représentations graphiques	Graphique avec le profil des stations, les départs et les retours de part et d'autre, heure par heure (une limite à 5 a sans doute été défini par le créateur) → É. Côme parle d'un "profil d'usage temporel d'une station"
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Javascript, D3
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Il s'agit ici d'un type de visualisation qui diffère des autres représentations que nous avons pu observer avec d'autres WebSIG. É. Côme tente ici une approche plus poussée, pour l'analyse des stations de vélib', par l'utilisation d'un algorithme spécifique permettant de regrouper les stations entre elles, par leur profil similaire représenté par une couleur différente. Cependant, c'est regrettable qu'il n'y ait pas de légende qui permette de faire correspondre chaque cluster en fonction de sa caractéristique (+ nocturne, + loisirs, activités). Ce WebSIG offre une nouvelle visualisation très intéressante pour l'analyse des usages de station qui parait compliqué à mettre en place pour des personnes qui ne seraient pas habituées à manier les algorithmes.

	Application Uber https://movement.uber.com/explore/paris/travel-times/query?si=9&ti=&ag=iris&dt[tpb]=ALL_DAY&dt[wd;]=1,2,3,4,5,6,7&dt[dr][sd]=2020-03-01&dt[dr][ed]=2020-03-31&cd=&sa;=&sdn=&lang=fr-FR&lat.=48.856666&lng.=2.3156249&z.=12
Présentation générale de la carte	
Fond de carte	Fond de carte très simple, on voit les labels des communes, le réseau routier principal, les cours d'eau. Quand on change d'échelle, la carte s'adapte, les labels des communes disparaissent au profit des plus grandes villes jusqu'aux pays.
Légende	La légende est claire, distincte, elle s'adapte en fonction des données demandées
Fonctionnalités	Les fonctionnalités sont nombreuses et assez originales par rapport à d'autres WebSIG. On doit d'abord sélectionner la ville parmi un très large choix. Ensuite nous pouvons représenter deux types de données : d'abord les temps de trajet en voiture. Ces derniers peuvent être exprimés soit entre deux iris ou entre deux communes. Le choix des iris est très pertinent ici car il permet de visualiser une information spatiale très précise puisque intercommunale (alors que par commune on peut imaginer qu'il s'agit du temps entre deux centroïdes ou deux mairies). L'origine et la destination peuvent être sélectionnés par deux moyens : soit en rentrant directement l'adresse, soit en déplaçant une icône au sein de la carte. Le moyen de réaliser des comparaisons ici est aussi original. On sélectionne la carte à afficher, puis on sélectionne une deuxième date/plage de date ; on obtient alors une carte représentant la différence vis-à-vis de la date de référence (légende en double progression avec valeurs positives et négatives). Celle-ci s'exprime en pourcentage → exemple : 10% supérieur ou inférieur. Quand on sélectionne un trajet entre deux points ; une fenêtre apparaît avec de nouvelles informations : le temps moyen pour ce trajet, le minimum et maximum ; des graphiques qui viennent détailler l'information avec les temps moyens par jours sur ce trajet, et à différents moments de la journées journée. Le deuxième type de données est une représentation de la densité moyenne du trafic en vélo dans Paris sous forme de carte de chaleur. On peut à ce titre faire apparaître les pistes cyclables. Les données sont disponibles pour les 5 jours de la semaine cumulés ou le weekend. Enfin on peut déclencher une animation qui montre l'évolution des densités sur 24h. Pour cela on sélectionne une plage : ex évolution des flux toutes les deux heures.
Analyse des données	
Les variables mises en avant	Le temps en voiture entre deux iris/communes et les densités de flux de vélos (comment sont-ils calculés, sont-ils simplement déduits depuis une matrice origine-destination ?)
Données brutes ou calculées	Le site se base sur des données obtenues avec les trajets des chauffeurs Uber. Ces données ont été agrégées d'abord en partie de la journée : un pic du matin, du midi, de l'après-midi, un très tôt le matin et un très tard le soir; et ce pour tous les jours de la semaine et du weekend. Ils ont ensuite été agrégés avec une moyenne pour chaque jour de la semaine. Ces données apparaissent sous forme de graphiques. Enfin une moyenne pour la semaine, et une moyenne pour le weekend. Ce sont ces dernières moyennes de temps qui sont calculées. Il est plus difficile de comprendre ce que sont les données vélos. Néanmoins il s'agit bien d'une densité donc d'une variable calculée
Date des données	Cela dépend des villes, pour Paris c'est du 1 janvier 2016 au 31 mars 2020
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Ici représenté comme du qualitatif d'intensité. Pour les vélos c'est une carte de flux qui a été choisie.

Représentations graphiques	Oui, ils apparaissent quand on sélectionne un point de départ et d'arrivée pour le temps de trajet. Ils viennent détailler la donnée avec les moyennes par jours et détaillées dans les journées (toutes en même temps par contre) (le pic du matin, de l'après-midi).
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients/ serveurs/ bases de données	"Difficile à comprendre, mais comme c'est Uber, j'imagine qu'ils ont leurs propres données. L'affichage est réalisé côté client par une bibliothèque qui ressemble à Mapbox.gl.js... Le fond de carte tuilé vient de Mapbox. Les données sont transférées en GéoJSON via une API de requêtage. Pas de piste pour en savoir plus sur le stockage des données côté serveur"
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Le WebSIG de Uber peut avoir des apports sur plusieurs points. Au niveau des temps de trajets de voiture par exemple, l'idée de représentation surfacique par Iris via un dégradé de couleurs peut, pourquoi pas, inspirer une représentation de la densité du trafic routier pour le futur projet. De plus l'expression de la comparaison graphique est assez intéressante. Effectivement lorsque l'on choisit deux plages temporelles, l'application prend la première comme référence et, est capable de réaliser des calculs pour produire une carte en double progression avec une diminution ou une augmentation exprimée en pourcentage. Également comme on l'a vu l'application propose un filtre temporel via un calendrier, des boutons à cocher pour les jours (un bouton pour le week-end et un pour la semaine). Pour les horaires, il est proposé une agrégation sur des tranches horaires (7h-10h,10-16h,16-19h,19-00h00,00-7h). L'application propose aussi des représentations graphiques toujours temporelle intéressantes. Enfin elle met également à disposition une carte de chaleur sur les mobilités douces par expression ici, comme pour les travaux d'É. COME d'une densité de trafic. La sélection temporelle est assez faible néanmoins ici on a une autre fonctionnalité intéressante, c'est la possibilité de lancer une animation sur 24h à partir d'une plage temporelle sélectionnée (ex 2h). En conclusion le WebSIG peut avoir plusieurs apports pour les chercheurs : une bonne ergonomie, une sélection temporelle poussée, une représentation des comparaisons temporelles intéressantes par calculs, et production d'une carte d'évolution; des représentations graphiques; une carte de chaleur sur les densités des mobilités douces avec un outil de visualisation par animation attrayante.

	Application flux de d'avions en direct https://www.flightradar24.com/DAH1001/29bf1f93
Présentation générale de la carte	
Fond de carte	Fond de carte type OpenStreetMap, toponymes classiques qui s'affichent en fonction de l'échelle ; le fond de carte est modifiable, tout comme sa transparence
Légende	Pas de légende
Fonctionnalités	Plusieurs options : – On peut revoir des trajets passés en sélectionnant une date. On peut sélectionner un avion en particulier en tapant son code ou en cliquant dessus (quand on sélectionne un avion, on a plusieurs données et statistiques qui s'affichent à son sujet : destination, identification, altitude, vitesse…). On peut aussi avoir une possibilité de vue de 3D de l'avion et de son environnement (ce modèle 3D est créé à partir d'images à très haute résolution spatiale et d'un MNT (modèle numérique de terrain). On peut aussi faire apparaître son trajet sous forme d'une ligne – On peut également superposer une couche avec des données météorologiques (allant de simples pictogrammes à des cartes détaillées sur les précipitations) – On peut choisir d'afficher les labels des avions, choisir la taille des icônes des avions. – Une fenêtre donne des grandes statistiques comme le nombre d'avions qui circulent actuellement dans notre viewer mais aussi au global : la source de ces données est d'ailleurs précisée (satellite, radar…). On peut cliquer sur un bouton qui nous permet d'accéder à des graphiques comparants le nombres de vols sur 3 courbes 2019/2020/2021 – Une fenêtre indique les derniers tweets blog au sujet des avions. On a les informations sur les avions etc. – On peut interroger la base de données sur les vols, les aéroports etc… ici ça reste purement statistique
Analyse des données	
Les variables mises en avant	Le nombre d'avions en circulation à l'instant présent
Données brutes ou calculées	Données brutes
Date des données	Continuellement actualisées
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Un pictogramme d'avion qui bouge en direct par rapport à la réalité de ce même avion
Représentations graphiques	Oui des courbes comparants le nombres de vols sur 3 courbes 2019/2020/2021
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	"C'est assez mystérieux… Le site est développé avec le FW AngularJS, la carte est réalisée avec l'API GoogleMaps. Les données des avions viennent de leurs propres serveurs au format GéoJSON"
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	

Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Une application qui reste peut-être assez limitée finalement pour le projet. En effet le requêtage est assez faible, on est plutôt sur de la visualisation. Elle connait quand même des fonctionnalités intéressantes comme la possibilité de vue en 3D, calculée à partir d'un Modèle Numérique de Terrain. Pour autant ce n'est pas ce genre de fonctionnalité qui sera utile pour le futur projet. Un point commun peut-être les données météorologiques. Ici elles sont représentées par pictogramme avec les températures. Du point de vue de la représentation graphique, on a tout de même des graphes qui présentent sur 3 courbes 2019/2020/2021 le nombre de vol. Cela permet d'étudier l'effet de la crise du covid. Pour autant cette représentation est peu poussée. En conclusion les apports de ce WebSIG, pour le futur projet sont assez faibles, bien trop dans la simple visualisation; il ne permet pas le requêtage de données, seul quelques points comme les données météos, ou encore la représentation graphique peuvent être retenues même si cela est limité.

	A. WAJIDI (rapport de stage) "**Développement d'une application de cartographie interactive associant PostgreSQL/Geoserver et OpenLayers pour la valorisation de la donnée de la commune de Meung Sur Loire**" https://www.researchgate.net/profile/Wajdi-Abdallah/publication/344379525_valorisation_de_la_donnee_de_la_commune_de_Meung_Sur_Loire_Developpement_d%27une_application_de_cartographie_interactive_associant_PostgresGeoserver_et_OpenLayers/links/5f6e1ad7458515b7cf4d01d4/valorisation-de-la-donnee-de-la-commune-de-Meung-Sur-Loire-Developpement-dune-application-de-cartographie-interactive-associant-Postgres-Geoserver-et-OpenLayers.pdf
Présentation générale de la carte	
Fond de carte	Fond de carte OpenStreetMap
Légende	Une légende qui s'affiche suivant la couche affichée
Fonctionnalités	Des outils encore une fois assez simple : – Zoom/dézoom – Calcul de distances et de surfaces sur la carte – Un outil de sélection d'objet pour faire apparaître des informations sur celui-ci / un outil de désélection – Une fenêtre pour sélectionner/désélectionner des couches – Enfin une fenêtre pour afficher les tables attributaires des couches / on peut aussi effectuer une recherche par attribut avec des opérateurs simples
Analyse des données	
Les variables mises en avant	Limites communales, les différents types de routes, les lignes de chemins de fer…
Données brutes ou calculées	Il ne s'agit pas d'une cartographie statistique
Date des données	Pas de date
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Pas de données de flux
Représentations graphiques	Pas de représentations graphiques
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Côté client on a des bibliothèques Javascript : GeoExt (apporte l'interface cartographique : légende, cocher/décocher une couche), OpenLayers (affiche le fond cartographique sous forme de tuiles, et les couches avec divers types de formats (shp, GeoJson, KML, GML, Post GIS), ExtJS (apporte des composants visuels : des panneaux, des fenêtres, des panels, onglets etc…)
Logiciels serveurs	Un serveur internet Apache (donne accès à internet au serveur cartographique) et un serveur cartographique Geoserver (Il permet également la visualisation de données géographiques ; l'utilisation de fonctionnalités d'analyses spatiales simples, l'exécution de requêtes et la consultation des données attributaires sur les différentes couches d'informations.)
Logiciels de base de données	PostgreSQL et PostGIS
Synthèse du WebSIG	

Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Ce WebSIG a été décrit car il présente ce qu'est ce genre d'application avec des fonctionnalités simples. Le point fort est qu'il est issu d'un rapport de stage, on a donc possibilité d'avoir le détail sur le fonctionnement technique de l'application. N'étant pas des données de flux, et l'application décrite n'utilisant pas du requêtage temporel, il faudrait s'assurer que ce type d'outils permettent de produire ce type de carte. En conclusion cette application est un exemple de production avec l'outil Géoserveur et le client OpenLayers associé à d'autres bibliothèques javascript. Il faudra toutefois approfondir sur ce type d'outil afin de connaître leur possible utilité, ou non, dans le futur projet.

	Flux de circulation sur les autoroutes à Taïwan http://webgis.geo.ntnu.edu.tw/ETC/
Présentation générale de la carte	
Fond de carte	Fond de carte OpenStreetMap. Le choix d'un thème sombre rend le graphisme particulièrement agréable.
Légende	La légende est fixe et représente un dégradé de rouge pour les valeurs fortes, et vert pour les valeurs faibles
Fonctionnalités	Les fonctionnalités ne sont pas très nombreuses : – Zoom par la molette – Une fenêtre avec un certain nombre de filtres : choix de la date et de la plage horaire à l'intérieur de celle-ci. Le choix du sens de direction sur l'autoroute, le choix du type de véhicules. Ces choix se font par cochage simple d'un bouton, menu déroulant, ou calendrier pour la date. Après avoir sélectionné notre date et plage horaire ; un graphique apparaît
Analyse des données	
Les variables mises en avant	Le nombre de voitures comptabilisées à une station donnée sur l'autoroute
Données brutes ou calculées	C'est une variable quantitative de stock, donc une donnée plutôt brute. C'est le nombre de voitures comptabilisées à une station, à un moment donné. Pourtant s'il s'agit bien d'une donnée de stock, le choix de la représentation pose question. En effet la donnée est représentée ici sous forme d'un dégradé comme s'il s'agissait d'une variable quantitative d'intensité.
Date des données	du 1 janvier 2017 au 31 décembre 2019
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Les flux sont représentés ici par les points qui correspondent aux stations d'identification du nombre de véhicules. Le problème ici, est qu'une variation de la couleur a été utilisé, ce qui est mal venu pour une variable quantitative de stock
Représentations graphiques	Oui, un graphique représente le nombre de véhicules en ordonnée et l'heure en abscisse
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Mapbox (une bibliothèque javascript basée sur Leaflet), des bibliothèques d'outils comme Bootstrap (boutons, outils de navigation), et autres bibliothèques JavaScript. "La carte utilise les bibliothèques D3JS, Mapbox_gl.js et deck_gl.js (Uber) pour fonctionner. Le fond tuilé est de MapBox. Les données sont au format GeoJSON et viennent de leurs propres serveurs. Elles proviennent de comptages fournis par l'autorité des transports (bureau des autoroutes du ministère des communications)"
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	

Synthèse sous forme du WebSIG en question : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Cette application présente des données de flux avec requêtage temporel. Du point de vue du mode de représentation, on est ici sur des points de comptage à intervalle régulier. La donnée brute est donc une variable quantitative de stock. Pourtant en faisant varier la valeur des points, on ne respecte pas la sémiologie graphique. Le requêtage temporel se fait lui, par jour grâce à une sélection sur un calendrier. On peut ensuite sélectionner un horaire (les données ayant été agrégées par heure). Un graphique sur la journée vient compléter l'information et proposer un complément de visualisation. En conclusion ce WebSIG propose une méthode de représentation des flux par points, et non bipoint (pour autant ce type de représentation ne sera pas reproductible sur les vélos). Les sélections temporelles sont simplifiées. C'est donc un outil très facile à prendre en main, mais qui peut être assez limité comme réel support pour de la recherche.

	Flux d'échanges des ressources dans le monde https://resourcetrade.earth/?year=2019&units=value&autozoom=1
Présentation générale de la carte	
Fond de carte	Fond de carte plus que simple. On a simplement la limite des continents dans un dégradé de gris ; aucune autre couleur n'est présente ; seuls les toponymes des pays concernés par les flux apparaissent
Légende	Oui elle est présente et fixe
Fonctionnalités	Fonctionnalités nombreuses et classiques : – Pas de zoom à la molette, ni par un bouton, mais un auto zoom vers les régions si les données se concentrent dans un seul endroit du monde (option décochable) – Un ensemble de menus déroulants qui permettent de filtrer les données : Un premier permet de sélectionner pour l'exportateur de ressource, un pays/une région ou un groupe). Un deuxième permet exactement de la même manière, de sélectionner une entité importatrice. Un troisième permet de sélectionner le type de marchandises échangées, avec le choix de toutes les additionner. Enfin un dernier menu déroulant permet de choisir l'année. On peut aussi choisir entre afficher la valeur ou le poids. – On a aussi trois boutons en bas à droite. Le premier est un bouton d'information qui explique rapidement l'enjeu en nous redirigeant vers une autre page du site internet. Un deuxième bouton permet de partager la page du WebSIG sur divers réseaux sociaux. Enfin un dernier bouton permet de télécharger les données au format. Xlxs. – En bas on peut voir un graphique et un tableau. Ces derniers apportent une data visualisation intéressante en classant les valeurs des flux ; les valeurs "exportateurs", les valeurs "importations"; ainsi qu'un classement des évolutions les plus positives, mais aussi les plus négatives.
Analyse des données	
Les variables mises en avant	Les flux des échanges de ressources sont en valeurs (en dollar) et en poids (en tonnes)
Données brutes ou calculées	Ici il s'agit de variables quantitatives de stock
Date des données	De l'année 2000 à l'année 2019
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Il s'agit d'une ligne proportionnelle aux quantités décrites
Représentations graphiques	Oui le graphique représente l'évolution de la valeur monétaire pour toutes les marchandises décrites de 2000 à 2019
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients Logiciels serveurs Logiciels de base de données	"Ce site utilise du code JS développé par la structure pour afficher les cartes à partir des données de leur propre API. La source initiale des données est la base ComTrade de l'ONU. L'interface utilise la bibliothèque preact.js. Je pense que les cartes en SVG sont précalculées, pas générées à la demande."

Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Un WebSIG détenant une bonne ergonomie. Pas de sélection temporelle ; mais une présence d'expression de données de flux. Le type de représentation est bipoint avec des données quantitatives de stock. On fait donc varier la taille de la flèche en fonction des quantités exprimées. L'auteur a pris le parti de réaliser une classification en trois classes pour exprimer ces flux. Même si ce choix peut être critiqué, notamment du point de vue de la discrétisation, la visualisation elle, est assez claire pour ce type de représentation. Enfin le deuxième point fort est la présence du graphique; ainsi que celle du tableau avec des indicateurs d'évolutions bien mis en forme. En conclusion ce WebSIG est intéressant par sa bonne ergonomie, un bel esthétique, et une visualisation efficace des flux. Les compléments avec le graphique et le tableau d'indicateurs sont également bien réalisés. (remarque: quand on clique sur un flux trop à droite de l'écran, on ne voit pas l'ensemble de la case)

	Représentation du risque lié à la covid19 lors des déplacements entre région en Espagne https://flowmaps.life.bsc.es/flowboard/board_mobility/2021-05-08
Présentation générale de la carte	
Fond de carte	Fond de carte type OpenStreetMap
Légende	Oui elle prend la forme d'une barre avec un dégradé de couleur
Fonctionnalités	Peu de fonctionnalités : – Zoom par molette – Passer la souris sur les entités géographiques pour voir les données exactes – Clique sur la région pour la sélectionner – Le seul filtre est celui de la date ; ici on peut seulement sélectionner un jour sur un calendrier – Des représentations graphiques
Analyse des données	
Les variables mises en avant	Ici la variable cartographiée est le nombre de voyages entre deux régions. On a d'un côté, les voyages depuis la région sélectionnée ; et de l'autre, les voyages vers celle-ci.
Données brutes ou calculées	Il s'agit d'une variable quantitative de stock.
Date des données	Février 2020-Mai 2021
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	A la vue de la donnée que l'on possède, on constate que le choix de représentation est assez curieux avec un non-respect de la sémiologie graphique. En effet les directions des flux sont représentées par des flèches ; mais c'est la valeur sur les entités surfaciques (les régions) qui viennent exprimer la proportion.
Représentations graphiques	Oui, il y a des graphiques qui apportent une analyse plus poussée que sur une seule journée ; avec la possibilité d'afficher des mobilités sur une moyenne de 7 ou de 14 jours (ces sont des moyennes glissantes tout simplement). On vient comparer les mobilités de "sortantes" et "rentrantes" avec les mobilités internationales. On s'intéresse aussi au nombre de flux par personne avec une classification un, deux, et plus de trois
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Des bibliothèques JavaScript : D3, jQuery. Du Bootstrap. "La carte utilise l'API GoogleMaps. Le fond de carte tuilé vient de chez Carto. Les graphes sont dessinés avec la bibliothèque JS Plotly. Les données proviennent d'un serveur de flux de données JSON qui envoie les infos. selon les origines/destinations des trafics."
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Il s'agit d'un WebSIG qui aborde des données de flux avec un filtre temporel. Le type de représentation est assez spéciale puisqu'il n'y a pas de respect de la sémiologie graphique. Le WebSIG propose également deux fenêtres avec deux cartes ce qui est un point commun nous pensons avec le futur projet. Il y a aussi des graphiques qui présente un intérêt certain en termes de statistiques. Néanmoins le requêtage est très faible et l'ergonomie laisse à désirer. En conclusion il s'agit d'un WebSIG peu poussé avec des requêtes temporelles simple ; une double carte intéressante pour notre futur WebSIG ; une proposition de représentation de flux particulière ; et pour finir des représentations graphiques complétant la visualisation.

	Etude sur les ressources, la géographie physique et l'occupation du sol dans l'Asie du Sud-Est http://resources-info.appspot.com/
Présentation générale de la carte	
Fond de carte	Un fond de carte plan Google et un fond satellite par la société TerraMetrics
Légende	Oui il faut cliquer sur l'onglet. Elle est fixe et regroupe l'ensemble des couches sélectionnables
Fonctionnalités	Le WebSIG se divise en 6 grandes parties : – Le viewer, au centre, avec la possibilité d'afficher et de changer le fond cartographique – Toujours au centre mais en bas ; de la visualisation de données par des graphiques avec des diagrammes ombrothermiques, un NVDI sur plusieurs dates etc. Ces informations se mettent à jour en fonction de la zone qui est sélectionnée – En haut à gauche on retrouve les outils de sélection avec la possibilité de sélectionner un bassin versant, un paysage, ou même dessiner soit même un polygone. Un bouton "compute" permet de mettre à jour les indicateurs vis à vis de l'entité fraîchement sélectionnée – En bas à gauche on a accès aux couches. On peut les cocher pour les faire apparaître à la manière d'un logiciel SIG classique. Deux couches : les glissements de terrains, la perte de forêt sont associés à une échelle temporelle avec curseur pour changer l'année sur la période 2001-2018 – En haut à droite on a des indicateurs : population, surface, topographie. Ces indicateurs sont mis à jour à chaque sélection d'une nouvelle entité – En bas à droite deux autres indicateurs : un indicateur sur la faune et l'autre sur la flore qui se mettent également à jour
Analyse des données	
Les variables mises en avant	Pas de variables statistiques cartographiées, seulement au niveau des graphiques
Données brutes ou calculées	Pas de variables statistiques cartographiées
Date des données	Dates varient selon les données
Visualisation des données	
Type de représentation – (Respect de la sémiologie ?)	On affiche simplement des couches: occupation du sol ainsi que topographie. Une représentation classique pour ce genre de couche; une différence pour la première, et un dégradé du vert-marron-blanc pour la seconde
Représentations graphiques	Il y a sept représentations graphiques, avec une représentation qui diffère vraiment de la visualisation sur le viewer. Ces représentations se mettent à jour en fonction de l'entité sélectionnée. On retrouve notamment : un diagramme ombrothermique, l'occupation du sol en hectare, la diminution de la forêt par année, le nombre de feu de forêt par an, la valeur de NVDI par année, la pluviométrie par année, les glissements de terrains
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Bootstrap, bibliothèque javascript (jQuery). "Pas mal d'infos en cliquant sur le bouton « I » dans le coin en haut à droite. La carte est du Google Maps, les graphiques utilisent HighCharts.js"
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	

Synthèse sous forme du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Ce WebSIG aborde une bonne ergonomie. Ici nous n'avons pas réellement de filtres temporels, ni de donnée de flux ; néanmoins la capacité de mise à jour des indicateurs et des graphiques est très intéressante pour le futur projet.

	WebSIG de gestion de couches sur l'Indonésie https://en.prims.brg.go.id/map?q=eyJsYXIlcnMiOltdLCJtYXBQcm9wZXJ0aWVzIjp7ImNlbnRlciI6WzExOC4yMjA1NjE5OTQ2MTIwMSw1Ljg3MDMzMzAzMDM2NzczOV0sInpvb20iOjV9fQ
Présentation générale de la carte	
Fond de carte	Trois fonds de cartes : un fond classique avec topographie et contour des continents, un fond satellite, un fond RBI
Légende	Non pas de légende en forme, mais une représentation dans l'onglet de la couche (exemple : on ouvre la couche tourbière : on un rectangle vert dans la couche, se comporte comme un outil SIG classique)
Fonctionnalités	Le WebSIG se rapproche beaucoup d'un logiciel SIG classique : – Des outils de mesures : dessin de lignes pour les distances, des polygones pour les surfaces, des points pour des coordonnées géographiques – Le seul WebSIG que nous avons trouvé qui propose une échelle capable de se calibrer avec le zoom-dézoom de la souris – Un bouton "aide" permettant d'ouvrir une interface d'explication sur l'utilisation de l'application – Un outil pour trouver une localisation à l'aide d'un menu déroulant. En cliquant dessus cela affiche l'entité géographique en question et zoom dessus – Plusieurs onglets thématisés (limites administratives, occupation du sol, hydrologie, tourbière, restauration) contenant des couches que l'on peut afficher dans le viewer – Les couches sélectionnées apparaissent en bas à gauche : on peut interchanger leur ordre, les masquer, modifier leur opacité, ou encore avoir des informations sur celles-ci.
Analyse des données	
Les variables mises en avant	Pas de variables statistiques cartographiées
Données brutes ou calculées	Pas de variables statistiques cartographiées
Date des données	Pas de date
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	On affiche des couches une par une. Les types d'implantations sont correctes
Représentations graphiques	Pas de représentations graphiques
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Des bibliothèques JavaScript ? OpenLayers ? "Les données proviennent de flux WxS d'un GeoServer (https://geoserver.prims.brg.go.id), sauf le fond de carte qui est d'ESRI ArcGIS Online."
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	

Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles…	Cette application ressemble beaucoup à un logiciel SIG classique comme QGIS ou ArcGIS. Elle offre ainsi une gestion des couches intéressantes et très ergonomique. Ce genre de fonction peut être intéressante si dans le futur projet on peut, mettre en place une gestion de plusieurs couches (les flux vélos, les densités des véhicules, le nombre d'entrée dans les transports en commun par exemple). D'autres outils et fonctions comme les outils de mesure et la gestion de l'échelle sont des éléments également intéressants à retenir. Enfin l'application utilise l'outil côté serveur "Geoserver", et probablement OpenLayers côté client ; le résultat est concluant et montre que ces outils sont efficaces pour parvenir à ce genre de résultat. En conclusion cette application ergonomique propose par ses outils une gestion intelligente des couches ainsi que des fonctions (mesure, échelle) qui pourraient être utiles aux chercheurs pour le futur projet.

	WebSIG sur les “empreintes urbaines” https://app.urbanfootprint.com/
Présentation générale de la carte	
Fond de carte	Très large choix de de fonds de cartes OpenStreetMap, minimal land light/dark, basic, bright, streets, sattelite, terrain, shematic.
Légende	Oui une légende dans la couche lorsqu'on la sélectionne
Fonctionnalités	Il s’agit d’un WebSIG encore une fois qui se rapproche d’un logiciel SIG visuellement mais aussi dans sa capacité à générer une cartographie. – Echelle qui se met à jour avec le zoom/dézoom – D’autres outils de zoom : zoom sur couche/zoom sur sélection – Divers outils de sélection : clique, polygone, boxe – On peut créer un projet : avec cette fenêtre on le nomme, on choisit une zone géographique (forcément aux Etats-Unis (une ville, un état et autres limites administratives) Sur ce projet on peut ensuite ajouter des couches, les possibilités sont considérables avec quatorze thèmes liés à l’urbanisme, l’écologie, l’économie, l’enseignement etc. On retrouve au total 262 couches ! Une fois sélectionnées ces couches sont retenues dans une fenêtre, on peut alors avoir de nouvelles interactions avec chacune d’elle. On peut alors modifier la symbologie, l’opacité, la thématique (exemple : dans la couche “occupation du sol” on a plusieurs sous couches. On a ensuite des outils de sélection dans la couche, de suppression et d’extraction. On peut créer un buffer toujours pour sélectionner, enfin on la source des données de la couche. – On peut ajouter ses propres couches au format GeoJson, GeoPackage et Shapefile – On a accès à des indicateurs statistiques sur le territoire – On a accès aux tableaux de données – On peut exporter nos couches au format shapefile, GeoJson, GeoPackage, CSV. – On peut exporter notre carte avec une mise en page semi-automatique
Analyse des données	
Les variables mises en avant	Pas de variables statistiques cartographique
Données brutes ou calculées	Pas de variables statistiques cartographique
Date des données	Pas de date
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	On affiche des couches une par une. Les types d'implantations sont correctes
Représentations graphiques	Des représentations graphiques mais qui semblent avoir du mal à fonctionner
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Un API de l'appli construit sur mesure ? Il contient du JavaScript, Json, HTML
Logiciels serveurs	Ne sait pas
Logiciels de base de données	Ne sait pas

Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Après décomposition de l'application on est sur un outil de cartographie assez puissant en ligne. Il propose une personnalisation des couches très intéressantes et des fonctionnalités poussées (créer un buffer, créer des sélections, créer de nouvelles couches etc..). Ces fonctions ne sont pas forcément inintéressantes mais dans le cadre du projet des chercheurs elles semblent difficiles à mettre en place pour une utilité relative. En conclusion ce WebSIG est assez intéressant, très complet et très puissant dans la cartographie de problématiques autour de l'espace urbain. Il peut être un bon outil pour les décideurs publics et les individus avisés aux outils de SIG classiques. Néanmoins ces apports dans le projet des chercheurs semblent, en dehors de quelques fonctionnalités, peu convaincant.

	WebSIG sur les transports en communs de Grenoble TAG Grenoble network simulator (vasile.ch)
Présentation générale de la carte	
Fond de carte	GoogleMaps
Légende	Pas de légende
Fonctionnalités	Peu de fonctionnalités: partage sur twitter et Facebook, accélérer le temps, entrer sa localisation, zoom et dézoom, dessin de l'itinéraire, suivi d'une ligne en temps réel par vue satellitaire et affichage des arrêts ainsi que leurs heures d'arrivées
Analyse des données	
Les variables mises en avant	Affichage des arrêts, chaque point qui correspond à une ligne se déplace sur la carte en temps réel
Données brutes ou calculées	Données brutes: destination de la ligne, heure d'arrivée, vitesse en km/h. Données possiblement issues du service de gestion des transports de Grenoble
Date des données	Données en temps réel
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Points correspondant aux lignes
Représentations graphiques	Pas de graphique
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients Logiciels serveurs Logiciels de base de données	Javascript et bibliothèque jQuery. "Ce site est une démo d'un projet de développement JS pour la visualisation de données GTFS (affichage de véhicules sur des parcours, d'après les horaires prévus à l'avance). Le code est ici : https://github.com/vasile/transit-map Carte avec l'API GoogleMaps. Les données sont dans des fichiers plats au format GTFS (horaires de lignes de transports)."
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	WebSIG qui offre peu de fonctionnalités mais cela se comprend puisqu'ici il n'est pas question d'une analyse de données, mais plus de visualisation de l'activité du réseau de transport de la ville. On retrouve une similarité avec le Bike Share Map, où le principe du WebSIG réside dans le suivi des différentes lignes en service sur le réseau de transport de Grenoble. Visualisation de l'activité en temps réel sur le réseau.

	Evolution du COVID en France https://demo.lizmap.com/lizmap/index.php/view/map/?repository=demo&project=covid_19_fr
Présentation générale de la carte	
Fond de carte	Fond de carte GeoJSON ?
Légende	Absence de légende
Fonctionnalités	Filtre par départements pour la carte, pour les graphiques il y a une possibilité de téléchargement au format png ; un zoom intérieur et extérieur, et une échelle automatique (remise à l'échelle pour une vision globale)
Analyse des données	
Les variables mises en avant	Pour la carte on ne sait pas quelles sont les données, ou la donnée représentée. En revanche, pour le premier graphique le nombre de personnes hospitalisés et réanimés d'avril 2020 à février 2021 sont affichés; pour le deuxième graphique, sur la même période, c'est le nombre de personnes décédées qui est représenté
Données brutes ou calculées	Données brutes
Date des données	Les dernières données datent de Février
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Carte choroplèthe
Représentations graphiques	Deux graphiques (description dans la ligne "variables" misent en avant) --> graphique en bâtons, en abscisse on a les dates, et en ordonnée c'est le nombre de personnes en milliers avec des pas de cinq, qui est représenté)
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Carte supportée par LizMap, un open source à partir de QGIS, bibliothèque de données jQuery
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Ce site est présenté comme un WebSIG, cependant il n'y a aucune interactivité avec la carte ; de plus l'absence de légende ne nous permet pas de lire la carte. Les graphiques sont intéressants, mais dans la plupart des cas que nous avons étudié, ces derniers servaient de complément à la carte; ici ce n'est pas le cas. Dans ce site les graphiques sont les seuls qui permettent l'analyse et la visualisation de données. Nous ne connaissons pas la date des chiffres-clés, les dernières données sur les graphiques datent de début février ; il semblerait que le site soit abandonné ou non abouti.

	WebGIS EUSALP mobility http://sdi.eurac.edu/AlpinePoKforTransportandMobility/
Présentation générale de la carte	
Fond de carte	Plain map (provenance ?), satellite et OSM
Légende	Légende sous forme de pictogrammes
Fonctionnalités	Zoom et dézoom ; affichage des routes et de régions ; onglet "indicateurs" qui permet l'analyse de sept indicateurs, comme les flux du trafic routier ou ferroviaire ; une fois la requête lancée le résultat s'affiche sous forme de carte, de tableau de données ou de graphique.
Analyse des données	
Les variables mises en avant	La fenêtre des couches, situe les projets de travaux, les smart villages où sont présents de innovations en termes de mobilité, les conflits, les routes et rails transfrontalières, routes transalpines, statistiques de mesures pour le Traffic et la pollution, les terminaux et les connexions d'intermodalité EUSALP
Données brutes ou calculées	Données brutes
Date des données	Inconnue
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Pictogrammes et lignes
Représentations graphiques	Il est normalement possible d'accéder à un graphique dans la partie requête, malgré tout, il y a souvent des erreurs et nous n'avons pas réussi à en afficher un. Je suppose qu'il permettrait donc de mettre en évidence les données sélectionnées sous forme de graphes empilés ?
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	Leaflet, SockJS --> bibliothèque JavaScript et Rshinny
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	

Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Ce WebSIG est très complet, proposant une interface facile d'utilisation, et facilement compréhensible. Il s'agit davantage d'une carte de localisations des projets ou des conflits par exemple. La partie analyse permettra au plus aguerri, de pousser cette dernière en couplant les informations en effectuant des requêtes. Sur ce WebSIG il s'agit d'une simple visualisation des données, que l'on peut faire à travers la carte, le tableau de données où le graphique est très souvent soumis aux erreurs et au final difficile à afficher. Ce WebSIG utilise des données brutes afin de visualiser les lieux d'intérêts sur le territoire, tels que les conflits, les placements des routes et rails. Peut-être est-il plus un outil d'aide à la décision qu'un outil d'analyse?

	Méthode Sankey de Etienne Côme https://www.comeetie.fr/galerie/sankeystif/#stations/Ch%C3%A2telet/37
Présentation générale de la carte	
Fond de carte	Pas de fond de carte
Légende	Pas de légende
Fonctionnalités	Minimap de localisation du réseau métropolitain parisien, menu déroulant pour le changement de station et réglette pour changer la date et l'heure qui fait réagir le graphique pour observer les pics de fréquentations
Analyse des données	
Les variables mises en avant	La méthode Sankey permet de visualiser le nombre de validations, entre deux stations, exprimé en kilo-validations par heure à l'entrée et à la sortie des stations (description faite à partir de ce que nous a dit Laurent Jégou.
Données brutes ou calculées	Données calculées à partir de la méthode Sankey
Date des données	Inconnue
Visualisation des données	
Type de représentation - (Respect de la sémiologie ?)	Représentation par flux
Représentations graphiques	Graphique relié au changement de dates et d'heures concernant les validations pour chaque ligne
Caractéristiques techniques du WebSIG (avantages/ inconvénients)	
Logiciels clients	D3, jQuery et JavaScript
Logiciels serveurs	
Logiciels de base de données	
Synthèse du WebSIG	
Synthèse du WebSIG : la récurrence de ce type de WebSIG, l'intérêt qu'il peut avoir vis à vis des chercheurs, la dimension temporelle, la représentation des flux, critiques éventuelles...	Une nouvelle fois Etienne Côme propose une nouvelle visualisation de la fréquentation des stations de métros parisiennes. La compréhension du fonctionnement du WebSIG est au premier abord complexe. La méthode de représentation sans fond de carte sous forme de schéma est perturbante, mais finalement réalisée de façon à simplifier la lecture. L'unité en kilo-validations/ heure est elle aussi novatrice. Etienne Côme apporte une nouvelle façon de visualiser les données sur le transport et leur utilisation, une nouvelle analyse qui permet alors d'avoir une autre approche peut être plus scientifique qui ne s'adresse pas à tous public contrairement à une bonne partie des WebSIG que nous avons pu recenser dans ce tableau.