Projet Géonumérique
Années 2019 - 2020

Crédits : scusi - Adobe Stock

OSCO : **O**SCO **S**écurise et **C**ontribue à **O**SM

Osco
Sécurise et
Contribue à
Osm

Thomas ANDRE

Victor BONNIN

Jérémy KALSRON

Pierre NIOGRET

Bénédicte THOMAS

ENS DE LYON

UNIVERSITÉ LUMIÈRE LYON 2

UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

GÉO NUM

Remerciements

Avant tout, nous souhaitons disposer de ces premières lignes pour remercier les entreprises comme *Vélogik* et associations telles que *La maison du vélo*, *Parlons vélo*, *Club Cycliste Lyon*, ainsi que les clubs de vélo de la Métropole Lyonnaise (Corbas, Vaulx-en-Velin, AC Lyon Vaise et Lyon Sprint Evolution) pour leurs réponses à nos questions et la diffusion de notre questionnaire à leurs membres.

Également, nous souhaitons remercier les membres du groupe d'utilisateurs local OSM de Lyon qui nous ont accueillis le 11 Février lors de l'assemblée mensuelle. Leur aide nous a été précieuse pour la réalisation de cette application contributive et notamment pour l'orientation de notre projet.

Enfin, nous remercions l'ensemble de nos camarades cyclistes pour leur apport sur ce développement.

Nous souhaitons également remercier les enseignants qui nous ont encadrés pour la réalisation de ce projet.

Sommaire

Introduction

L'application **OSCO**, pour **O**sco **S**écurise et **C**ontribue à **O**SM se base sur un constat simple. Aujourd'hui, de plus en plus d'habitants optent pour des mobilités plus douces en raison des temps de transports ou des difficultés liés aux embouteillages. Pour cela, plusieurs solutions s'offrent à eux. Certains prônent le vélo, là où d'autres lui préfèrent l'assistance électrique ou encore la trottinette. Dans tous les cas, l'ensemble de ces usagers sont vulnérables et doivent partager les pistes cyclables et autres chaussées avec souvent un grand nombre de voitures qui les entourent. Pour faciliter leurs déplacements, certaines applications comme le mode vélo de Google Maps ou encore Geovélo permettent de prévoir un itinéraire et proposent celui qui emprunte le plus de piste cyclable. En revanche, les choix d'itinéraire sont parfois obscurs. Le discours porté par OSCO est donc de mettre en avant la dangerosité des tronçons empruntés par l'utilisateur, pour se déplacer d'un point A à un point B.

Dans cette optique de routage (= conception d'itinéraire), le projet, réalisé exclusivement au niveau de la Métropole de Lyon, se base sur une enquête préliminaire afin de mettre en avant les principaux critères que les cyclistes considèrent comme dangereux. Ces derniers permettent la mise en place d'un indice de dangerosité mobilisé comme coût pour pondérer nos itinéraires. Afin de réaliser des analyses et d'établir un indice de dangerosité, nous mobilisons les données d'OpenStreetMap (OSM), un projet collaboratif de cartographie.

Enfin, dans une dimension participative et citoyenne, l'application permet la contribution à OSM à l'aide d'une interface dédiée au sein de l'application. Pour l'exemple, l'utilisateur pourra indiquer le type de revêtement (asphalte, sable, etc.) ou encore la vitesse maximum des véhicules motorisés de manière à parfaire la base de données et donc, de fait, les futures propositions d'analyse.

L'application OSCO s'adresse donc particulièrement à des utilisateurs quotidiens des nouveaux modes de transports doux. Mais elle permet aussi aux novices de choisir le meilleur itinéraire pour débuter en toute sécurité.

Vous pouvez désormais découvrir l'application OSCO, en ligne, en suivant ce lien : https://osco.anatidaepho.be/

I. État de l'art et questionnement : cyclabilité et dangerosité

a) Présentation rapide de l'état de l'art

La mobilité se définit comme étant un changement de lieu accompli par une ou des personnes, par la réalisation d'un déplacement sur une distance donnée (Lévy, Lussault, 2003). Cependant, cela ne consiste pas seulement en un changement physique de position, puisque cela peut inclure également un spectre sociétal par les pratiques utilisées pour mettre en œuvre cette mobilité.

Chaque individu dispose d'un capital propre lié à sa mobilité, prenant en compte les conditions environnementales de son lieu de vie, ses valeurs sociales voire ses exigences, ses moyens techniques et relationnels, etc. Cet ensemble composite d'éléments définit ainsi un jeu de paramètres qui permet à l'individu de choisir un type de déplacement optimal pour accéder à un endroit souhaité.

Cette agrégation de valeurs et de conditions peut donc évoluer selon le progrès sociétal (multiplication des besoins en mobilité, innovations technologiques) ou alors par l'animation de convictions plus personnelles (notion d'habiter). Les individus ont donc une certaine maîtrise de leur mobilité, par leurs lieux d'habitation de travail, de loisirs. De nombreuses questions sociales peuvent alors être soulevées, pour caractériser ces mobilités, parfois choisies, parfois subies, dans un espace géographique particulier. De même, l'exercice d'une mobilité est révélateur d'une représentation de l'individu, par les choix qu'il réalise pour accomplir son déplacement. Par exemple, l'utilisation – si ce n'est même le simple accès – d'un type de véhicule peut signifier un certain niveau de richesse, et/ou d'une pensée consumériste (Vincent-Geslin et Ravalet, 2015).

Cette question de la mobilité, à englober dans les thématiques de recherche sur l'anthropocène, est ainsi porteuse d'un grand nombre d'entrées d'études. En effet, face à la réalité accomplie de l'impact des activités humaines sur les changements climatiques à venir, il s'agit de s'interroger sur les changements comportementaux à édifier pour contenir cette dégradation planétaire. Ainsi les grandes tendances qui émergent pour faire évoluer nos modes de vie peuvent s'adapter aux domaines de la mobilité et des transports, avec le partage des ressources, la réduction de la consommation, la co-concertation, entre autres. De fait, les politiques publiques s'organisent pour intégrer ces nouveaux enjeux sociétaux et environnementaux, comme c'est le cas en France avec le projet de loi *Orientation des mobilités*, déposé en novembre 2018. Portée dans le cadre de la transition écologique et solidaire, deux des quatre axes principaux retiennent notamment notre attention : « l'urgence environnementale et climatique, qui appelle à changer nos comportements » et la « révolution de l'innovation et des pratiques, qui constitue une formidable opportunité ».

En résumé, il faut ainsi retenir dans les souhaits du législateur que lorsque d'un côté, la voiture individuelle à moteur thermique est amenée à être refoulée en dehors des villes ; d'un autre, le développement de solutions de transports alternatifs et durables est largement soutenu, à l'instar d'un « Plan vélo » crée à hauteur de 350 millions d'euros pour en faciliter et encourager la pratique.

En effet, les mobilités douces sont aujourd'hui mises en avant pour réaliser des trajets courts, considérant notamment que l'usage de la voiture est encore prépondérant : pour des trajets domicile-travail de moins de 4 km, ce sont 60% des actifs qui prennent la voiture – et plus de 75% à partir du 4e km (en 2015, Insee). Pour autant, d'autres moyens de locomotion voient leur utilisation en forte progression, que ce soit par une remise au goût du jour (bicyclette dans les années 2000) ou l'innovation technologique (monoroue, hoverboard) ; en faisant désormais une catégorisation bien dénommée : **les « engins de déplacement personnel » (EDP)**, contenant les « nouveaux véhicules électriques individuels » (NVEI) (*fig. 1*). L'utilisation de ces différents types de transport est donc

soumise à des logiques d'envie et d'efficacité, selon des questions de vitesse, d'ergonomie, de confort, de sécurité et même de style (Héran, 2018).

Figure 1 : Types d'EDP (illustration F. Héran, 2018)

Il faut aussi considérer l'usage de ces engins dans un schéma de déplacement élargi à l'échelle d'une (ou plusieurs) ville(s), prenant ainsi en compte les aménagements conçus pour leur octroyer un espace dans la ville, ainsi que l'intermodalité qui peut s'établir avec d'autres transports intra ou interurbains (Bührmann, 2008 ; Rabaud et Richer, 2019). Cette multiplicité des aménagements dédiés (bandes et pistes cyclables, places de stationnement, capacité d'embarquement dans les trains, etc.) est d'ailleurs elle aussi en forte augmentation, mais pas suffisamment selon les retours des pratiquants (Pucher et Buehler, 2009). La pratique du cyclisme, notamment dans le cadre du **« vélotaf[1] »**, connaît une forte progression depuis les années 1970 et ce petit monde s'organise : les collectivités développent des schémas directeurs favorisant la pratique du vélo (Wendling, 2011), les pratiquants s'organisent en association pour porter leurs revendications et agir – tant que possible – en concertation avec les entités précédentes, les fabricants innovent sur les machines et leurs équipements, la population est éduquée aux bonnes pratiques dès le plus jeune âge, les entreprises publiques et privées peuvent participer aux frais de transport de leurs employés venant à vélo (IKV[2]). Le succès de ces moyens de transport alternatifs réside également dans les bienfaits environnementaux qu'ils apportent, ainsi qu'en la constitution d'un vecteur de santé publique pertinent, par l'amélioration de la qualité de vie avec la réalisation d'un déplacement en étant physiquement actif (Brutus *et al.*, 2017). Nombreux sont ainsi les pratiquants qui, sans affinité particulière pour la bicyclette, s'y sont essayés pour changer leur train de vie quotidien, échangeant du stress pour du temps et du bien-être[3]. Agir dès aujourd'hui doit permettre de mieux envisager l'ensemble des opportunités qui s'offre à la pratique de la mobilité urbaine, en le faisant de façon intelligente et sans céder à la démesure, surtout dans le contexte déjà actuel de multiplication des moyens de transport alternatifs : personnels, collectifs, publics ou par opérateurs privés (Nikolaeva *et al.*, 2019). Toutefois, ce changement de conception est à prendre sur le long-terme, le « tout automobile » imposé durant la période des Trente Glorieuses ayant installé le véhicule motorisé

individuel sur un piédestal, d'apparence indéboulonnable. Il faut alors avoir la capacité et l'audace de mener une politique originale, comme le fait la ville de Grenoble depuis 2014, sous l'impulsion de son maire étiqueté « écolo »[4]. L'espace dédié à la voiture est progressivement transféré aux mobilités douces, afin de leur affecter des espaces sécurisés de circulation apaisée. Forcément, il faut compter pour ce type de réalisation une forte opposition, des résultats longs à arriver et la possibilité qu'un changement de bord politique aux prochaines élections y porte un coup d'arrêt ; mais le parti pris est ici à souligner.

En effet, nous ne faisons que le répéter, mais l'aménagement d'un espace dédié aux usagers des transports alternatifs dans la ville est primordial pour assurer une pratique en toute sécurité.

La connaissance d'une circulation dans un espace relativement sous contrôle doit ainsi permettre de **rassurer les pratiquants**, à la fois par la maîtrise de leurs machines propres, comme de se savoir isolés des autres flux de véhicules lourdement motorisés. Il s'agit que les usagers puissent **se déplacer sans mettre leur vie en danger** à chaque intersection, par leur différence de gabarit ou de réactivité (Pucher et Dijkstra, 2003). Pour l'exemple, une enquête menée en Angleterre auprès de potentiels cyclistes concernant un trajet suivant quatre itinéraires possibles a montré que le critère de la sécurité prévalait notamment sur celui du temps (Hopkinson et Wardman, 1996). Autrement, seulement 1 à 2% des personnes actives des grandes villes australiennes effectuent leur trajet domicile-travail à vélo alors que près de 20% des australiens déclarent rouler au moins une fois par semaine (Fishman *et al.*, 2012). **En France, ce taux d'actifs usant de la bicyclette pour se rendre au travail est de 1,9% en 2015 (Insee, 2017)**. Ce faible chiffre peut s'expliquer en partie par cette fragilité des usagers de modes de transports doux par rapport aux autres types de transports présents sur la route, comme les voitures, les bus et les camions. Différents travaux de recherche s'intéressent ainsi aux comportements cyclistes, mélangeant des approches sociales et techniques, pour déterminer si des facteurs récurrents apparaissent, selon les trajets et pratiques. Pour n'en citer que quelques-uns, il en résulte que les aménagements cyclables sont autant plébiscités que des détours peuvent être concédés pour contourner un endroit dangereux. Il faut toutefois exprimer quelques limites à ces constatations.

En premier lieu, tous les territoires n'ont pas les ressources pour mettre en place ces aménagements, notamment dans les espaces ruraux. Deuxièmement, le comportement de ces usagers varie également selon des critères personnels, suivant leur sexe, leur âge, leur catégorie sociale, leur capacité physique, etc. ; influant ainsi sur leur hardiesse à s'engager sur telle ou telle voie (Fishman, 2016).

Enfin, selon le type même de machine utilisée, peut varier l'efficacité de l'usager. Ceux ne disposant que d'une roue ou plusieurs de petites tailles auront moins de stabilité, tandis que ceux bénéficiant d'une assistance électrique pourront mieux gérer leur effort et franchir les zones avec du relief plus rapidement et sûrement (Pucher et Buehler, 2017 ; Behrendt, 2017). Dans une autre mesure, la sécurité peut aussi passer par la relation aux autres, par rapport à l'appropriation d'un espace et la qualité des aménagements dans la prévention des accidents (Brenac, 2015). L'étude de l'accidentalité routière pour les cyclistes amène également quelques pistes de réflexions (Rapport Santé publique France, 2019). En **2018, 175 cyclistes ont été tués pour 4328 blessés**, représentant ainsi **5% de la mortalité routière** (Bilan de l'accidentalité, ONISR, 2018). Sur la période 2010-2018, la part des cyclistes tués est la seule en augmentation **(+2,2% annuel)** par rapport aux piétons (-0,4% annuel), aux motards (-1,4% annuel) et aux automobilistes (-3,1% annuel). Les principales victimes sont les populations les plus jeunes (plus grande part des blessés, entre 15 et 24 ans) et les plus âgées (plus grande part des tués, au-delà de 55 ans), tandis que la localisation en agglomération ou non joue aussi un rôle puisque les accidents mortels sont, plus nombreux hors agglomération (92 contre 83) et plus importants à des intersections en agglomération (47% contre 22%). Ces données reviennent sur les

caractéristiques individuelles des pratiquants ainsi que la différenciation d'infrastructures aménagées entre les territoires.

[1] Contraction argotique de « vélo » et « taf », pour désigner la pratique consistant à se rendre sur son lieu de travail en vélo.

[2] Indemnité kilométrique vélo (2012 dans le privé, 2016 dans le public).

[3] Hashtag, *France Culture*, 31/08/2018. https://www.franceculture.fr/emissions/hashtag/velo-trottinette-gyroroue-quelle-alternative-aux-transports-traditionnels

[4] *Le Parisien*, 22/08/2019. http://www.leparisien.fr/elections/municipales/municipales-cinq-ans-d-eelv-a-grenoble-ca-change-quoi-22-08-2019-8137218.php

b) Questionnaire réalisé et analyse des réponses

Notre application se veut la plus collaborative possible. Pour ce faire, afin de définir les critères de dangerosité à prendre en compte pour l'établissement de l'indicateur, une enquête a été réalisée auprès des cyclistes. Afin d'être au plus près de la réalité, cette enquête a été relayée par différentes associations cyclistes, clubs sportifs et citoyens. Le sondage est disponible à cette adresse : https://forms.gle/FiG1drAfyhW9F9n77. Celui-ci permet de mettre en avant les critères à prendre en compte à travers les 62 réponses obtenues pour la réalisation de notre itinéraire le plus sécurisé.

Parmi les enquêtés exerçant une pratique cycliste au sein de la Métropole de Lyon, 44.81% d'entre eux privilégient leurs déplacements sur les aménagements dédiés aux cyclistes. Sur une échelle de 1 à 5, ceux-ci affectent une note de ⅗ pour qualifier la sécurité des cyclistes dans la Métropole de Lyon.

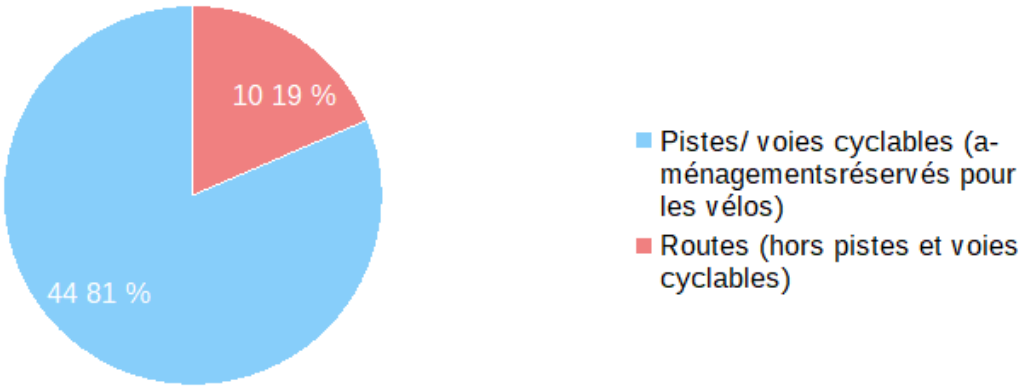

Figure 2 : Présentation des résultats du sondage à la question " Quel type de voie de circulation privilégiez-vous lors de vos déplacements à vélo ? "

Lorsque nous nous intéressons aux facteurs de dangerosité qui impactent le plus les trajets des enquêtés, ceux-ci répondent pour les critères les plus impactant :

- Le trafic
- Les intersections
- Les largeurs de voies empruntées
- La mauvaise visibilité
- Les accotements
- Le mauvais éclairage
- La mauvaise signalisation
- Le revêtement

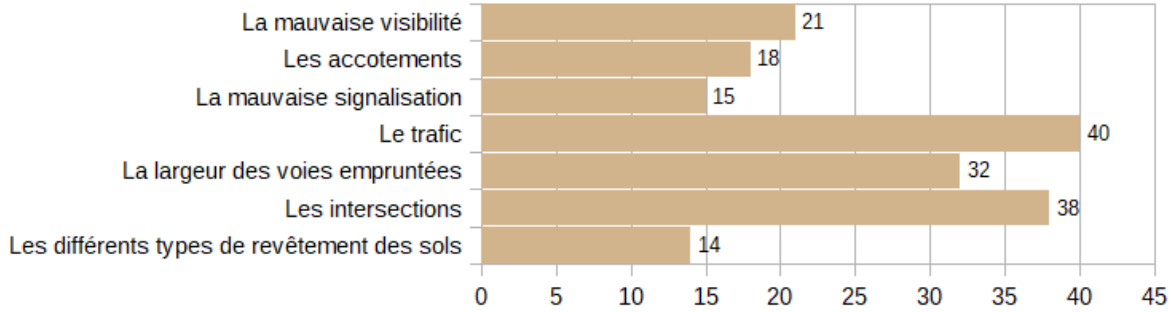

Figure 3 : Présentation des résultats du sondage à la question "Quels facteurs rendent vos déplacements dangereux ?"

À partir de ces réponses, de la disponibilité et accessibilité de la donnée, nous élaborons notre indice de dangerosité.

Pour ce faire nous conservons comme critère de dangerosité :

- Le type de revêtement des voies,
- La vitesse autorisée pour les automobilistes sur les tronçons routiers,
- La présence ou non d'aménagements cyclables et leur nature (pistes cyclables, bandes cyclables).

II. Présentation des enjeux et du fonctionnement de l'application OSCO

a) Pour quel public ?

L'application est donc destinée à un public favorisant ses déplacements via les modes doux : vélo, vélo à assistance électrique, etc.

OSCO propose de comparer différents trajets via des options :

- Trajet recommandé par OSCO (compromis entre le trajet le moins dangereux et le plus rapide),
- Trajet le plus court,
- Trajet présentant le plus d'aménagements cyclables.

À l'aide de son apparence sobre et ressemblante à diverses grandes plateformes de recherche d'itinéraire, OSCO est pensé pour être intuitif, ergonomique et rapide d'utilisation.

b) Pour quoi faire ?

L'application OSCO se compose de deux volets principaux sur le thème du routage. Le routage correspond à une proposition d'itinéraire en incluant une notion de coût influençant le passage par tel ou tel itinéraire. Ici, le coût correspondra, par exemple, à l'indice de dangerosité ou la longueur selon le critère. L'aspect principal de l'application est de proposer un itinéraire en fonction des options choisies pour répondre aux besoins de l'utilisateur. Le trajet sera caractérisé par une note sous la forme d'une échelle alphabétique variant de A à F, définissant la dangerosité du parcours. Par ailleurs, les kilométrages et temps de trajet seront également indiqués.

Dans un second temps, l'outil inclut une notion de contribution. Elle permet de mettre à jour certains tags d'OpenStreetMap, une cartographie collaborative, sur lesquels reposent l'analyse de dangerosité. Par conséquent, l'utilisateur peut contribuer à améliorer le routage jour après jour pour l'ensemble de la communauté. L'application revêt donc un aspect participatif en plus de la fonction opérationnelle de proposition d'itinéraire.

En résumé, OSCO propose quatre types d'itinéraire (trajet recommandé par OSCO, le plus court, le plus aménagé et la possibilité de personnaliser son itinéraire) et souhaite fédérer la communauté cycliste au sein de la Métropole de Lyon.

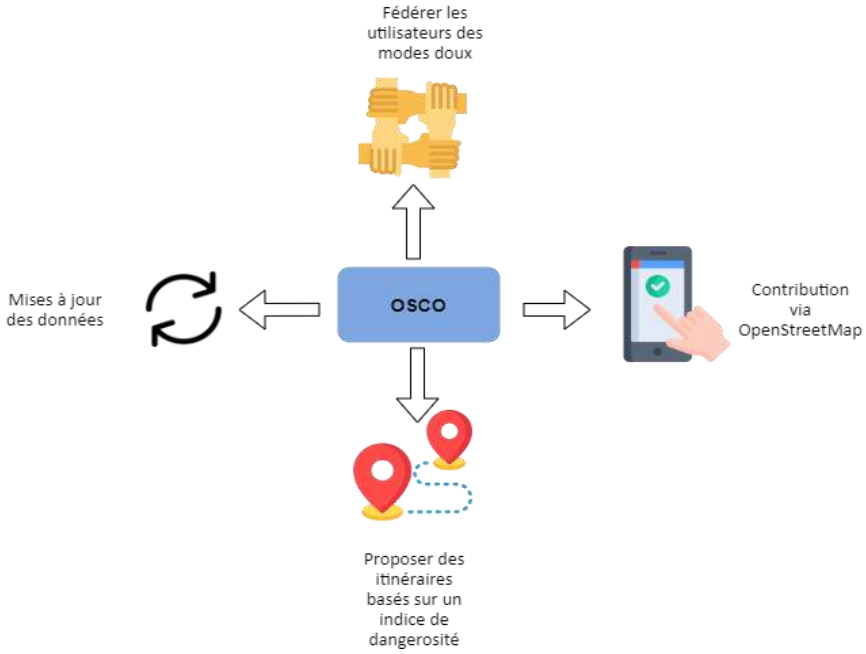

Figure 4 : Schéma des objectifs de l'application OSCO

c) Scénario nominal : guide d'utilisation de l'application

Décrivons le scénario nominal de fonctionnement de l'application, illustré par quelques écrans :

Dès l'ouverture de l'application, l'utilisateur est confronté à un message informatif décrivant :

- Le but de l'application
- La manière de contribuer sur OpenStreetMap
- La donnée représentée

Figure 5 : Présentation de l'interface utilisateur : page d'accueil

Une fois fermée, l'interface de l'application fonctionne comme suit :

L'utilisateur est invité à entrer son point de départ et son arrivée, dans les zones de texte prévues à cet effet (**1**). Dès lors que l'adresse commence à être renseignée, alors une liste, de suggestion de lieux associés à l'adresse inscrite, apparaît.

Une fois indiqués, une série d'options est disponible. Celles-ci permettent de choisir le type d'itinéraire souhaité (**2**) :

- Le trajet conseillé OSCO : trajet le plus sécurisé selon le calcul de l'indice de dangerosité (pondéré par la distance)
- Le plus rapide (en distance)
- Le plus aménagé : Trajet qui prend en compte, au maximum, les aménagements cyclables (pistes, bandes cyclables, voies vertes etc…)

Figure 6 : Interface utilisateur : paramétrage itinéraire

Si aucune option n'est cochée, l'itinéraire généré est **l'itinéraire recommandé par OSCO**. Il s'agit d'un compromis entre l'itinéraire le moins dangereux et le plus rapide.

Par ailleurs, l'utilisateur peut personnaliser le calcul de son itinéraire. En affectant des poids à différents paramètres, il a le choix d'accorder de l'importance ou non, sur une échelle de 1 à 5, à trois paramètres sources de dangerosité :

- **Le type de revêtement** : en affectant une valeur importante au type de revêtement, à cet indicateur, l'usager considère qu'il s'agit d'un paramètre dangereux à prendre en compte dans l'itinéraire.
- **Les aménagements cyclistes** : en affectant une valeur importante aux aménagements cyclistes, l'itinéraire généré favorisera dans la mesure du possible l'emprunt d'aménagements réservés aux cyclistes (bandes, pistes cyclables, voies vertes etc…)

- **La vitesse maximum autorisée par les véhicules** : en affectant une valeur importante à la vitesse maximale autorisée pour les véhicules, l'usager considère qu'il s'agit d'un paramètre dangereux à prendre en compte dans la génération de son itinéraire.

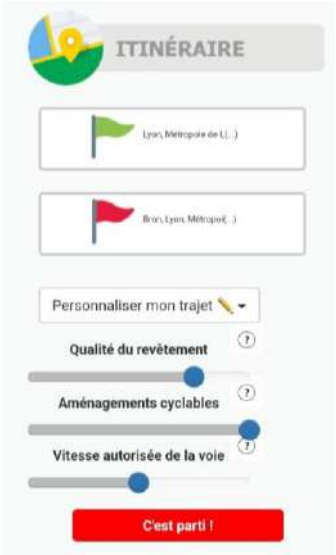

Figure 7 : Interface utilisateur : personnalisation itinéraire

Afin d'orienter l'utilisateur dans le poids à accorder aux trois précédents critères, des aides sont disponibles :

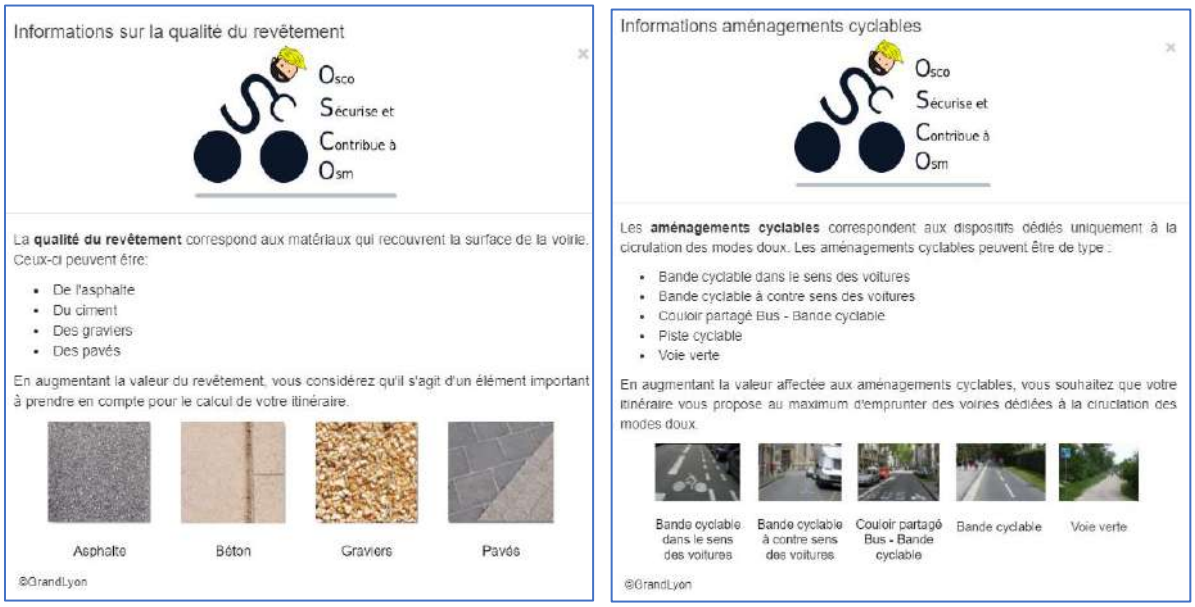

Figure 8 : Aides à la création de l'itinéraire

Une fois le type d'itinéraire choisi, il est nécessaire de valider son choix en cliquant sur le bouton "C'est parti !".

Le linéaire représentant le trajet apparaît alors à l'écran avec quelques informations associées comme :

- Un score caractérisant le niveau de dangerosité du trajet sur une échelle alphabétique oscillant de A à F
- La distance totale de l'itinéraire
- Le temps de trajet

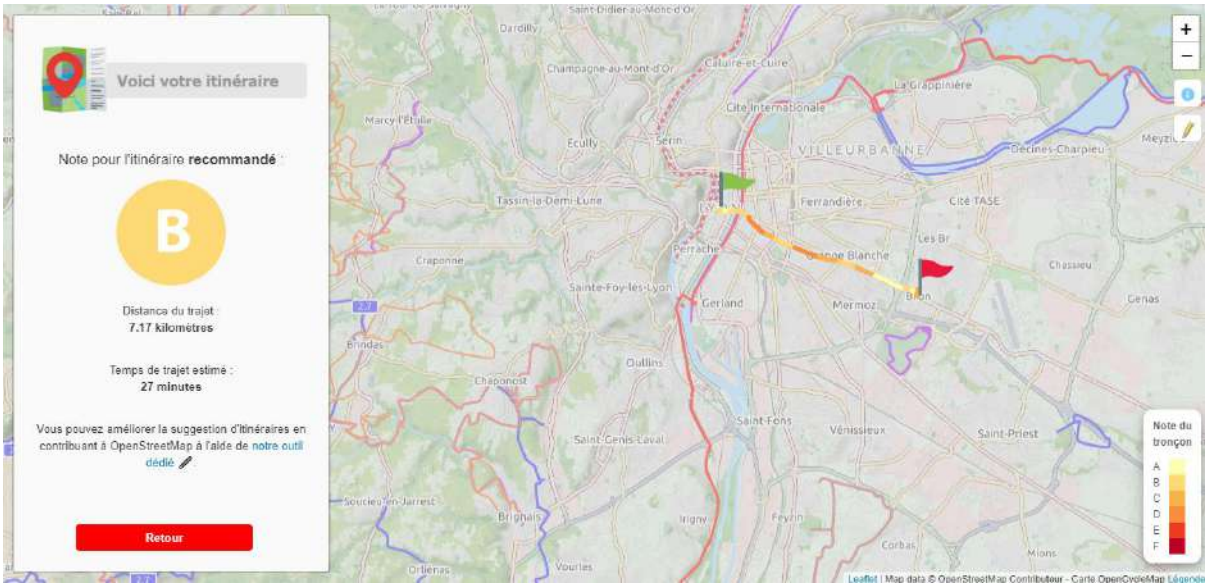

Figure 9 : Interface utilisateur : affichage de l'itinéraire et des informations associées

Variantes possibles : Une fois généré, si l'utilisateur souhaite modifier son point de départ ou d'arrivée, il peut le faire directement en déplaçant le marqueur de départ (drapeau vert) ou d'arrivée (drapeau rouge) sur la carte. L'itinéraire est alors recalculé.

Par ailleurs, un onglet dédié à la contribution est présent sur l'interface. En cliquant sur cette icône, l'utilisateur peut se connecter à OSM **tout en restant sur l'interface de l'application** et ainsi procéder à la partie contribution. Les informations à renseigner pour enrichir les données OSM sont présentes dans une liste déroulante.

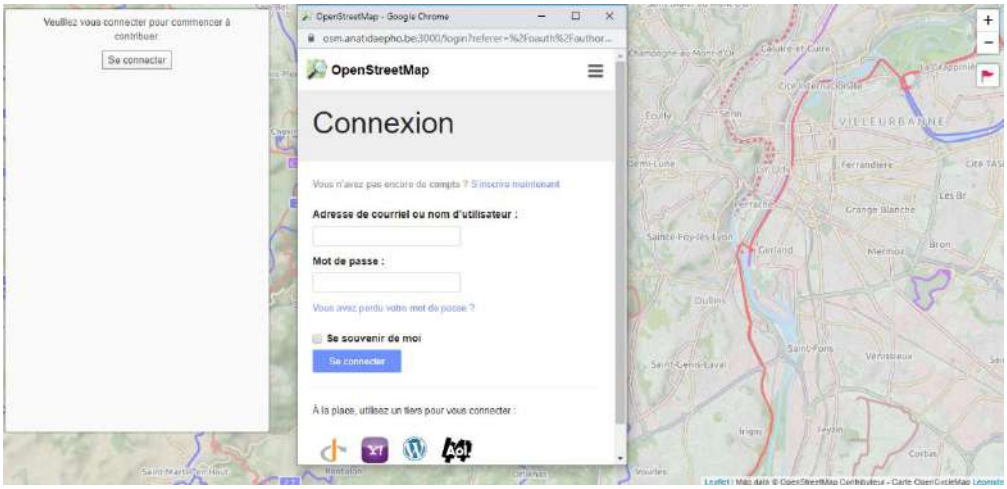

Figure 10 : Interface utilisateur : Connexion à OpenStreetMap(OSM)

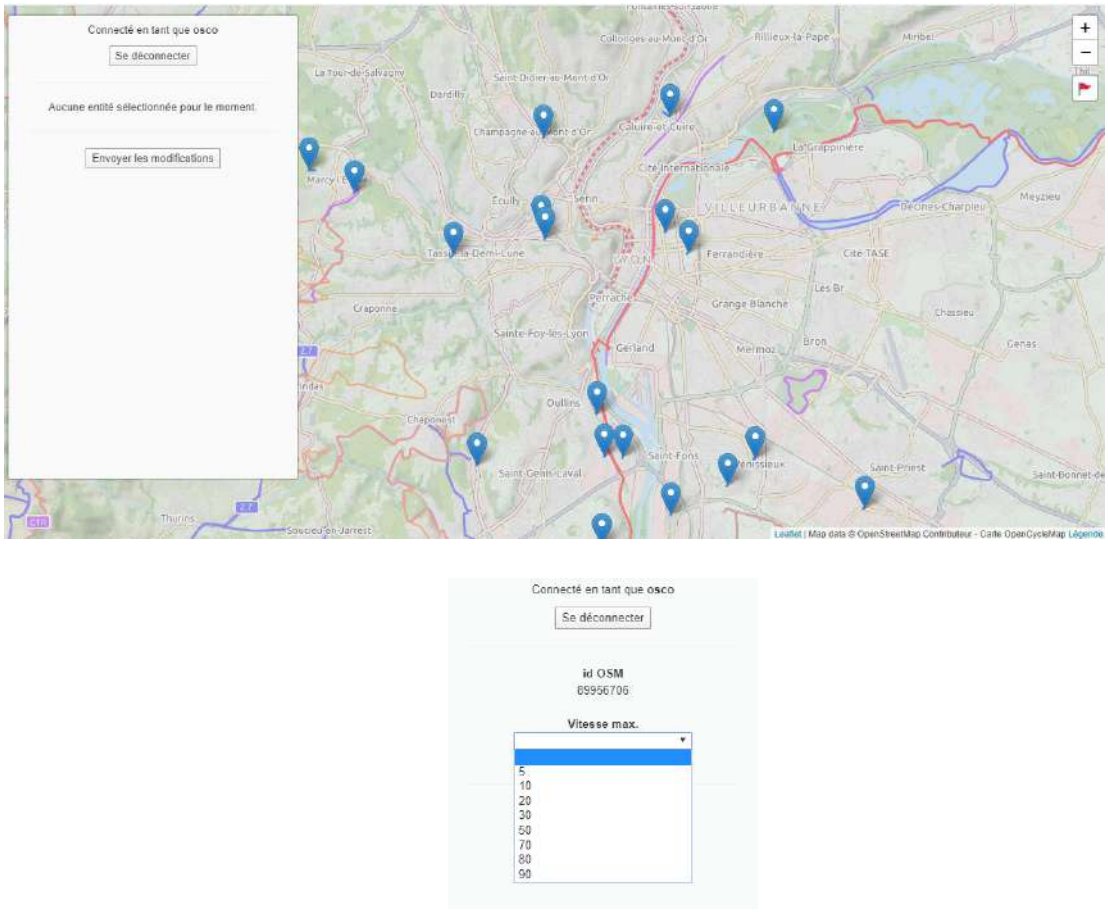

Figure 11 : Interface utilisateur : modification des entités sur OSM

À tout moment, l'utilisateur peut revenir sur l'écran d'accueil de l'application via un le bouton prévu à cet effet.

d) Variante du scénario nominal : gestion des erreurs et temps d'attente lors de l'exécution de la requête

Cependant, divers cas de figure peuvent être réalisés. Il s'agit de variantes du scénario nominal. Tout d'abord, si l'adresse indiquée, dans le champ réservé à cet effet, n'obtient aucune correspondance, alors un message d'avertissement apparaît pour en informer l'utilisateur.

Figure 12 : Affichage message erreur adresse

Par ailleurs, si l'adresse de départ est identique à l'adresse d'arrivée, alors une alerte s'affiche. Celle-ci indique que l'itinéraire ne peut pas être généré.

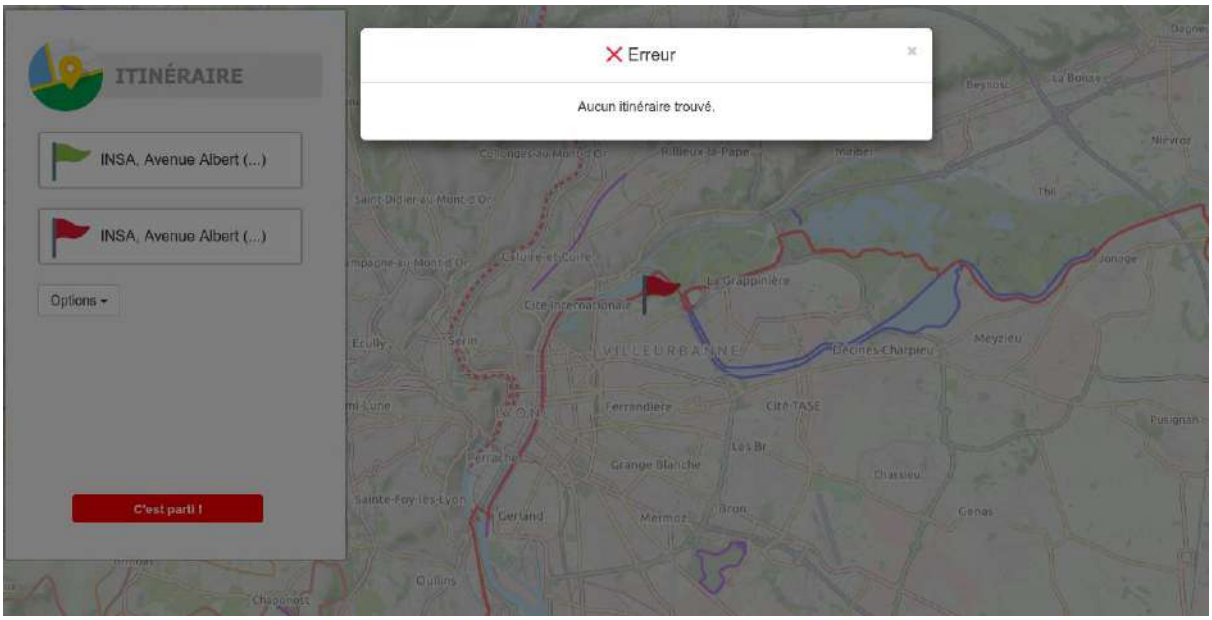

Figure 13 : Affichage message d'alerte aucun itinéraire généré

Identifier les différentes variantes possibles du scénario nominal, permet de lister l'ensemble des cas de figures à traiter pour accompagner au mieux l'utilisateur dans la prise en main de l'application.

III. Gestion de projet, prévision, production

a) Organisation générale du projet

Afin de mener à bien le projet, nous avons décomposé en plusieurs étapes notre recherche. Tout d'abord, nous avons défini le besoin de la façon suivante :

- Réflexion sur le type de public visé
- Description des objectifs de l'application, des usages envisagés
- Rédaction des spécifications fonctionnelles
- Estimation de la faisabilité des objectifs : concepts, logiciels, données… afin de ne pas se laisser surprendre en cours de projet par l'infaisabilité du projet. Cette étape regroupe à la fois la faisabilité technique et la faisabilité prévisionnelle de la charge de travail à prévoir sur le temps imparti.
- Établissement des plans d'action : création d'un planning prévisionnel concernant la réalisation fonctionnelle de l'application. Il s'agit lors de cette étape de décomposer le projet en différentes tâches, estimer leur charge / temps de travail (diagramme de Gantt)
- Liste des difficultés prévisionnelles

Enfin, une fois les objectifs et la faisabilité validés, pour développer l'application nous avons suivi la démarche comme suit :

- Élaboration du scénario nominal de l'application (= utilisation sans gestion d'erreur de l'application) et identification des scénarii qui généreront des erreurs afin de pouvoir traiter exhaustivement l'ensemble des scénarii possibles et accompagner l'utilisateur lors de sa navigation sur l'application
- Élaboration des spécifications fonctionnelles
- Description du processus
- Schéma conceptuel des données
- Maquette ergonomique de l'application

Ainsi, le projet a été décomposé méthodiquement et progressivement, de l'expression du résultat attendu à sa formalisation et validation, afin de le structurer au mieux, de contrôler le déroulement du projet au cours du temps et de pouvoir gérer les imprévus.

b) Répartition des tâches au sein du groupe

Au sein d'un groupe de cinq personnes, la répartition des tâches est la clé de la réussite de manière à tirer parti des cinq profils. Pour cela, nous avons choisi de mettre en place deux répartitions des tâches à deux échelles de temps différentes ; celles-ci correspondant aux deux phases du projet. La première phase correspond au travail réalisé en filigrane de l'année universitaire, l'autre partie correspond à la période des quinze jours dédiés au projet en fin d'année. Afin de percevoir les tâches, deux diagrammes de Gantt ont été mis en place ; sur ces diagrammes la partie en violet correspond au temps estimé pour réaliser une tâche et en jaune, le temps supplémentaire nécessaire pour l'achever.

c) Répartition des tâches dans le temps à l'échelle de l'année

Avant de débuter la semaine de workshop, nous avons tenu à commencer à travailler sur les données et sur l'interface globale de l'application de manière à pouvoir développer l'application dès les premiers jours. Pour cela, l'organisation générale s'aperçoit sur le diagramme suivant :

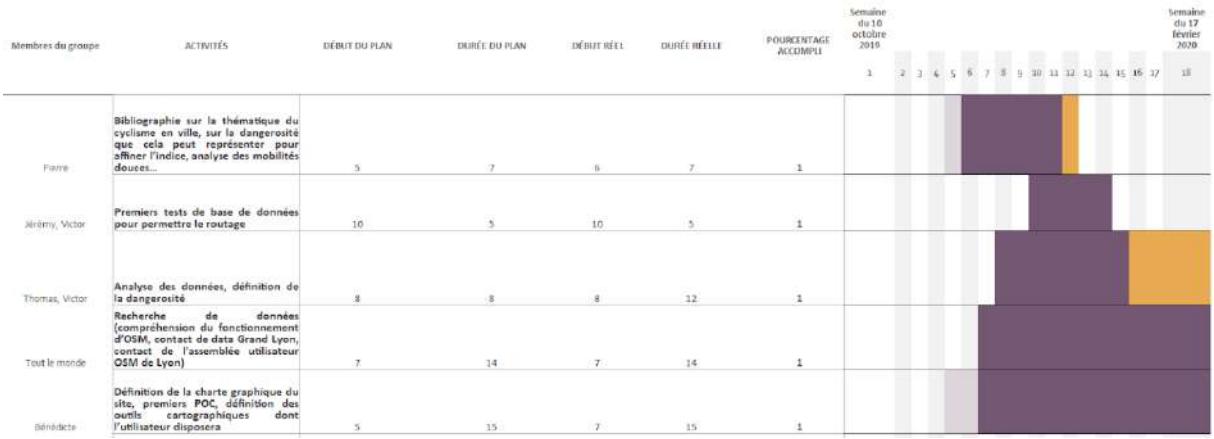

Figure 14 : Diagramme gestion de projet : répartition des tâches (annuel)

À cela, il faut ajouter quelques rendez-vous témoins :

- Rendez-vous avec Luc Merchez le lundi 13/01/2020 pour faire le tour de la question, orienter les recherches bibliographiques et la pertinence d'une telle application

- Rendez-vous avec le groupe d'utilisateur OSM de Lyon le 11/02/2020 pour discuter des méthodes d'import en masse d'open data dans OSM

d) Répartition des tâches à l'échelle du workshop

Pour les deux semaines du Workshop (du 17 au 28 février), la répartition des tâches a été initiée de manière technique pour procéder au développement des composants de l'application en parallèle. Deux membres du groupe se sont donc spécialisés sur les questions de base de données et de développement back, là ou deux autres ont œuvré pour le développement du site web (front). Enfin, un dernier membre s'est occupé du serveur de manière à permettre la mise en commun des deux précédents groupes de travail. Afin de s'échanger les connaissances acquises, nous avons choisi d'ajouter un laps de temps correspondant au sein de notre planning, lui - même partagé en ligne (tableur Google Sheet).

Membre du groupe	ACTIVITÉ	DÉBUT DU PLAN	DURÉE DU PLAN	DÉBUT RÉEL	DURÉE RÉELLE	POURCENTAGE ACCOMPLI
Jérémy, Victor	Création de la base de données (implémentation données, fonctions, etc)	1	7	1	9	1
Thomas, Victor	Analyse des données, définition de la dangerosité	5	2	5	3	1
Jérémy	Administration du site internet	4	4	4	5	1
Thomas, Jérémy	Mise en place du serveur	4	4	4	5	1
Bénédicte, Pierre	Développement du site (informations de base + itinéraire)	1	7	1	9	1
Jérémy	Développement du site (contribution)	6	3	6	4	1
Tous les membres	Mise en commun / intégration des différents composants de l'application	6	4	6	5	1
Tous les membres	Rapport / vidéo / flyer production en anglais	6	5	6	5	1

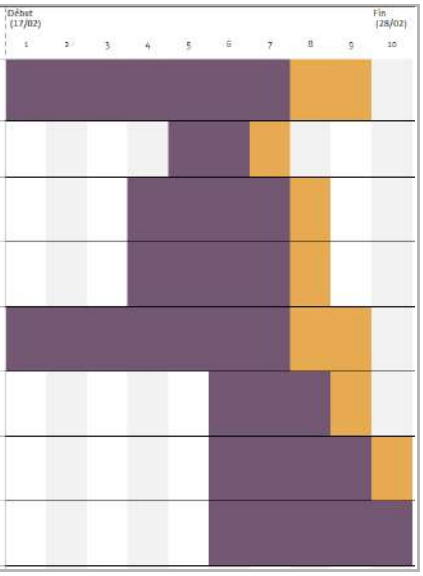

Figure 15 : Diagramme gestion de projet : répartition des tâches (workshop)

IV. Préparation des données

La phase de préparation de données était nécessaire de manière à générer un graphe à partir des données OSM pour permettre le routage à l'échelle de l'étude. L'analyse porte sur la Métropole de Lyon (et les alentours pour limiter les effets de bords que gênerait le routage).

La deuxième mission a été de mettre au point l'indice de dangerosité permettant de pondérer le coût des branches de notre graphe. L'objectif est de mettre en avant principalement 3 critères : la vitesse des automobilistes, la qualité du revêtement et la présence ou non de piste cyclable et si oui le type. Cependant, les informations étant trop lacunaires dans la base OSM, nous avons souhaité l'enrichir avec les données territoriales du Grand Lyon.

a) Récupération des données pour le routage

Nous avons souhaité élaborer notre projet à l'aide de technologies libres. De fait, pour réaliser le routage, nous nous sommes logiquement basés sur la plateforme OSM2PgRouting. Celle-ci permet de comprendre précisément le fonctionnement d'un graphe et de le créer à l'aide des données OSM dans une base PostgreSQL.

1) Fonctionnement général d'un graphe

De manière générale, un graphe correspond à l'abstraction géométrique de l'organisation du réseau routier. Il permet de modéliser les relations entre les routes à l'aide de nœud et d'arc. Chaque nœud du graphe étant mis en relation à l'aide d'un chemin nommé arc ou arête.

Le déplacement entre deux nœuds est pondéré par le coût attribué à chaque arête. Nous souhaitons donc obtenir le chemin à moindre coût pour se rendre d'un nœud A à un nœud B.

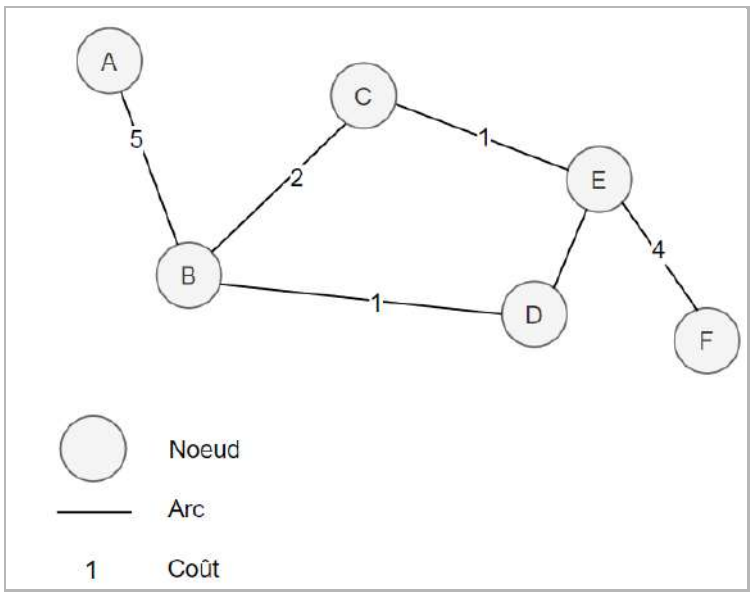

Figure 16 : Représentation schématique d'un graphe routier et son fonctionnement

2) *Création de la base PostgreSQL / PostGIS*

L'installation de la base de données se base sur la technologie PostgreSQL et sur sa couche spatiale PostGIS pour les traitements géométriques. La base de données construite possède principalement une table de filaire de voies (ways) et une table de noeuds (nodes).
Les tables sont créées et gérées automatiquement par la fonction pgr_dijkstra.

ways	
int	**gid**
int	osm_id
double	length_m
text	name
int	source
int	target
hstore	tags
int	nintersect
double	qualite_route_recodee
double	vitesse_recodee
double	qualite_cyclable_recodee
double	qualite_cyclable_recodee_reverse
double	cost
double	reverse_cost
double	secu_cost_risk
double	secu_reverse_cost_risk
double	amenag_cost_risk
double	amenag_reverse_cost_risk
double	court_cost_risk
double	court_reverse_cost_risk
geometry	the_geom

ways_vertices_pgr	
int	**id**
int	osm_id
geometry	the_geom

Figure 17 : Modèle Entité/Association de la base permettant le routage

b) Enrichissement des données

1) *Choix des données*

Afin de générer notre indice de dangerosité, quatre données nous intéressaient particulièrement. Il s'agit de la longueur des voies, de la vitesse des véhicules, de la qualité du revêtement, et du type de pistes cyclables. Toujours dans le but de mobiliser OSM, nous nous sommes donc orientés vers les tags correspondants. En effet, au sein d'OSM, les tags correspondent à ce que nous nommons habituellement "données attributaires". Chacunes des voies se décomposent donc de la sorte :

Figure 18 : Exemple de données OSM - consultation sur https://www.openstreetmap.org/

Cependant, nous nous sommes rapidement heurtés à un problème de complétude et de qualité puisque les attributs étaient relativement peu présents par endroit dans OSM. Un état des lieux permet de se rendre compte des limites de ces données issues de VGI (Information Géographique Volontaire, Goodchild, 2007) :

Données	Problème	Solution
Longueur	Néant	Rien à rajouter
Vitesse	Manque de données sur OSM	Import de données du Data Grand Lyon dans OSM pour compléter la donnée
Surface	Manque de données sur OSM, principalement sur les zones plus rurales de l'espace d'étude Le centre-ville est OK	Import de données du Data Grand Lyon dans OSM pour compléter la donnée
Type de piste cyclable	Néant	Rien à ajouter

2) Problème de qualité de données

De manière générale, il faut bien comprendre que l'ensemble de l'analyse repose sur un agrégat de données issues majoritairement de contributions volontaires. De fait, la complétude comme la pertinence des attributs mobilisés peut attirer l'attention quant à la véracité de l'information dispensée. Là où la base de données OSM nous permet un accès libre à une grande quantité de données régulièrement mise à jour, la qualité peut être variable. Nous avons donc pris le pari de sélectionner cette base de données pour sa reconnaissance actuelle, mais les calculs suivants peuvent en pâtir.

Quelques statistiques simples à l'échelle de notre lieu d'étude pour appréhender la qualité des données :

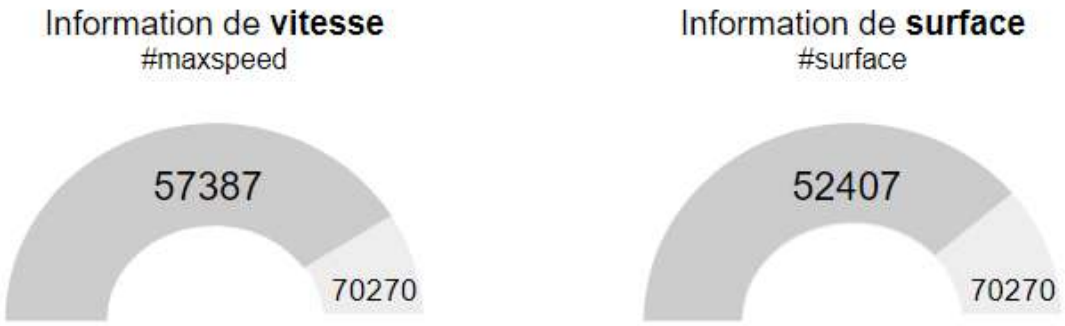

Figure 19 : Analyse de la complétude d'OSM sur les données de vitesse et de surface (à l'échelle du Grand Lyon, en nombre de tronçons)

Environ 75 à 80% des tronçons possèdent donc un tag de type "vitesse max" ou "surface". Ce chiffre assez encourageant doit cependant être nuancé. Puisque la simple présence du tag ne signifie aucunement qu'il est rempli et/ou qu'il est rempli correctement. Cependant la base semble tout de même assez renseignée pour permettre des analyses pertinentes.

3) *Choix de l'import*

Pour compléter cette base de données, l'idée a été de se baser sur les données du Data Grand Lyon qui sont soumises à une licence libre qui permet leur mobilisation sans contrepartie. Les tags que nous souhaitions mettre à jour dans OSM (vitesse et type de revêtement) se retrouvent dans la table *Chaussée et trottoirs* disponible ici : https://data.grandlyon.com/jeux-de-donnees/chaussees-trottoirs-metropole-lyon/donnees. La technique a donc consisté en son import sur notre instance OSM pour parfaire l'analyse de routage.

c) Enrichissement d'OpenStreetMap

1) *Étude de faisabilité*

Dans le but d'accroître la qualité du routage, nous avons choisi d'importer les données manquantes sur notre serveur OSM. Pour cela, nous nous sommes rendus à la réunion mensuelle des utilisateurs OSM de Lyon du 11/02/2020 afin d'obtenir plus d'informations sur les imports en masse et les technologies mobilisables pour réaliser l'opération. Nous avons pris connaissance de plusieurs notions importantes notamment les règles de gestion des attributs. Les échanges lors de cette soirée ont été décisifs dans l'orientation du projet : la jointure attributaire de données linéaires est complexe et nécessite d'allouer un temps non-négligeable à la qualification d'un import automatique, impossible à tenir selon nos échéances. De plus, un certain nombre de critères attributaires sont à respecter pour maintenir la cohérence de la base. Cela nous a amené à héberger une **copie d'OpenStreetMap**, utilisée comme serveur de développement, afin d'effectuer sans risque nos imports et éditions en maintenant une compatibilité de nos développements avec l'instance principale. La mise en place technique de ce serveur est détaillée dans la partie V. L'ensemble de la procédure décrite serait, par conséquent, réplicable sur la véritable instance en changeant au sein du script l'URL de connexion au serveur.

2) Procédure d'import

La procédure de complétion des données OpenStreetMap suit la démarche explicitée dans le schéma suivant. La ventilation des données de chaussée du Grand Lyon a été faite sur les données OSM pour obtenir l'ensemble des attributs. Après recodage pour correspondre aux tags, le script python final permet l'import dans OSM.

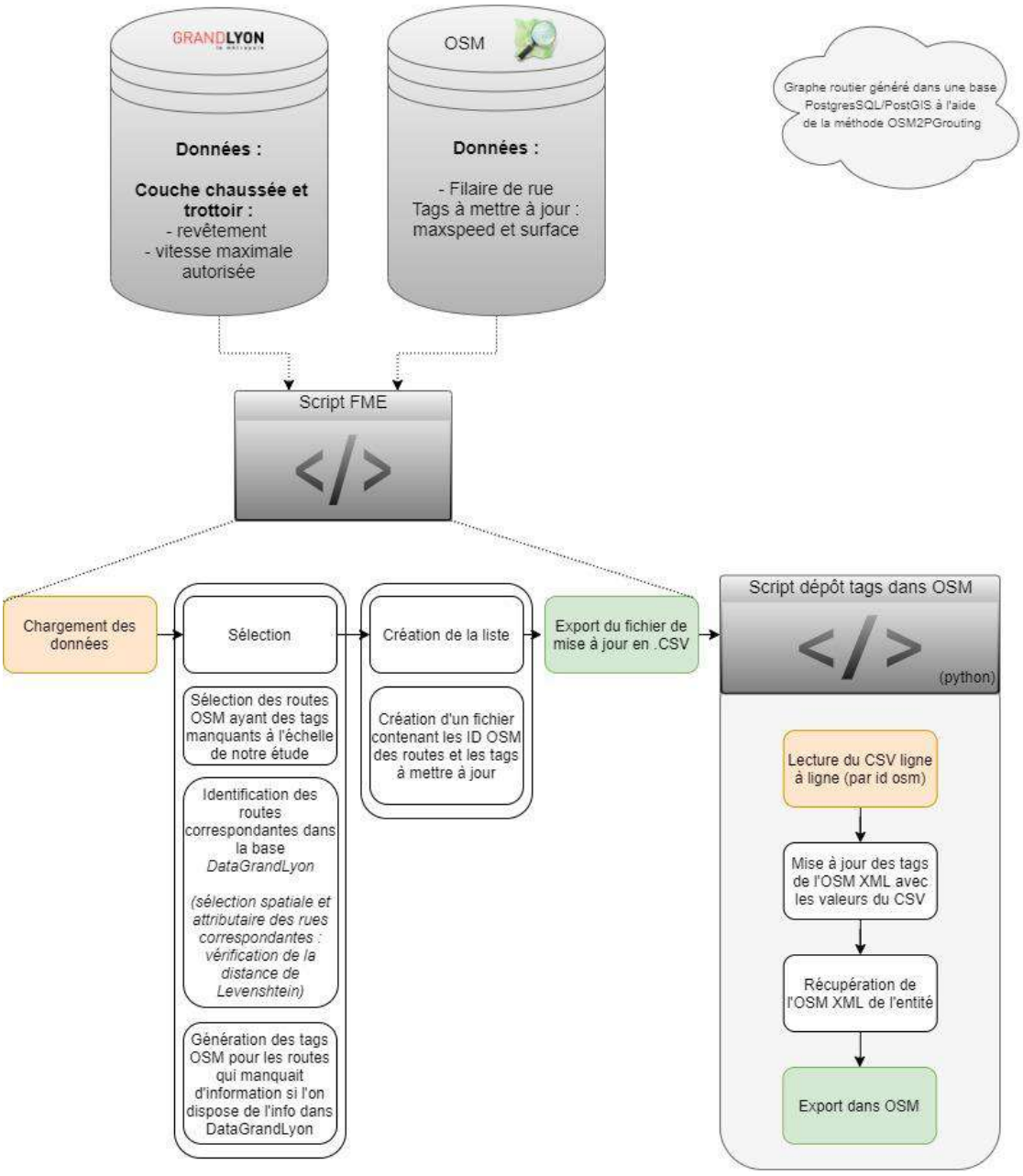

Figure 20 : Schéma de synthèse d'import de données dans OSM

3) *Génération d'un fichier d'entités à modifier*

Pour sélectionner les données (routes OSM) à mettre à jour, nous avons donc mis au point un script FME qui permet de faire le lien de manière géométrique et en se basant sur le nom des rues (distance de Levenshtein) entre les données du Grand Lyon et le filaire de rues OSM. Les entités correspondantes sont déposées dans un .CSV formaté simplement en trois colonnes.

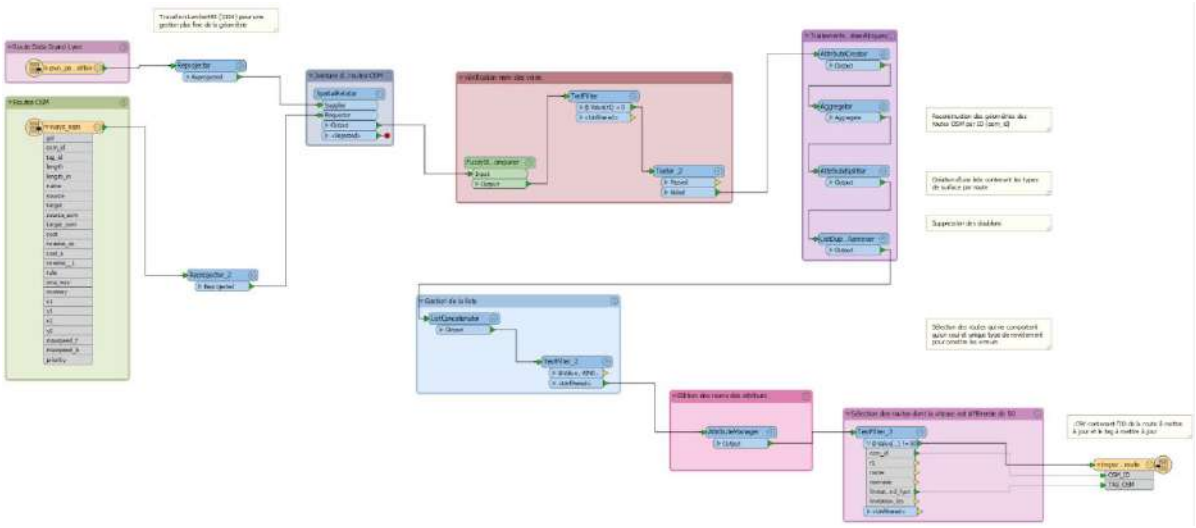

Figure 21 : Modèle FME de génération des données .CSV pour la mise à jour d'OSM

	A	B	C
1	OSM_ID	TAG_OSM_Vitesse	TAG_OSM_Surface
2	4378802	30	paving_stones
3	72994989	30	sett
4	31252817	30	sett
5	37067081	30	sett
6	117395070	30	concrete
7	18464367	20	concrete
8	4274846	30	sett
9	95496076	30	compacted
10	327288096	30	sett
11	9726321	70	compacted

Figure 22 : Données à mettre à jour dans OSM (.csv)

Dans le .CSV, l'identifiant de la route à mettre à jour est présent ainsi que les tags OSM de surface et de vitesse à modifier.

4) *Mise à jour d'OSM depuis le fichier d'entités*

La mise à jour d'OpenStreetMap a été réalisée automatiquement depuis le fichier généré à l'aide d'un script Python. Celui-ci repose sur la librairie Osmapi, qui permet à la fois de se connecter à OSM, de **télécharger** et **téléverser** les données, mais aussi de **manipuler** les fichier OSM XML.

L'import par Osmapi n'est pas la méthode d'import recommandée, qui repose normalement sur la création d'un changeset et son import par un outil type JOSM. Notre instance d'OSM étant déconnectée de l'originale, nous avons cependant voulu expérimenter cette procédure (avec succès).

Après import de la librairie, celle-ci s'instancie à l'aide de la commande suivante :

```
api = osmapi.OsmApi(api="https://osm.anatidaepho.be:3000", username = "osco", password = "20GeoNum20")
```

La méthode d'authentification repose sur un couple identifiant/mot de passe et non OAuth (voir la partie contribution page 4) qui n'est pas implémenté.

Un changeset est ensuite déclaré et regroupe l'ensemble de nos modifications :

```
api.ChangesetCreate({u"comment": u"Mis à jour d'attributs vitesse et surface sur les routes de Lyon. Source : Data Grand Lyon, 2020."})
```

Pour chaque ligne parcourue du CSV, l'entité OSM correspondante est téléchargée :

```
way = api.WayGet(id)
```

Une vérification est effectuée pour analyser si l'entité n'a effectivement pas le tag renseigné afin de ne pas écraser d'information existante, puis on intègre les données à l'entité :

```
if "maxspeed" not in way['tag'] and maxspeed :
  way['tag']['maxspeed'] = maxspeed
  change = True
if "surface" not in way['tag'] and surface :
  way['tag']['surface'] = surface
  change = True
```

Si le flag change a été passé à Vrai, nous avons effectivement mis l'entité à jour et nous téléversons alors la modification sur OSM :

```
if change :
  api.WayUpdate(way)
```

Enfin, on ferme le changeset :

```
api.ChangesetClose()
```

Nous avons mis en place un fichier de log, complété au cours de l'import, afin de suivre les changements effectivement réalisés sur OpenStreetMap par entité. Celui-ci prend la forme suivante :

```
(...)
151370996 : conflit sur surface
91009751 : màj de surface
6074246 : conflit sur surface
34967210 : màj de surface
99685364 : conflit sur surface
8589493 : màj de maxspeed
(...)
```

Les conflits indiquent que nous essayons de mettre à jour un tag avec une valeur différente de celle présente dans OpenStreetMap. Celle-ci n'est alors pas modifiée : priorité à OSM.

Au terme de l'import, un résumé est écrit dans le fichier et permet d'apprécier l'intégralité des changements effectuées, notamment au sujet des informations mises à jour :

```
--- Rapport de màj ---
Nb. de maxspeed màj     : 1722
Nb. de surface màj      : 516
Nb. de conflits maxspeed : 368
Nb. de conflits surface  : 507
Nb. d'erreurs           : 0
```

Une rapide manipulation du fichier log en bash nous permet de vérifier le nombre d'entités mises à jour :

```
$ cat import.log | grep màj | cut -d: -f1 | uniq | head -n-3 | wc -l
2238
```

2238 tags de tronçons OSM ont donc été mis à jour par cette manipulation. Nous pouvons visualiser leur répartition spatiale et remarquer que la complétion s'effectue sur l'intégralité de la métropole :

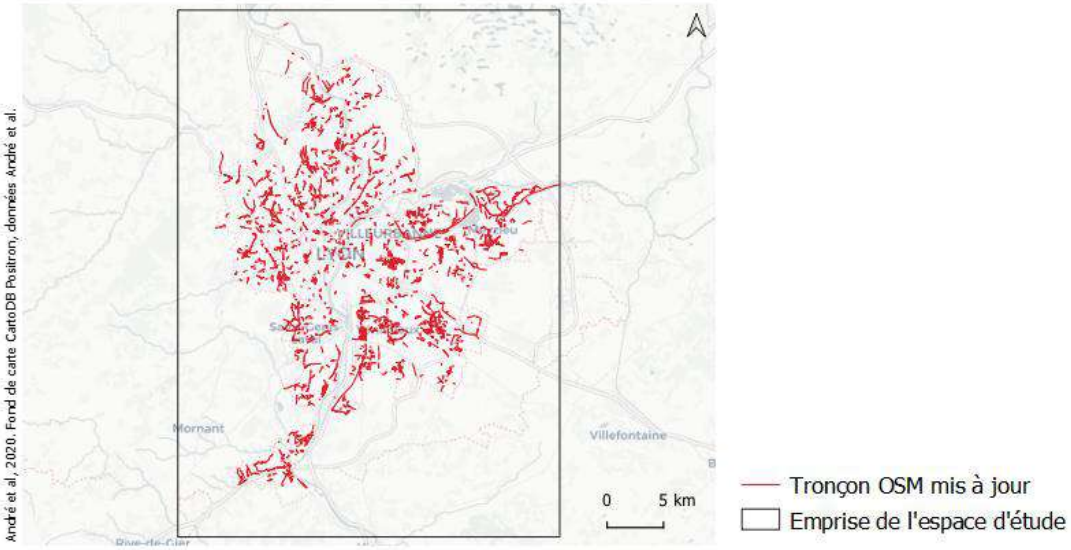

Figure 23 : Données effectivement mises à jour dans OSM

d) Calcul de l'indice de dangerosité

À l'aide des données précédentes, un indice de dangerosité a pu être mis au point. Pour cela, nous avons souhaité tester plusieurs combinaisons afin de déceler les limites de chacun et de sélectionner le meilleur. À la suite de nombreux essais-erreurs réalisés sous tableur, nous avons statué sur la formule suivante :

$$IDa = Qualite_{route} + Vitesse_{vehicule} + Qualite_{cycle}$$

Figure 24 : Indice de dangerosité OSCO

Avec :

- Qualité route : type de revêtement de la route dont voici quelques valeurs possibles :


Asphalte


Béton


Béton long


Pavés artificiels


Goudron


Pavé

- Vitesse véhicule : vitesse maximum autorisée par les véhicules sur la route. Lorsque la valeur de ce champ n'est pas renseignée, par défaut la vitesse maximale sera considérée comme étant de 50 km/h (excepté pour les pistes cyclables).

- Qualité cycle : nature de l'aménagement cyclable dont voici les valeurs possibles :


Bande cyclable


Bande cyclable à contre sens des véhicules


Pistes cyclables


Voie verte

Voie de bus partagée avec une bande cyclable

Pour tirer pleinement partie de l'information et homogénéiser les scores, nous avons dû **coder** l'information de qualité des routes, de la vitesse et du type de piste cyclable.

Ceci permet de rendre l'indice intelligible du point de vue des ordres de grandeur mobilisés. Une note (entre 0.1 et 1) leur est attribuée. 0.1 correspondant à la meilleure qualité, 1 à la plus basse qualité. Pour l'exemple ci-dessous : les routes recouvertes d'asphalte sont bien plus praticables à vélo qu'une route de pavés.

	A	B	C
	ID	Type	Dangerosite
	1	paved	0,4
	2	asphalt	0,1
	3	concrete	0,1
	4	concrete:lanes	0,2
	5	concrete:plates	0,2
	6	paving_stones	0,3

Figure 25 : Recodage de l'attribut type correspondant à la qualité des routes

Cet indice va nous permettre de mettre au point nos coûts, attribués à chaque route. Ceux-ci permettront de **pondérer** et donc **d'impacter** l'itinéraire proposé par l'algorithme.

Pour l'exemple, ici, le Boulevard Laurent Bonnevay, se distingue comme plus dangereux (rouge/jaune) que l'avenue Albert Einstein qui apparaît avec un bon score puisqu'elle dispose d'une piste cyclable (juste au Sud du campus de la Doua).

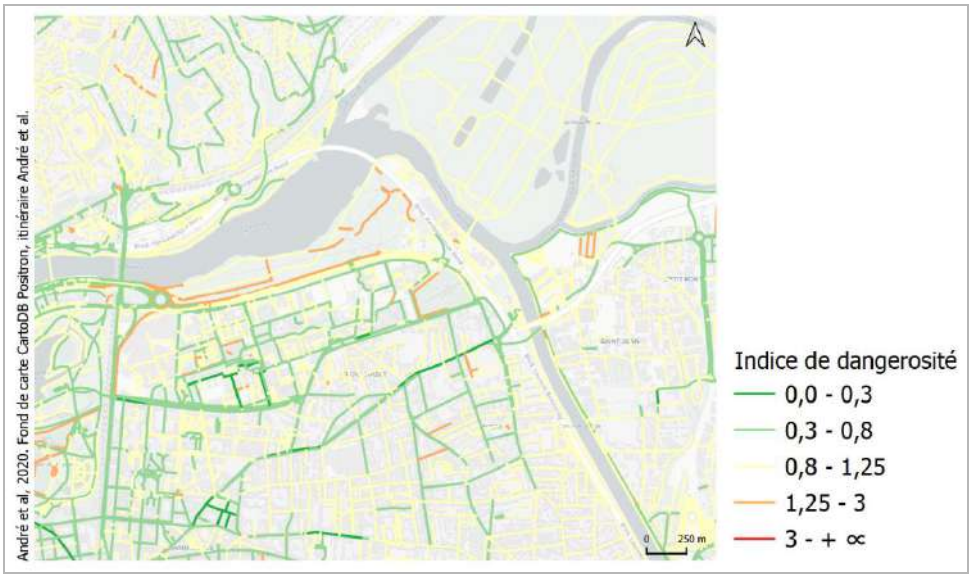

Figure 26 : Exemple de cartographie de l'un des premiers essais de l'indice de dangerosité créé

V. Mise en œuvre technique

L'ensemble de la partie technique suivante s'attache à présenter, au moins de manière générale, les différents procédés et technologies mobilisés pour mener à bien le développement d'OSCO. Comme beaucoup d'applications, elle se compose d'un site internet, visible par l'utilisateur que nous nommons Front et d'une partie base de données / traitements que nous nommons Back. L'ensemble étant déposé sur deux serveurs distants pour permettre le fonctionnement. L'ensemble des explications reprennent cette organisation logique pour livrer les clés du développement de l'application.

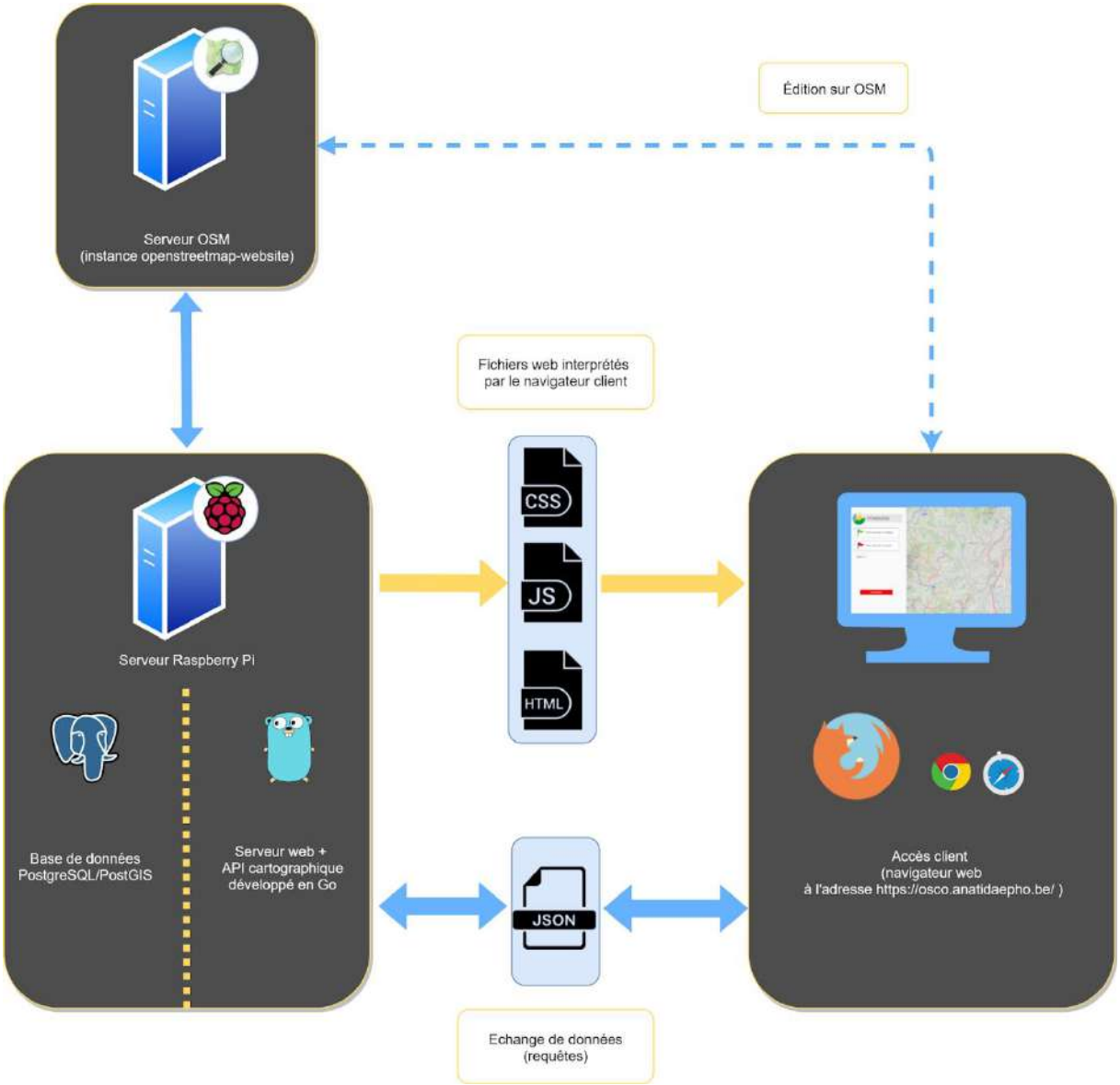

Figure 27 : Schéma d'architecture générale de l'application OSCO

a) Hardware

1) *Serveur applicatif assuré par un nano-ordinateur*

Figure 28 : Un Raspberry Pi 4

Afin de mettre en ligne notre application OSCO à des fins de démonstration, nous avons choisi de nous baser sur un Rapsberry Pi 4, un ordinateur de la taille d'une carte de crédit. Il est utilisé par toute une communauté de passionnés pour des réalisations électroniques. Exploité sous GNU/Linux et connecté au réseau, celui-ci peut jouer le rôle de serveur.

Le système d'exploitation installé sur notre Raspberry Pi est Debian GNU/Linux Buster, complété notamment des paquets PostgreSQL, PostGIS, pgRouting pour la base de données, et Golang pour servir les pages HTML et l'API.

2) *Serveur OpenStreetMap*

L'import de données en masse implique de s'assurer de la qualité de celles-ci. Le temps imparti, insuffisant pour effectuer cette vérification, et la nécessité de manipuler les données OSM pour tester l'interface de contribution nous ont amené à installer notre propre instance d'OpenStreetMap afin de maintenir la compatibilité de l'intégralité de nos développements avec l'API OSM. Déployée uniquement le temps du projet à l'adresse https://osm.anatidaepho.be:3000, celle-ci est hébergée sur un serveur virtuel (VPS) afin de ne pas alourdir notre Raspberry Pi, mais également de respecter la contrainte d'un serveur distant et ses possibles aléas (performance, disponibilité, etc.).

b) Back : serveur et échange de données

1) *Serveur applicatif*

Le serveur applicatif permet la communication entre l'interface web de l'application (cliente, exécutée sur le navigateur de l'utilisateur) et la base de données. Elle va générer les requêtes SQL selon les ordres du client (point de départ, point d'arrivée et type d'itinéraire), récupérer l'itinéraire généré par PostgreSQL et le renvoyer à l'utilisateur.

Plusieurs technologies peuvent se prêter au développement d'un serveur applicatif. On peut notamment citer Python avec la librairie Flask (introduit en cours par Vincent Mora), mais également Ruby et la librairie Rails (utilisés par OpenStreetMap). L'étude de différentes solutions candidates nous a amené à développer notre serveur en langage Go, élaboré par Google en 2009. Utilisé par de nombreux projets d'applications web, celui-ci nous a paru particulièrement indiqué pour sa légèreté (langage compilé) et la possibilité d'utiliser l'application développée à l'aide de celui-ci comme **brique unique du serveur** applicatif : le service du site statique et de l'API sont ainsi assurés par un unique exécutable, éliminant la nécessité d'un serveur HTTP agrémenté d'une passerelle applicative. De plus, sa proximité syntaxique avec le Python nous a permis une prise en main rapide.

2) Fonctionnement général du serveur applicatif

Le serveur est accessible à l'adresse https://osco.anatidaepho.be/ et nous pouvons l'exécuter dynamiquement par SSH à l'aide de la commande `go run server.go` pour effectuer les tests sur l'adresse web.

Plusieurs adresses sont servies : une première pour afficher le site de l'application (fichier HTML, CSS et JS) à l'adresse de base du site ; et celles servant d'API pour faire le lien entre la base de données PostgreSQL et l'interface utilisateur.

De ce fait, nous avons 6 grand types d'URL :

Description de l'action	URL
Accéder à l'application	osco.anatidaepho.be/
API : générer l'itinéraire recommandé par OSCO à partir des points de départ et d'arrivée renseignés par l'utilisateur	osco.anatidaepho.be/api/recommande
API : générer l'itinéraire le plus rapide	osco.anatidaepho.be/api/rapide
API : générer l'itinéraire empruntant le plus de voies aménagées	osco.anatidaepho.be/api/amenagement
API : générer l'itinéraire selon les critères renseignés par l'utilisateur	osco.anatidaepho.be/api/perso
Permettre la contribution à OSM	osco.anatidaepho.be/api/contribution

Nous avons choisi de faire transiter les informations entre l'interface client et le serveur Go par la méthode POST afin de ne pas avoir une URL illisible de type http://osco.anatidaepho.be/requete_itineraire/@45.7170122,4.8976905/@42.7170122,4.8976905 par exemple. Une requête POST nous permet de plus d'envoyer les données servant à la génération de l'itinéraire directement dans **un format JSON**, rendant le traitement plus facile entre la partie Leaflet du front et la base PostgreSQL.

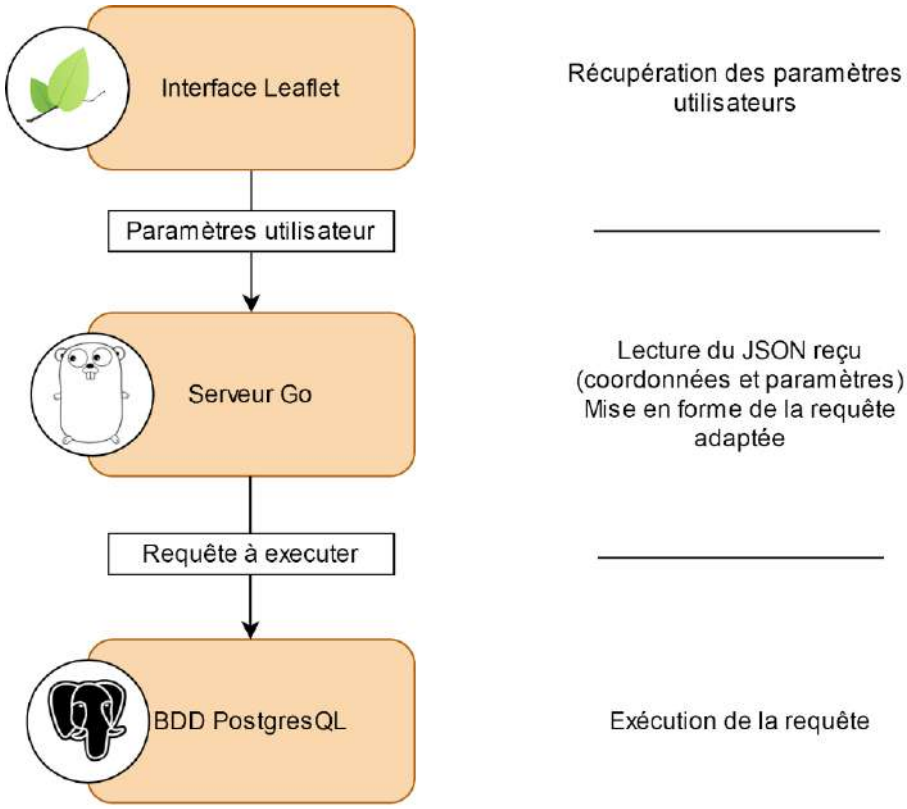

Figure 29 : Transmission client → BDD lors de la demande d'un itinéraire

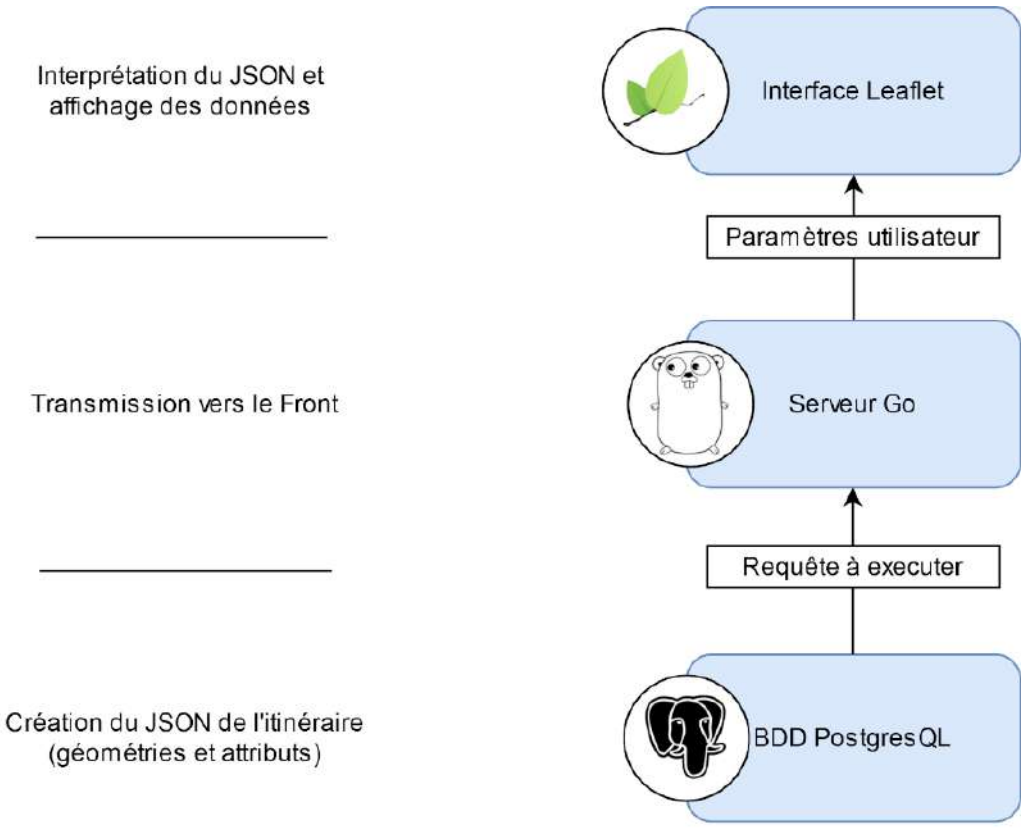

Figure 30 : Transmission BDD → client lors de la demande d'un itinéraire

Le GeoJSON généré par la base de données est ensuite traité par le front pour permettre l'affichage de l'itinéraire, la symbologie des tronçons, le calcul du temps de trajet, etc.

Les plus :

- *Gestion des types d'itinéraire par différentes URL*

La multiplication des URL par type de trajet permet de ne pas complexifier le contenu des requêtes, essentiellements limitées aux positions de départ et d'arrivée.

- *Anticipation*

Dans le cadre du développement, nous avons également voulu anticiper les fonctionnalités pouvant être ajoutées à l'application en vue d'améliorations.
Pour la partie Go, nous avons donc rendu possible l'ajout d'autres coordonnées dans l'envoi de la requête et le traitement réalisé dans le code. Par cela, il nous sera facilement possible d'ajouter des points de passage pour calculer les itinéraires passant par plus de deux points **(prévoir des détours)**.

3) Gestion d'une requête de demande d'itinéraire - exemple pour l'itinéraire personnalisé

Dans le programme principal, pour chaque URL servie est déclarée une fonction associée :

```
http.HandleFunc("/api/perso", handleAPIperso)
```

Ici, la fonction handleAPIperso gère les itinéraires personnalisés. Elle attend, à l'instar des autres fonctions, un fichier JSON envoyé par requête POST par le client (voir partie page 47) qui contient les points de départ et d'arrivé du trajet, complétés pour ce cas précis de critères de surface, de vitesse, et d'aménagement. Une fois reçu, le fichier est analysé et les différentes variables réparties au sein d'une structure de données prévue pour accueillir les informations. Enfin, la requête SQL est générée et envoyée au serveur PostgreSQL dont nous renvoyons la réponse au client.

```
func handleAPIperso(w http.ResponseWriter, r *http.Request) {
   enableCors(&w)
   r.ParseForm()
   geometryJson := r.Form.Get("json")
   var geometries []jsonStructPerso
   json.Unmarshal([]byte(geometryJson), &geometries)
   x1 := geometries[0].GEOMETRY[0].X
   y1 := geometries[0].GEOMETRY[0].Y
   x2 := geometries[0].GEOMETRY[1].X
   y2 := geometries[0].GEOMETRY[1].Y
   qualSurface := geometries[0].CRITERE.SURFACE
   vitesseVehi := geometries[0].CRITERE.VITESSE
   qualCycli := geometries[0].CRITERE.AMENAGEMENT
   reqIti := fmt.Sprintf("SELECT itineraire_perso(%f,%f,%f,%f,%v,%v,%v)\n", x1, y1, x2, y2,
qualSurface, vitesseVehi, qualCycli)
   perso := pgQuery(reqIti)
   fmt.Fprintf(w, perso)
}
```

La fonction pgQuery utilisée à la fin du bloc de code précédent encapsule la connexion à la base de données et l'envoi de la requête. Elle repose sur la librairie Go pq dont la documentation nous a aidé à la mise en place de celle-ci.

4) *Gestion de la requête de contribution*

Lors de l'utilisation de l'interface de contribution à OSM d'OSCO (cf « Contribution à OSM » page 54), l'API est exploitée (/api/contribution) pour récupérer un nombre donné d'entités au sein d'une bounding box dont il manque au moins un des attributs donnés en paramètre. Le JSON envoyé depuis le client est présenté sous cette forme :

```
[{"tags" : ["tag1","tag2",...,"tagn"], "bbox" : [x1,y1,x2,y2], "nelt":nombre d'entités}]
```

Le code exécuté est similaire au code présenté en partie précédente : les attributs sont formatés et insérés en attributs d'une fonction SQL qui renverra un GeoJSON exploitable par le client. Celui-ci contiendra comme seul attribut l'identifiant OSM de l'entité.

5) *Méthode de développement - connexion au serveur*

Le développement a été réalisé à distance directement sur le serveur par connexion SSH. Pour bénéficier d'un environnement de développement moderne, l'IDE libre Visual Studio Code permet l'accès distant de manière transparente à un serveur SSH pour l'édition et l'exécution du code. La mise en place de cette méthode nous a permis de travailler sur le même environnement depuis plusieurs machines, en bénéficiant de la souplesse d'une interface graphique conviviale et des fonctionnalités d'auto-indentation, d'auto-complétions et de coloration syntaxique.

6) *Déploiement du serveur en mode production*

Une fois le développement terminé, nous pouvons déployer le serveur en tant que service du système d'exploitation. Il faut pour cela compiler notre application à l'aide de la commande `go build -o osco_server` afin d'obtenir un exécutable `osco_server` que nous plaçons dans le dossier /usr/local/bin, puis créer un fichier /etc/systemd/system/osco_serverd.service contenant :

```
[Unit]
Description=Serveur OSCO
After=network.target

[Service]
Type=simple
ExecStart=/usr/local/bin/osco_server
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Après avoir activé le service par la commande `sudo systemctl enable osco_serverd.service`, le serveur sera maintenant lancé au démarrage de la machine et redémarré automatiquement en cas de plantage.

c) Back : base de données

1) *Procédure d'import du graphe - osm2pgrouting*

La procédure de création et d'import est décrite pas à pas ici https://github.com/pgRouting/osm2pgrouting et permet de mettre en place un routage simple avec un graphe à l'échelle de notre étude en se basant sur OSM.

Également, pour réaliser nos calculs de coût et permettre l'analyse, il faut compléter le graphe par les attributs (tags) des routes (ways) OSM. Pour cela, osm2pgrouting dispose d'une option qui intègre tous les tags au sein d'un même attribut nommé "tags" que nous retrouvons ensuite dans la base de données.

	osm_id bigint	tags hstore
18	124190080	"name"=>"Rue Carnot", "highway"=>"secondary", "surface"=>"asphalt", "cycleway"=>"lane", "maxspeed"=>"50"
19	215773719	"name"=>"Cours Bayard", "oneway"=>"no", "highway"=>"unclassified", "surface"=>"asphalt", "maxspeed"=>"50"
20	355503830	"name"=>"Avenue Pierre Brossolette", "highway"=>"residential", "surface"=>"asphalt", "maxspeed"=>"50"
21	8109815	"ref"=>"D 1084", "name"=>"Avenue des Platanes", "lanes"=>"2", "highway"=>"primary", "old_ref"=>"N 84", "surface"=>"asphalt", "ma...
22	156639670	"name"=>"Chemin du Vavre", "oneway"=>"no", "highway"=>"unclassified", "maxspeed"=>"50"

Figure 31 : Tags OSM importés dans notre base

Dans la figure précédente, on remarque un type de champ inhabituel pour les tags : le **hstore**, qui permet de stocker des couples de clés/valeurs. Les tags, bien qu'étant stockés sous une même colonne, restent ainsi accessibles indépendamment. Ci-dessous, un exemple de requête utilisant la colonne tags :

```
select osm_id, tags
from ways
where tags -> 'maxspeed' = '50'
```

2) *Calcul des coûts*

L'originalité de l'application OSCO repose sur la proposition de plusieurs itinéraires à l'utilisateur. Celui-ci à le choix entre le plus court, le plus aménagé, celui recommandé par OSCO ou alors il peut jouer sur les paramètres pour concevoir son propre itinéraire. Plusieurs coûts différents sont donc calculés.

Coût sécurisé recommandé OSCO

Le coût sécurisé recommandé prend en compte les trois variables construites précédemment. Il permet d'obtenir le meilleur ratio entre qualité de la surface, vitesse des véhicules et qualité des aménagements cyclables. Le tout est pondéré par la distance.

$$Cost_{securite} = (Q_r + V_v + Q_c) * L$$

Notes :
Qr : Qualité de la surface de la chaussée
Vv : Vitesse maximale autorisée des véhicules motorisés
Qc : Qualité des aménagements cyclables
L : la longueur du tronçon en m.

Coût trajet le plus aménagé

Pour afficher le coût le plus aménagé, l'analyse se base sur le recodage de l'aménagement cyclable réalisé à l'échelle de la métropole. Le tout est pondéré par la distance.

$$Cost_{amenage} = Q_c * L$$

Coût le plus court

L'itinéraire le plus court prend seulement en compte la distance cumulée des tronçons de manière à générer l'itinéraire le plus rapide entre les deux points choisis par l'utilisateur.

$$Cost_{court} = L$$

Enfin, l'ensemble de ces coûts calculés dans la base de données à l'aide d'une requête simple se basant sur les champs recodés précédemment.

Coût personnalisé par l'utilisateur

L'itinéraire reprend la formulation du coût pour l'itinéraire proposé par OSCO, mais y ajoute des indices (a, b et c) qui serviront à moduler le poids donné à chaque variable.

$$Cost_{securite} = (a * Q_r + b * V_v + c * Q_c) * L$$

Voici un exemple de calcul de coût recommandé par OSCO :

```
UPDATE ways
SET secu_cost_risk =
(qualite_route_recodee+vitesse_recode+qualite_cyclable_recodee)*(length_m);
```

Également, afin de gérer **les doubles sens cyclables et les sens uniques**, il a été nécessaire de définir des coûts inverses nommés reverse_cost dans la base de données. Cela permet aux itinéraires d'emprunter les sens interdits aux voitures mais autorisés aux vélos. Pour cela, une série de requête basées sur les tags OSM permet de s'assurer des reverse_cost à mettre à jour. Pour l'exemple, on sélectionne les entités contenant le tag "opposite" qui constitue l'information de contresens cyclable dans OSM et on leur applique un fort poids car cette pratique est réputée dangereuse mais reste possible.
Voici un exemple de calcul de reverse_cost :

```
--- Peuplement de la qualité des pistes cyclables reverse ---
update ways
set qualite_cyclable_recodee_reverse = (
CASE
 WHEN    (tags -> 'highway' = 'cycleway' AND exist(tags,'oneway') AND tags->'oneway' != 'yes')
      OR (tags -> 'highway' = 'cycleway' AND not exist(tags,'oneway'))
```

```
    OR (tags -> 'cycleway:left' like '%track%')
    OR (tags->'cycleway' = 'opposite_track')
    OR (tags->'bicycle' in ('designated', 'permissive', 'yes') AND tags->'highway' in ('path',
'pedestrian', 'service', 'footway'))    then 0.1   --- Piste cyclable ---
 WHEN   (tags -> 'cycleway:left' like '%lane%')
    OR (tags-> 'cycleway:left' like '%opposite%')
    OR (tags-> 'cycleway' in ('opposite', 'opposite_lane'))                    then
0.7 --- Double sens cyclable → davantage dangereux ---
```

3) Calcul des itinéraires

Afin de réaliser les itinéraires choisis par l'utilisateur, l'algorithme **dijkstra** a été mobilisé à travers la fonction éponyme disponible à l'aide de l'installation de l'extension PgRouting. En effet, après un rapide benchmark, il se révèle être un bon compromis par rapport aux autres algorithmes générateurs d'itinéraire comme aStar. Pour réaliser un itinéraire simple, la requête SQL suivante est mobilisée :

```
SELECT a.seq, a.edge, b.the_geom
    FROM pgr_dijkstra('
        SELECT gid as id, source, target,
            cost_risk as cost FROM ways',
        7008, 10060, false
    ) a INNER JOIN ways b ON (a.edge = b.gid) ORDER BY seq;
```

Remarques :

- pgr_dijkstra correspond à la fonction mobilisée pour générer l'itinéraire
- les nombres 7008 et 10060 correspondent au noeud de départ et d'arrivée de l'itinéraire

En retour, la fonction renvoie un tableau avec les tronçons routiers mobilisés :

	seq integer	edge bigint	the_geom geometry
1	1	2508	0102000020E61...
2	2	20715	0102000020E61...
3	3	29044	0102000020E61...
4	4	130558	0102000020E61...
5	5	135375	0102000020E61...
6	6	129175	0102000020E61...
7	7	186118	0102000020E61...

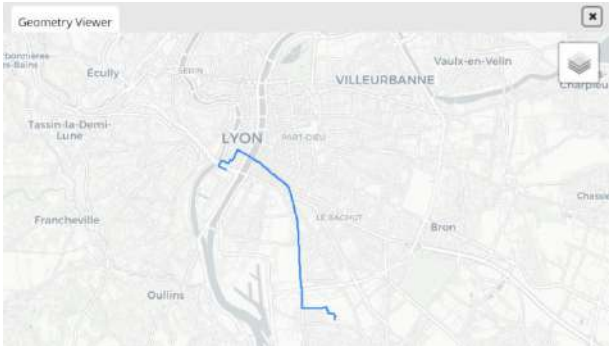

Figure 32 : Tronçons routiers composant l'itinéraire et visualisation dans PgAdmin du résultat

Cependant, afin de permettre le lien avec le front, nous avons choisi de créer des **fonctions PL/pgSQL** qui permettent de générer l'itinéraire à partir des paramètres envoyés. Ces fonctions peuvent se créer à l'aide de l'assistant fonction de n'importe quel client de base de données ou bien en ligne de commande SQL. L'avantage de les réaliser en ligne de commande réside dans le fait que nous pouvons paramétrer plus finement les entrées et les sorties (inputs/outputs).

Étant donné que l'application propose quatre types d'itinéraire (le plus court, celui qui emprunte le plus de voies aménagées, celui recommandé par OSCO et celui personnalisé par l'utilisateur), quatre fonctions sont créées, et correspondent à chacun de ces cas.

Également, les outils Leaflet permettant le développement du site web renvoient une paire de coordonnées lors de la recherche de l'adresse de départ et d'arrivée (X,Y). Les fonctions implémentées permettent donc de sélectionner les noeuds du graphe les plus proches.

L'ensemble du fonctionnement d'une fonction type est récapitulée dans le schéma ci-dessous :

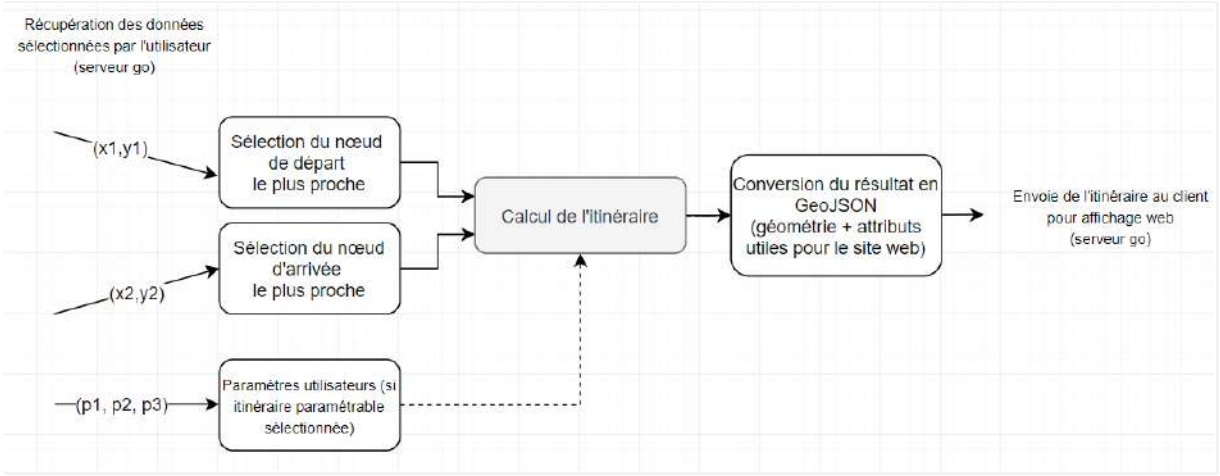

Figure 33 : Schématisation du fonctionnement des requêtes d'itinéraire

Voici la requête pour générer un itinéraire paramétré selon le point de départ et d'arrivée et le type d'itinéraire sélectionné par l'utilisateur

```
# Fonction pour le calcul de l'itinéraire le plus aménagé

CREATE OR REPLACE FUNCTION itineraire_amenag(n1 numeric, n2 numeric, n3 numeric, n4
numeric) RETURNS TABLE(geojson json) AS $$
BEGIN
RETURN QUERY select json_build_object(
'type','FeatureCollection',
'features',json_agg(
   json_build_object(
      'type','Feature',
      'geometry',ST_AsGeoJSON(b.the_geom)::json,
      'properties',json_build_object(
         'length', b.length_m, 'ida', b.ida,
      )
   )
)
) as geojson from pgr_dijkstra('
            SELECT gid as id, source, target,
```

```
                amenag_cost_risk as cost, amenag_reverse_cost_risk as reverse_cost FROM ways',
            (select ways_vertices_pgr.id
      from ways_vertices_pgr, st_distance((ST_SetSRID(ST_MakePoint(n1, n2),4326)),
ways_vertices_pgr.the_geom)
      order by st_distance asc
      limit 1), (select ways_vertices_pgr.id
      from ways_vertices_pgr, st_distance((ST_SetSRID(ST_MakePoint(n3, n4),4326)),
ways_vertices_pgr.the_geom)
      order by st_distance asc
      limit 1), true
      ) a INNER JOIN ways b ON (a.edge = b.gid);
END;
$$ LANGUAGE plpgsql;
```

Ici, l'utilisateur choisit donc itineraire_amenag qui correspond à l'itinéraire le plus aménagé. Il part de l'adresse représentée par le couple de coordonnées (n1, n2) pour se rendre à l'adresse (n3, n4).
Enfin, pour l'itinéraire personnalisé, l'utilisateur peut choisir de moduler un paramètre (entre 1 et 5) pour optimiser le passage par des voies plus sécurisées, des voies où les véhicules motorisés évolueront à des vitesses plus faibles ou par des voies de meilleure qualité selon ses préférences.
Pour cela, il faut disposer de trois paramètres en plus lors de la déclaration de la fonction :

```
# Fonction pour le calcul de l'itinéraire personnalisé par l'utilisateur

CREATE OR REPLACE FUNCTION itineraire_perso(n1 numeric, n2 numeric, n3 numeric, n4 numeric,
n5 numeric, n6 numeric, n7 numeric) RETURNS TABLE(geojson json) AS $$
BEGIN
RETURN QUERY


select json_build_object(
'type','FeatureCollection',
'features',json_agg(
   json_build_object(
      'type','Feature',
      'geometry',ST_AsGeoJSON(b.the_geom)::json,
      'properties',json_build_object(
      'length', b.length_m, 'ida', b.ida
      )
   )
)
) as geojson from pgr_dijkstra('
            SELECT gid as id, source, target,
                ((qualite_route_recodee*'||n5||'+vitesse_recode*'||n6||'+qualite_cyclable_recodee*'||n7||')/(1/(length_m+1))) as
cost, ((qualite_route_recodee*'||n5||'+vitesse_recode*'||n6||'+qualite_cyclable_recodee_reverse*'||n7||')/(1/(length_m+1))) as
reverse_cost FROM ways',
            (select ways_vertices_pgr.id
      from ways_vertices_pgr, st_distance((ST_SetSRID(ST_MakePoint(n1, n2),4326)), ways_vertices_pgr.the_geom)
      order by st_distance asc
      limit 1), (select ways_vertices_pgr.id
      from ways_vertices_pgr, st_distance((ST_SetSRID(ST_MakePoint(n3, n4),4326)), ways_vertices_pgr.the_geom)
      order by st_distance asc
      limit 1), true
      ) a INNER JOIN ways b ON (a.edge = b.gid);
END;
$$ LANGUAGE plpgsql;
```

n5 (numeric) paramètre 1 : modulation du coût alloué à la qualité de surface des routes
n6 (numeric) paramètre 2 : modulation du coût alloué à la vitesse max. des véhicules
n7 (numeric) paramètre 3 : modulation du coût alloué au type de voies cyclables

4) Mise à jour en continu

Afin de conserver une donnée à jour et de proposer des services de routage précis, nous avons choisi d'établir une procédure permettant la **mise à jour quotidienne** des données de notre base provenant d'OSM. Des exports sont proposés quotidiennement par plusieurs fournisseurs, parmi lesquels Geofabrik . Cependant, gérant notre propre instance d'OpenStreetMap, nous avons dû mettre en place cette fonctionnalité côté serveur OSM. L'export est réalisé à l'aide de l'outil Osmosis, qui permet de manipuler les données OSM au sein d'une base APIDB (le schéma de base de données d'OpenStreetMap) :

```
osmosis --read-api-db host="localhost" database="openstreetmap" user="osco"
password="20GeoNum20" validateSchemaVersion="no" --write-pbf
file="/var/www/html/export/export.pbf"
```

L'export est automatisé au sein d'un script bash qui renomme et conserve la version précédente, et l'exécution est planifiée quotidiennement à **1h00** à l'aide d'une règle cron qui écrit un log de l'opération :

```
0 1 * * * /home/osco/export.sh >> /home/osco/export.log 2>&1
```

Enfin, les fichiers sont servis par le serveur HTTP Apache à l'adresse http://osm.anatidaepho.be/export/ .

Côté serveur OSCO, la procédure d'installation créée comporte un script de peuplement de la base de données qui peut être exécuté quotidiennement. La règle cron suivante est appliquée pour l'exécuter quotidiennement à 3h00 :

```
0 3 * * * /home/pi/installation_osco/import_data.sh >> /home/pi/import.log 2>&1
```

5) Pérennisation de l'application

Dans le but de pérenniser l'application, une procédure d'installation d'OSCO a été créée sous la forme d'un ensemble de scripts bash et SQL permettant l'initialisation d'OSCO, le peuplement de la base, et le déploiement du serveur Go.
L'intégralité du code produit a été mis à disposition sur Github afin que chacun puisse récupérer le code et refaire l'installation que nous avons mis en place. Les instructions y sont présentes.
Nous avons également souhaité apposer une **licence libre** au projet. Le serveur Go est ainsi sous licence AGPLv3 et le client sous GPLv3, pour que quiconque intéressé par ce développement puisse mettre en place sa propre instance, mais également l'adapter à son besoin et l'améliorer.

d) Front

1) *Visualisation*

i. *Granularité de l'information*

L'information affichée présente une granularité (= un niveau de détail de l'information) par tronçon routier. Les tronçons routiers ont été importés d'OSM puis redécoupés via un script, pour créer un graphe à partir des nœuds routiers correspondant aux intersections.

ii. *Choix des données à afficher*

OSCO propose de visualiser les linéaires des itinéraires. Chaque tronçon est représenté avec une couleur caractérisant le degré de dangerosité de celui-ci. La légende est présente dans un cadre à part à droite de la carte dès lors qu'un itinéraire est généré. Si aucun trajet n'est généré alors la légende ne s'affiche pas. La progression géométrique est la méthode de discrétisation mobilisée pour proposer une répartition de l'indice de dangerosité en six classes (ensuite affinée manuellement).

Pour la partie contribution, le choix a été fait d'indiquer par des markers, les localisations des tronçons où les informations ne sont pas entièrement renseignées dans OSM. Ainsi, l'utilisateur souhaitant contribuer aperçoit au premier regard les entités non complètes et par conséquent celles où l'ensemble des informations est renseigné. Ceci a été pensé dans une optique d'enrichissement de données. En effet, nous ne souhaitions pas que les utilisateurs puissent modifier ou ré-assigner les informations d'un tronçon alors qu'elles sont déjà toutes renseignées dans OSM.

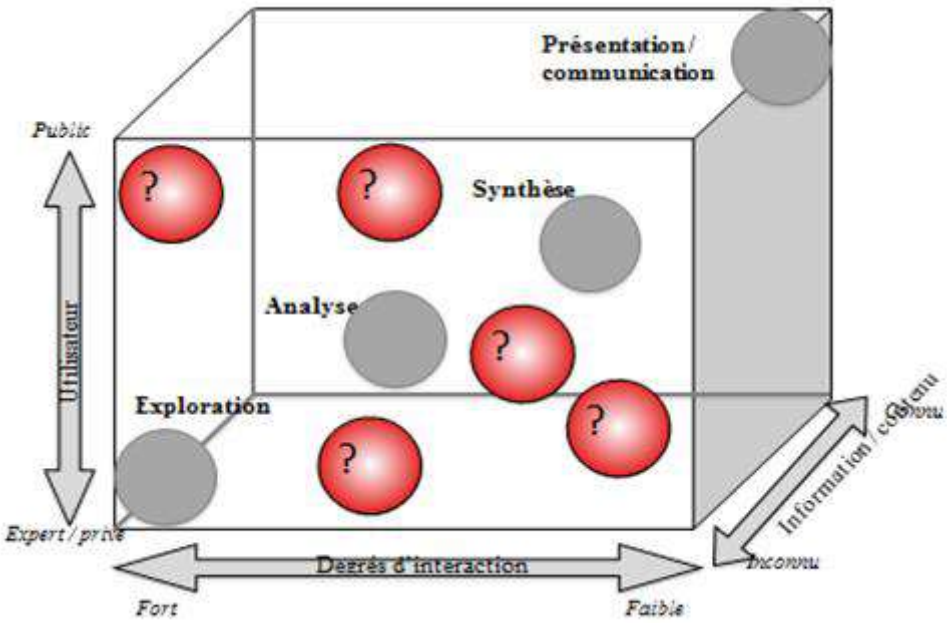

Figure 34 : Cube analyse Mac Eachren, cours de M2

Selon le cube de Mac Eachren, qui effectue une synthèse entre type d'utilisateur, le degré d'interactivité et le degré de connaissance de la donnée représentée sur des applications, notre application propose de **la synthèse de donnée**.

En effet, le public visé est un public large pour lequel OSCO propose un degré d'interactivité limité à moyen. Cependant, le processus collaboratif mis en place permet de rendre acteur l'utilisateur au sein du projet OSCO. La donnée est explicitée simplement dès l'ouverture de l'application mais ne détaille pas l'ensemble des poids affectés aux critères constituant l'indice de dangerosité. L'application se positionne donc entre les catégories synthèse et analyse de donnée.

iii. Choix de visualisation

Fond de carte :

La mise en œuvre graphique s'est articulée autour de deux maîtres mots : simplicité et clarté. L'objectif est de mettre en œuvre un aspect graphique adapté aux utilisateurs de l'application.

L'information représentée se positionne sur un fond de carte généré par OSM, appelé **OpenCycleMap**. Ce fond cartographique présente l'avantage, de représenter une multitude d'information concernant les aménagements cyclables (leur nature pistes cyclables, bandes cyclables…), parkings à vélo, les fontaines à eau etc. En effet, il nous semblait judicieux de compléter l'affichage de l'itinéraire par l'apparition d'éléments présents sur le fond OpenCycleMap.

Pour garder une interface épurée, un renvoi vers la légende OpenCycleMap a été préféré plutôt que de l'indiquer clairement dans un cadre sur l'application qui surchargerait l'espace de visualisation d'itinéraire.

Adresse :

Dès lors que l'adresse de départ/arrivée est validée par l'utilisateur, un marker se positionne pour localiser l'emplacement renseigné. Ainsi, l'utilisateur visualise, rapidement et simplement, sur la carte, son point de départ et d'arrivée.

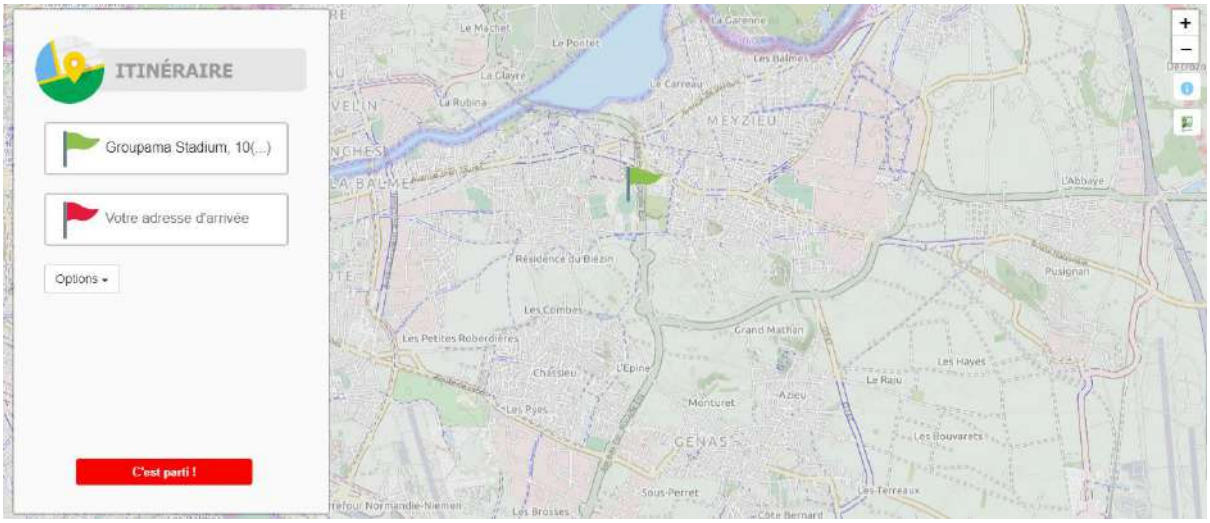

Figure 35 : Visualisation par un marqueur de l'adresse renseignée

Itinéraire :

L'itinéraire quant à lui est représenté par un dégradé de couleur. L'intensité de la couleur correspond au degré de dangerosité des tronçons. Ce choix sémiologique a été réfléchi de façon à ce que la représentation colorée de l'itinéraire ne se confonde pas avec des données de trafic. En effet, de nombreuses applications concernant le trafic routier mobilise une palette de couleur oscillant du vert

au rouge pour représenter les tronçons en fonction du degré d'affluence des routes. Or, ici, pour notre application, il s'agit bien de représenter le degré de dangerosité. Par ailleurs, les perspectives d'évolution de l'application prévoient d'intégrer les données trafic en temps réel. Représenter symbologiquement ces deux informations avec la même palette de couleur ne serait pas adaptée.

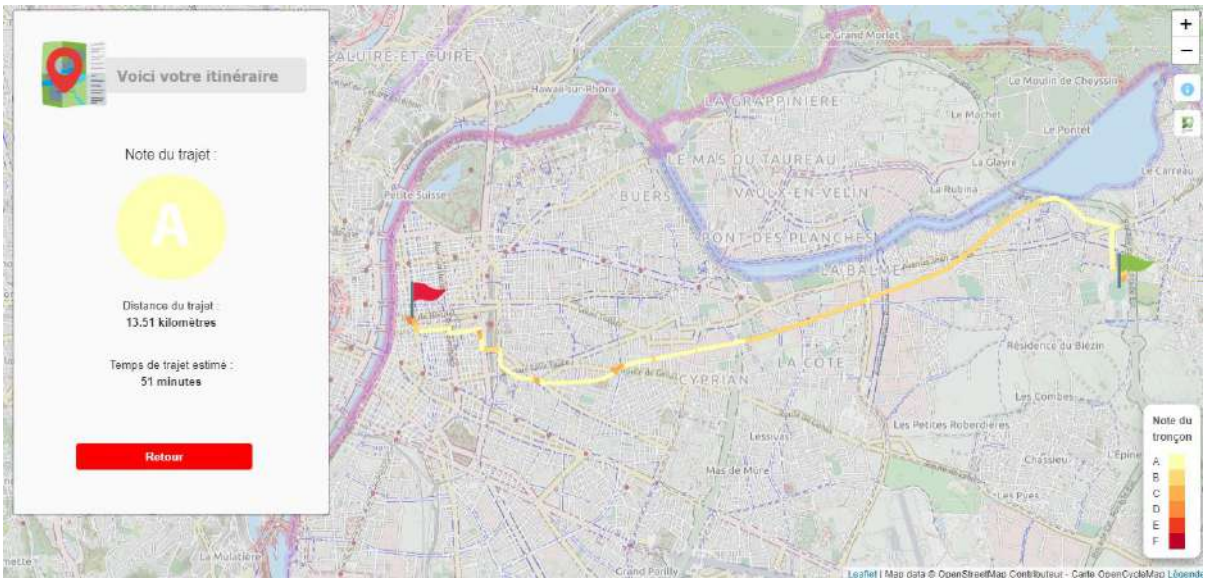

Figure 36 : Visualisation de l'itinéraire et des informations associées

Contribution :

Concernant la contribution, les entités pour lesquelles les champs ne sont pas renseignés ou pas entièrement, sont indiqués par un marker. Ainsi, l'utilisateur perçoit au premier coup d'œil, la localisation des tronçons à enrichir. Les informations manquantes pour certains champs sont disponibles via des listes déroulantes afin de garantir la qualité de l'information transmise à OSM.

Temps d'attente lors de l'exécution de la requête :

L'exécution de la requête qui génère l'affichage de l'itinéraire nécessite un petit laps de temps d'attente. Pour ce faire, un message apparaît pour indiquer à l'utilisateur que son trajet est en cours d'affichage. Ce choix de visualisation permet à l'usager d'être informé de ce qui est en cours d'exécution.

Figure 37 : Visuel d'attente pendant l'exécution de la requête

iv. Outil d'interactivité avec la carte

Le support numérique cartographique implique de proposer un certain degré d'interactivité avec l'utilisateur.

Voici les éléments interactifs de l'application OSCO :

- Niveau de zoom : par la molette de la souris ou les boutons en haut à droite

- Sélection spatiale : la sélection spatiale est possible dès lors que l'utilisateur souhaite modifier son point de départ ou d'arrivée, une fois l'itinéraire généré. Il lui suffit simplement de déplacer les marqueurs sur la carte.

  En outre, une fois l'itinéraire généré, l'utilisateur peut modifier son point de départ et d'arrivée, en déplaçant directement le marker départ/arrivée sur la carte.

  Par ailleurs, l'onglet collaboration permet en quelque sorte d'effectuer des sélections spatiales. En effet, après avoir entré ses identifiants, l'utilisateur peut modifier des entités repérées par un marqueur. L'interface de modification, épurée et simpliste donne lieu à une modification des entités (informations modifiées à sélectionner dans des champs) de manière facile et rapide sans être noyé au milieu de multiples informations.

- Requête : L'utilisateur requête la base de donnée dès lors qu'il clique sur le bouton « C'est parti ! » ou bien lorsqu'il personnalise son itinéraire. Un système de slider permet d'accorder de l'importance au critère sélectionné. L'utilisation des sliders permet à l'utilisateur d'affecter de l'importance à tel ou tel critère. Ils lui permettent d'être le **créateur** de son propre itinéraire.

- Interactivité bouton d'information : Un bouton information est présent pour guider l'utilisateur sur le but de l'application, la contribution à OSM.

- Interactivité de la légende associée aux itinéraires : La légende est interactive avec les itinéraires générés. Elle n'est pas permanente. Si aucun itinéraire n'est affiché ou dès lors que l'utilisateur sort de la visualisation d'itinéraire alors la légende ne s'affiche plus. Il s'agit d'un élément qui ne doit pas être permanent pour ne pas surcharger l'application d'information non appropriées en fonction des interactions effectuées par l'utilisateur.

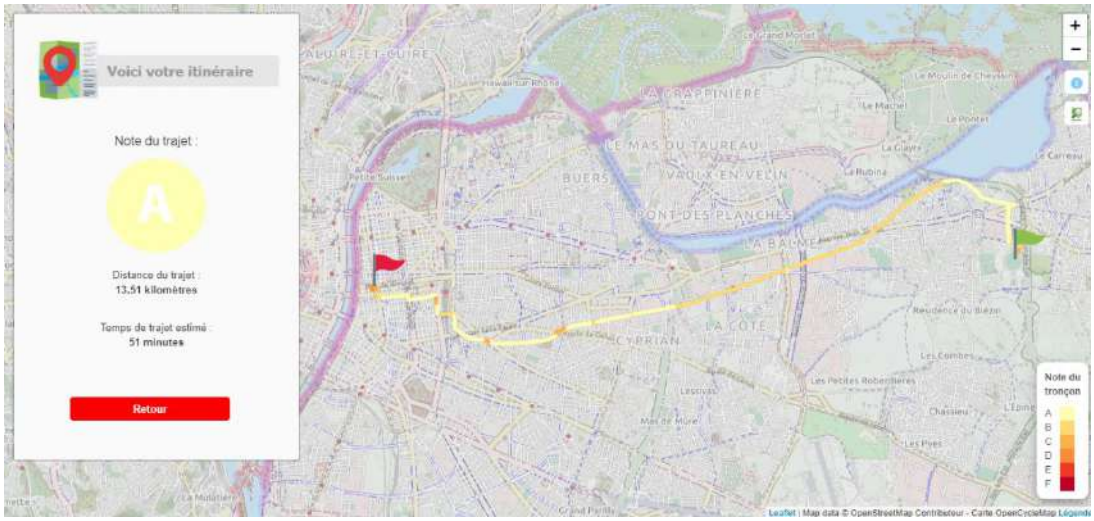

Figure 38 : Représentation de la légende

2) *Composantes principales*

La réflexion autour de la conception de notre application web a fait émerger deux lignes de conduite : une **interface ergonomique** et une **prise en main facilitée** des différentes facettes la composant. À la première connexion, l'utilisateur doit pouvoir comprendre les tenants et aboutissants de l'application, par le biais d'une fenêtre temporaire d'accueil. Celle-ci présentera les objectifs et le fonctionnement de l'application. La partie cartographique est alors grisée, en arrière-plan. En fermant cette fenêtre, l'utilisateur peut alors accéder à cette cartographie. À noter, l'utilisateur peut afficher à nouveau cette fenêtre informative en cliquant sur l'icône dédiée. L'interface principale est composée de quatre blocs d'éléments : le fond cartographique, la recherche d'itinéraires, les outils complémentaires et les attributions.

- Le fond cartographique, avec la bibliothèque Javascript libre *Leaflet*, qui supporte une cartographie interactive, avec affichage d'un fond *OpenStreetMap* (OSM), en l'occurrence la carte cyclable *OpenCycleMap*;

- Le module de recherche d'itinéraires, qui permet de saisir des adresses de départ et d'arrivée, puis de lancer une recherche d'itinéraires;

- Les outils complémentaires, à savoir : les contrôles cartographiques propres à *Leaflet* (zoom +/-), mais aussi deux boutons pour activer la contribution au sein d'*OpenStreetMap* et afficher la fenêtre informative sur l'application;

- Les attributions pour créditer la licence *OpenStreetMap* ainsi que spécifier la légende du fond cartographique employé.

Une fois la sélection d'un itinéraire faite par localisation ou adresse, celui-ci est calculé et affiché sur la carte. Une note qualitative lui est attribué tandis que son niveau de dangerosité est graphiquement détaillé par tronçons.

La contribution à OSM s'active par le bouton dédié, affichant alors une sélection d'objets dont les attributs peuvent être complétés. Une fenêtre de connexion est affichée pour saisir ses identifiants OSM. Une fois connecté, l'utilisateur pourra mettre à jour certains champs avec des valeurs prédéfinies. Il a ensuite la possibilité d'enregistrer (déconnexion et retour à la fenêtre de connexion) ou d'annuler ses modifications. Dans ce dernier cas, une pop-up s'affiche lui demandant de confirmer son abandon de l'édition en cours.

Nous présentons maintenant quelques extraits de scripts de fonctionnalités primordiales au bon fonctionnement de l'application, synthétisés de manière à saisir la façon dont nous nous sommes approprié les scripts et de fait les questionnements et problèmes qui ont pu en découler. Ces scripts sont à mettre en parallèle avec les captures d'écran présentes dans la partie visualisation page 43.

3) *Composantes spécifiques*

i. *Recherche de localisation (adresses de départ et d'arrivée)*

Leaflet propose un ensemble de plugins pour mettre en place un système de **géocodage** (recherche d'une localisation par adresse). Nous avons examiné leurs caractéristiques particulières pour retenir celui qui conviendrait le mieux à notre application. Une première sélection s'est établie sur le type de **fournisseur d'adresses** en place, privilégiant notamment la base d'adresses *Nominatim*. Celle-ci est aussi un outil qui permet de rechercher dans la base de données OSM des lieux et adresses. Trois

plugins l'utilisant ont retenu notre attention : *Leaflet GeoSearch*, *Leaflet Control OSM Geocoder* et *Leaflet Control Geocoder*. Au vu de leur fonctionnement intrinsèque et des démonstrations qui en sont faites, nous avons retenu *Leaflet Control Geocoder*, celui-ci permettant une récupération facilitée des coordonnées, ainsi qu'un module de géocodage inversé (recherche d'une adresse par localisation).

Le fonctionnement du plugin se fait en deux temps. Il faut d'abord instancier l'API permettant d'interroger la base d'adresses (ici *Nominatim*). Nous définissons une "viewbox" afin de restreindre la zone de requête d'adresses à l'espace du Grand Lyon (*geocodingQueryParams*).

```
//Geocoding queries service using Nominatim with spatial result limitation (viewbox
southwest/northeast)
   var geocoder = new L.Control.Geocoder.Nominatim({
      geocodingQueryParams: {
         "viewbox": "4.714508,45.5497,5.07019,45.95473",
         "bounded": "1"
      }
   });
```

Figure 39 : Instanciation d'une base d'adresses

Ensuite, il faut créer un geocoder, qui fonctionne comme un contrôle *Leaflet* classique, permettant une interaction pour l'utilisateur. Cela ajoute à la carte une barre de recherche pour saisir une adresse (ou des coordonnées lat/lng), que nous pouvons personnaliser (messages informatifs, marqueur *markgeocode* par défaut ou icône personnalisée).

```
//Search geocoding: starting point geocoder control + parsing coordinates to get lat/lng with
GeocodingResult
var geocoderDep = L.Control.geocoder({
  collapsed: false,
  position: "topleft",
  placeholder: "Votre adresse de départ",
  errorMessage: "Aucun résultat trouvé :(",
  geocoder: geocoder,
  defaultMarkGeocode: false
})
```

Figure 40 : Création d'un contrôle geocoder (simplifié)

Ensuite, une fonction est appelée pour interroger les éléments relevant de ce *markgeocode* (“.on”). Nous pouvons notamment obtenir les coordonnées du centre de la location (“event.geocode.center”), que l’on peut parser ensuite pour récupérer dans des variables séparées les coordonnées lat et lng. Ces dernières sont appelées ensuite pour la construction de la requête permettant d’initier un itinéraire (“xdep” et “ydep” pour le point de départ).

```
.on("markgeocode", function(event) {
 var center = event.geocode.center;
 var y1 = center
     .toString()
     .substring(
      center.toString().lastIndexOf("(") + 1,
      center.toString().lastIndexOf(",")
  );
 var x1 = center
     .toString()
     .substring(
      center.toString().lastIndexOf(",") + 1,
      center.toString().lastIndexOf(")")
  )
 xdep = x1;
              ydep              =              y1;
 })
```

Figure 41 : Récupération d’attributs du geocoder (simplifié)

Ce geocoder est dupliqué afin d’avoir deux champs de recherche d’adresses, pour les points de départ et d’arrivée.
Ces deux paires de coordonnées servent ensuite à l’élaboration d’un itinéraire. L’envoi de ces coordonnées se fait au format **json** (“dataType”) vers l’API, paramétré par une url (“url”, qui servira à déterminer le type d’itinéraire sur le serveur Go), avec un type de méthode (“POST”) et la donnée du type d’itinéraire (“data”, correspondant à la combinaison des coordonnées et des éventuelles valeurs de critères (type personnalisé uniquement).
La donnée de l’itinéraire calculé est retournée au format **geoJSON** dans une variable “itineraire” préalablement définie, puis affichée sur la carte.

```
var itineraire = L.geoJSON().addTo(map);
    $.ajax({
    url : url,
    type : 'POST',
    data : dataAPI,
    dataType : 'json',
    success : function(json, statut){ // success est toujours en place, bien sûr !
       itineraire.clearLayers()
       itineraire.addData(json)
       map.fitBounds(itineraire.getBounds())
    }
  });
```

Figure 42 : Requête pour l’échange de données pour concevoir un itinéraire (simplifié)

L'intérêt de ce plugin est de pouvoir aller plus loin avec les différentes fonctionnalités qu'il propose, en les associant aux informations géographiques. Ainsi, nous avons pu **augmenter de façon notable l'usage basique** qu'il en est prévu.

- Une mise à jour des champs textuels est implémentée au cas où une adresse ne serait saisie que partiellement (le nom "usuel" d'un lieu par exemple serait remplacé par son adresse complète, avec troncation à la longueur du champ de saisie).
- Les marqueurs peuvent désormais être déplacés (activation par "draggable : true"). Pour cela, une fois qu'une adresse est saisie dans un champ et un marqueur placé sur la carte, l'utilisateur peut déplacer celui-ci (par glisser-déposer) sur un autre lieu. Les nouvelles adresses et coordonnées sont récupérées et mises à jour automatiquement. Nous penchons ici vers une semi-fonctionnalité de géocodage inversé, dans la mesure où l'utilisateur peut déplacer un marqueur à sa guise mais n'a pas encore la main sur son placement initial sur la carte.
- Reprenant la logique de limitation de l'emprise spatiale de requête, nous souhaitions l'adapter dans notre instance *Nominatim* avec une duplication de la "viewbox" dans un *reverseQueryParams*, mais la fonctionnalité semble connaître quelque instabilité, nous résignant à définir une "bounding box" générale.
- Poussant la réactivité de l'application un peu plus loin, nous retravaillons le calcul d'un itinéraire de manière à ce que celui-ci se relance automatiquement dès que la position d'un marqueur est modifiée sur la carte, sans avoir à revenir sur notre choix des itinéraires pour le faire.

```
var markerDep = new L.marker(center, {
    draggable: true
  })
    .addTo(markerDepGrp)
    .on("dragend", function(e) {
      var latlng = e.target.getLatLng();
      if (inBbox(latlng, bbox)) {
        geocoder.reverse(
          latlng,
          map.options.crs.scale(map.getZoom()),
          function(addr) {
            if (addr.length > 0) {
              xdep = latlng.lng;
            ydep = latlng.lat;
            if (itineraire.getLayers().length > 0) {
              createItineraire();
            }
           } else {
            markerDepGrp.clearLayers();
            geocoderDep.setQuery("Aucune adresse trouvée");
            xdep = "";
            ydep = "";
           }
          }
}}
```

Figure 43 : Comportement lié au déplacement manuel d'un marqueur (simplifié)

Ces développements ont été rendus possibles en JavaScript par la réutilisation des fonctionnalités initiales du plugin, puis le recours à de la programmation interne pour gérer la récupération et le requêtage des données. Nous obtenons au final un service de recherche de localisation **fonctionnel, spatialement limité et optimisé** dans son affichage pour faciliter la compréhension pour l'utilisateur.

ii. Recherche d'adresse (sélection d'un lieu)

L'un de nos objectifs était de permettre un géocodage inverse, à savoir la recherche d'une adresse suite à la localisation d'un marqueur sur une carte. En ré-utilisant partiellement le plugin précédemment présenté, nous avons pu définir un système s'activant par un clic sur la carte et renvoyant une pop-up avec les informations d'adresses.
Cependant, au vu du développement atteint par nos outils de géocodage classiques et la possibilité de les rendre déplaçables sur la carte par l'utilisateur (avec mise à jour automatique), nous avons opté pour nous concentrer uniquement sur cette solution ; considérant également les temps d'implantation nécessaires.

iii. Navigation dans l'interface

Conformément à notre réflexion, nous mettons en place un panneau latéral sur la gauche de l'écran, qui contiendra les différents contrôles et informations nécessaires à l'application.
Le principe est de pouvoir afficher/cacher les éléments selon l'étape d'utilisation atteinte et proposer les différents contrôles ou résultats en conséquence.
Ces contrôles peuvent être des plugins Leaflet mais aussi des collections d'outils (*frameworks*), comme *Bootstrap*, une librairie proposant des composants prêts à déployer en HTML/CSS et JS.
De fait, ces contrôles sont instanciés dans des éléments HTML, et éventuellement complétés par une interaction en JS. Nous retrouvons donc les champs liés au géocodage, mais aussi des paramètres pour calibrer l'itinéraire souhaité, à travers une liste déroulante de paramètres. Des boutons permettent la validation des paramètres et de naviguer entre les différents panneaux.

```
var aide_revetement = L.easyButton('<img src="img/aide.png" style="width:16px">', function(){
   $("#myModalAide").modal("show");
}).addTo(map);
```

Figure 44 : Exemple d'un bouton personnalisé avec une image et appelant une fonction (ici, affichage d'une fenêtre d'information au clic sur le bouton) (simplifié)

Ces boutons sont très utiles pour l'affichage des éléments, avec le recours à la librairie JS jQuery, qui permet entre autres de manipuler de façon transversale des éléments HTML. Les attributs "show" et "hide" dissimulent ou non les éléments auxquels ils sont associés.

```
var configInit = L.easyButton('<strong style="color: white">Retour</strong>', function(){
   $("#BGDep").show();
   $("#BGArr").show();
   $("#dropdownMenu1").show();
   $("#resnote").hide();
   $("#restime").hide();
   $("#Legende").hide();
})
```

Figure 45 : Exemple d'un bouton qui va afficher les paramètres initiaux ("BGDep", "BGArr" et "dropdownMenu1") et cacher les éléments concernant des résultats d'itinéraire ("resnote", "restime" et "Legende") (simplifié).

Le résultat final présente une interface qui se veut complète et interactive, par l'apparition des différentes fonctionnalités lorsqu'elles sont souhaitées, sans surcharger d'information l'utilisateur. À l'instar de la première fenêtre informative d'accueil, d'autres sont ajoutées lorsque cela est jugé nécessaire pour expliciter des notions peu communes, avec le plugin Modal (dans la partie HTML et affichage selon les boutons cliqués).

Le paramétrage de l'itinéraire par l'utilisateur par le biais d'une liste déroulant fait appel au composant Dropdowns de Bootstrap, définissant dans le fichier HTML un bouton faisant dérouler une liste d'autant d'options que nécessaire. Ces dernières correspondent à nos fonctions plpgsql de calcul d'itinéraires qui, nous le rappelons, calculent plusieurs coûts de trajets en fonction des paramètres choisis.

```
<ul
  class="dropdown-menu type-itineraire" aria-labelledby="dropdownMenu1">
   <li id="TYPEsecurise"><a href="#">Trajet recommandé OSCO 👷</a></li>
   <li id="TYPEcourt"> <a href="#">Trajet le plus court ⚡</a></li>
   <li id="TYPEamenage"><a href="#">Trajet le plus aménagé 🚲</a></li>
   <li role="separator" class="divider"></li>
   <li id="personnalisation"><a href="#">Personnaliser mon trajet ✏</a>
     </li>
</ul>
```

Figure 46 : Liste d'options du menu déroulant (simplifié)

En fonction de l'option choisie, celle-ci est affichée sur le bouton (optimisation de l'expérience utilisateur) et est directement connectée à une requête spécifique de l'API, faisant le lien entre la base de données et l'interface web.
Concernant l'un de nos itinéraires - personnalisé - qui diffère dans son élaboration des autres, nous lui avons adjoint trois sliders afin de permettre à l'utilisateur de choisir lui-même la valeur qu'il accorde aux paramètres. Chacune des valeurs des critères est ainsi stockée puis mobilisée dans l'élaboration de la requête pour l'API.

```
<div class="slider2" id="slider2">
   <label class="slider-label" for="SL2">Aménagements cyclables</label>
     <p>
       <input
        id="SL2"
        data-slider-id="Slider2"
        type="text"
        data-slider-min="1"
        data-slider-max="5"
        data-slider-step="1"
        data-slider-value="50"
       /><br />
    </p>
</div>
```

Figure 47 : Création d'un slider de valeur 1 à 5, avec un pas de 1 (simplifié)

Concernant la génération des résultats, l'affichage de l'itinéraire vu précédemment est bien sûr la clé de voûte de l'application, associée à une discrétisation de l'indice de dangerosité par tronçons. La légende de l'indice est construite avec des seuils définis manuellement, ainsi qu'une palette de couleur en accord avec nos six classes, allant du jaune au rouge.

```
var legend = L.control({ position: "bottomright" });
legend.onAdd = function(map) {
  var div = L.DomUtil.create("div", "info legend"),
    grades = [0, 0.5, 0.8, 1.25, 2, 3],
    labels = ["A", "B", "C", "D", "E", "F"];
```

Figure 48 : Création d'un contrôle pour la légende avec les seuils (simplifié)

Des informations complètent l'expérience de l'utilisateur, afin de l'aider à mieux appréhender l'itinéraire qui lui est proposé. Ainsi, une note globale est attribuée à celui-ci, basée sur une moyenne des notes de chaque tronçon de l'itinéraire. Reprenant ensuite les mêmes seuils tels que définis dans notre légende, nous pouvons appeler dynamiquement une image, constituée d'une lettre de A à F dans un cercle de couleur identique à la légende.

```
    note = 0;
    json.features.forEach(feature => {
      note += feature.properties.ida;
    });
    note /= json.features.length;

document.getElementById("imgnote").src =
      "img/score_OSCO/" + getLettreNote(note) + ".png";
```

Figure 49 : Génération d'une note moyenne (lettre) et appel d'une image correspondant à cette lettre (simplifié)

De la même manière, nous proposons la distance du trajet en kilomètres et une estimation du temps de trajet. Cette dernière est calculée à partir de la longueur de l'itinéraire (distance) et d'une vitesse moyenne de déplacement à vélo (16 km/h). Le résultat est proposé en minutes.

Notons par ailleurs, que l'interface développée se veut la plus responsive possible (adaptation à la taille de l'écran).

iv. Gestion des erreurs

Pour prévenir d'éventuels arrêts dans le processus de l'application, ou des situations qui mèneraient l'utilisateur dans une impasse sans qu'il ne le sache et puisse en sortir, des messages d'erreurs sont planifiés à différentes étapes de l'application. Le but est donc de ne pas bloquer l'utilisateur d'une part, ni de laisser la possibilité à l'application de générer des informations incohérentes d'autre part.
Les potentielles erreurs recensées relèvent essentiellement de la bonne saisie des adresses, de la validité des fichiers géographiques ou de la connexion au serveur.

```
showMessage(
  " ✕ Erreur",
  "Le marqueur est placé en dehors de la zone autorisée.",
  true
);
```

Figure 50 : Erreur sur le placement manuel d'un marqueur hors zone (simplifié)

```
if (!xdep || !xarr) {
  showMessage(
    " ✕ Erreur",
    "Veuillez renseigner correctement les adresses de départ et d'arrivée.",
    true
   );
  return;
}
```

Figure 51 : Erreur sur la recherche d'un itinéraire aux lieux de départ et d'arrivée identiques (simplifié)

4) Contribution à OSM

i. Concept

Au-delà de fournir l'itinéraire, OSCO permet la contribution à OSM sur certains tags définis, comme la surface (qualité des routes) ou la vitesse maximum des véhicules. Le concept est basé sur un **cercle de rétroaction positif**. Comme l'ensemble de l'analyse est basée sur OSM, contribuer pour améliorer la base de données devrait améliorer notre routage. Pour cela, l'utilisateur peut accéder à l'interface à l'aide de l'icône présent en haut à droite de l'interface.

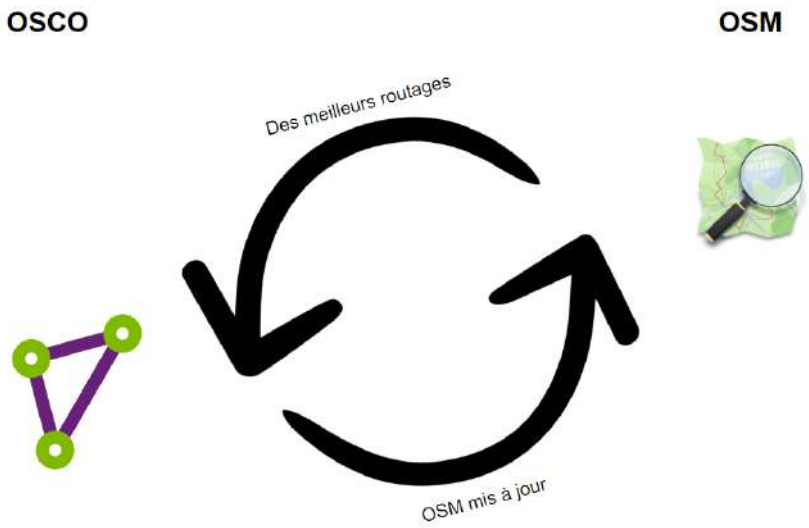

Figure 52 : Cercle de rétroaction positif OSCO

i. Mise en œuvre

1. Connexion à OpenStreetMap

La connexion à OpenStreetMap est permise selon deux méthodes : l'authentification HTTP et l'authentification OAuth. Leur principe diffère assez largement, l'authentification HTTP se repose sur un simple couple identifiant/mot de passe qui authentifie l'utilisateur. OAuth gère quant à lui une clé d'identification de l'application. Concrètement, cela permet à l'utilisateur d'allouer à l'application par laquelle il s'authentifie des permissions :

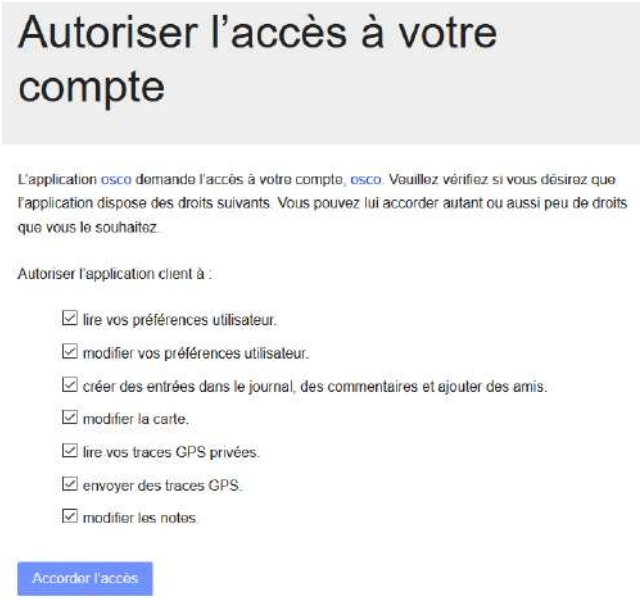

Figure 53 : fenêtre d'autorisation OSM faisant suite à une authentification OAuth

Actuellement, l'authentification HTTP est dépréciée au profit de l'authentification OAuth, qui pourrait devenir obligatoire dans le futur. Pour cette raison, nous avons choisi d'utiliser cette méthode.

La librairie Javascript osm-auth en propose une implémentation. Il s'agit de la librairie utilisée par l'éditeur en ligne iD. Des exemples sont de plus fournis sur la page du projet. La connexion est ainsi réalisée simplement par le code suivant :

```
auth = osmAuth({
   url: "https://osm.anatidaepho.be:3000",
   oauth_secret: "x5IZIEgVETkl1Yd9nHxgvS2RkAjor6nca9N8dVHN",
   oauth_consumer_key: "GVsqdgYgHP8NOERhsvCOLq6Wnx7XWXjynIHJFe0E"
})
auth.authenticate(callback);
```

Avec callback une fonction qui sera exécutée une fois la connexion effectuée.

L'URL correspond à l'adresse de l'instance OpenStreetMap. Les clés oauth_secret et oauth_consumer_key sont fournies par OpenStreetMap lors de l'enregistrement de l'application.

2. *Manipulation de l'API OpenStreetMap*

osm-auth fournit une méthode permettant d'envoyer des requêtes authentifiées à l'API. Cependant, la librairie ne propose pas de méthode encapsulant les requêtes OSM et il reste nécessaire de préciser l'URL de la requête et de construire l'OSM XML à envoyer à l'API. De plus, l'en-tête HTTP d'envoi par défaut d'osm-auth n'est pas correcte pour toutes les requêtes. Pour simplifier la procédure, nous avons développé une mini-librairie osmapi.js encapsulant osm-auth et implémentant les appels qui nous sont nécessaires. Ci-dessous un exemple de l'implémentation de l'ouverture d'un changeset :

```
this.createChangeset = function(comment, callback) {
   uri = "/api/0.6/changeset/create";
   xml =
    `
        <osm>
          <changeset>
            <tag k="created_by" v="` + this.tool + `"/>
            <tag k="comment" v="` + comment + `"/>
          </changeset>
        </osm>
    `;
   this._put(uri, xml, callback);
};
```

Nous avons également implémenté des méthodes pour manipuler plus aisément les fichiers OSM XML pour, par exemple, **supprimer un tag** :

```
this.removeWayTag = function(xml, key) {
   var tags = xml.documentElement.getElementsByTagName("tag");
   for (var i = 0; i < tags.length; i++) {
    if (tags[i].getAttribute("k") === key) {
      xml.getElementsByTagName("way")[0].removeChild(tags[i]);
      return;
    }
   }
};
```

Les requêtes à l'api se résument alors à l'appel des méthodes créées précédemment et simplifient grandement le code final :

```
api.createChangeset("Modification OSCO", function(err, xhr) { (...) } )
api.editWayTag(xml, "maxspeed", maxspeed.value);
api.updateWays(id_changeset, ways, function(err, xhr) { (...) } )
```

3. *Fonctionnement de la contribution*

Une fois connecté, l'API du serveur OSCO est requêtée pour récupérer 20 entités situées dans la zone affichée à l'écran (voir la partie sur la gestion de la requête pour la contribution page 36) et les placer sur la carte, avec une ligne et un marqueur cliquable par entité :

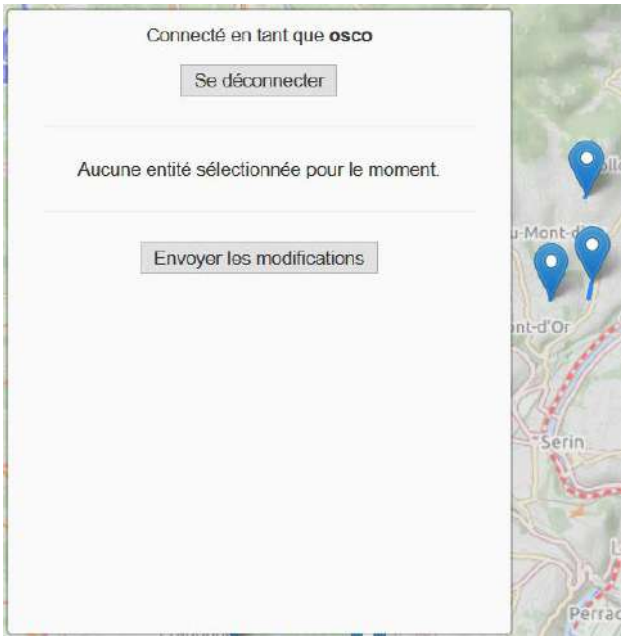

Figure 54 : Interface de contribution après connexion

Au clic sur le marqueur, une requête est effectuée sur l'API OpenStreetMap pour récupérer l'OSM XML lié à l'entité. Celui-ci est alors lu pour déterminer si les attributs de revêtement ou de vitesse sont encore vides (la donnée a pu être mise à jour côté OSM mais pas encore côté OSCO) et l'interface de contribution est alors générée dynamiquement et affichée dans le panneau gauche :

```
if (!api.getWayTag(xml, "surface")) {
   html += `<strong>Type de revêtement</strong><br/>
    <select id="surface" style="width: 200px">
     <option></option>
     <option value="asphalt">Asphalte</option>
     <option value="paved">Pavé</option>
     <option value="concrete">Béton</option>
     <option value="compacted">Matériaux compactés</option>
     <option value="ground">Pas de revêtement</option>
    </select><br/><br/>
   `;
 }
```

Exemple de génération de l'interface de contribution pour le tag de revêtement

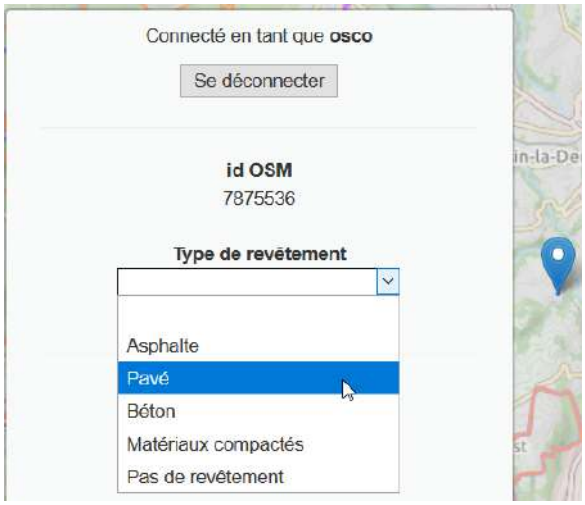

Figure 55 : Interface générée par le code précédent

L'interface limite la contribution à des valeurs prédéfinies pour limiter les erreurs d'entrée et simplifier la contribution à l'utilisateur.

4. Compte de test

L'inscription à notre instance d'OpenStreetMap nécessite une activation manuelle du compte côté serveur. Pour permettre **au lecteur de tester** cette fonctionnalité, nous avons mis en place un compte de test accessible par les identifiants suivants :

Login : geonum
Mot de passe : 20geonum20

Toutes les modifications effectuées par l'interface pourront être visualisées par la suite dans [l'historique des modifications d'OpenStreetMap](#) :

Figure 56 : Exemple de modifications réalisées par le compte geonum

VI. Évolutions envisageables

a) Évolutions envisageables à court terme

1) Intégration des données de trafic en temps réel

Une des idées intéressantes du projet était d'intégrer les données en temps réel du trafic produite par la Métropole de Lyon afin de proposer un itinéraire pondéré par ces valeurs. Après nous être procuré la donnée, nous avons cependant constaté que la complétude de celle-ci ne permettait pas de prendre en compte ce paramètre dans le calcul des itinéraires. Mais nous avons tout de même pu plancher sur la manière d'intégrer un flux en temps réel au sein de notre base de données PostgreSQL.

Pour l'intégration de ces données, nous nous sommes en premier lieu tournés vers l'utilisation d'une extension à PostgreSQL pour prendre en charge les flux WFS et pouvoir les traiter dans le calcul de l'indice.

Pour tenter l'import des données en temps réel, nous avons ajouté une nouvelle extension à notre base de données PostgreSQL : ogr_fdw. Cette extension nous permet d'ajouter des données provenant d'autres sources afin de pouvoir les traiter, en tant que table étrangère. Les données sont alors directement lues sur le serveur du Grand Lyon. Alors nous pouvons effectuer toutes les opérations nécessaires pour l'ajouter à l'indice de dangerosité.
Cependant nous constatons un très grand temps de latence, ce qui rend ces données inutilisables.

Nous pouvons également penser à une mise à jour régulière de l'état du trafic dans notre base, à l'aide d'une requête SQL exécutée à date fixe en mobilisant le "cron", ou avec une interrogation lors de la demande de calcul d'itinéraire.

Nous avons rencontré une erreur de certificat SSL lors de nos tests sous infrastructure Windows. Mais en passant par une infrastructure Debian, nous avons pu nous affranchir de ce problème de connexion.

À partir de cela, nous avons effectué une connexion au serveur distant par la commande suivante :

```
osco=# CREATE SERVER grandlyon FOREIGN DATA WRAPPER ogr_fdw OPTIONS( datasource
'WFS:https://jeremyk6...+@download.data.grandlyon.com/wff/rdata?SERVICE=WFS$VERSION=2.0
.0, format, format'WFS'):
```

Puis nous pouvons ajouter la table distante par la requête :

```
osco=# CREATE FOREIGN TABLE trafic_gdlyon_fdw (gid varchar, msGeometry geometry, etat
varchar, libelle varchar) server grandLyon OPTIONS (layer 'pvo_patrimoine_voirie.pvotrafic');
```

Nous avons alors effectué des tests de connexion à cette table étrangère. Le temps de requête avoisinant les 20 secondes est trop élevé. Une solution serait de générer un import toutes les 5 minutes par exemple.

```
Foreign Scan on trafic_gdlyon_fdw (cost=25.00..1025.00 rows=1000 width128) (actuel time
=374.277..13265.788 rows=3035 loops=1)
Planning time : 3019.136ms
Execution Time 16272.701 ms
(3 lignes)
```

Une optimisation de la connexion pourrait être envisagée lors de l'interrogation de la base en limitant les informations à rafraichir dans une boite englobante lorsque l'itinéraire est affiché à l'utilisateur.
Nous pouvons également envisager d'implémenter le même type de traitement avec les compteurs cyclables.

b) Évolutions envisageables à long terme

1) Développement d'un routage plus complet

Afin d'augmenter la qualité du routage, il pourrait être intéressant de proposer à l'utilisateur un récapitulatif des voies et directions à prendre au fur et à mesure du trajet, à la manière de *Google Maps*. Cela permettrait de rendre possible l'utilisation de l'application sur un smartphone. Pour cela, il faudrait également reprendre un peu le développement web pour qu'elle soit totalement responsive (adaptation des menus à la taille des écrans).

2) Permettre le départ depuis n'importe quel point de la carte

Une fonctionnalité que nous avons pensé à mettre en place dans l'application est le départ depuis n'importe quel point de la carte et non depuis le noeud le plus proche. Dans la pratique, nous avons déjà cherché la manière d'implémenter cette fonctionnalité. Mais par manque de temps, nous n'avons pas pu la rajouter. La fonction de calcul d'itinéraire à utiliser alors serait pgr_withPoints(). Il est alors nécessaire de créer un point temporaire (récupéré par le plugin Leaflet puis mis en forme en SQL). Le calcul de l'itinéraire part alors de ce point.
Il faut ensuite travailler l'affichage pour montrer l'itinéraire à partir de ce point (avec ST_LineSubstring par exemple).
Un tutoriel expliquant la démarche est disponible en ligne en espagnol :
https://www.unigis.es/rutas-optimas-entre-puntos-interes/

3) Perspectives de gamification

Pour mobiliser l'utilisateur et l'inciter à contribuer à OSM et donc à améliorer le routage effectué, l'application pourrait prendre en compte des profils et distribuer des points pour débloquer certaines fonctionnalités (des styles d'affichage, des niveaux et des badges/accomplissements par exemple). Également, le développement web permettant la contribution à OSM pourrait être amélioré, à la manière de l'application *StreetComplete*, pour rendre la complétion d'OSM plus ludique.

4) Présentation des itinéraires

Pour donner à l'utilisateur la possibilité de comparer la dangerosité des trajets, il nous semble intéressant de pouvoir afficher les différents types d'itinéraires à l'utilisateur en simultané. Pour optimiser l'affichage, nous pensons stocker les trajets précédemment demandés par l'utilisateur sur son navigateur internet afin de pouvoir les afficher grisés.
Toutes les informations étant stockées, nous pouvons penser afficher la note globale de l'itinéraire au survol au dessus du trajet grisé. L'itinéraire serait également plus détaillé sous la forme d'une feuille

de route décrivant les rues à emprunter, et la distance à parcourir sur les tronçons, entre le point de départ et le point d'arrivée.
Nous souhaitons également pouvoir visualiser la variable contribuant à rendre un tronçon dangereux. Par un clic sur un tronçon, apparaît la variable en cause et le descriptif (par exemple : "sur ce tronçon, la vitesse des véhicules est de 70 km/h"). On pourrait envisager d'y adjoindre des informations concernant les risques d'accident.

5) Optimisation des traitements

Un autre point de travail correspond à la réduction du temps de calcul de l'itinéraire. Certaines lenteurs sont à remarquer, lors de la demande d'itinéraire par l'utilisateur. Pour pallier à cela, nous avons pensé à optimiser les requêtes SQL, le code en Go et à changer l'infrastructure hardware pour augmenter ses performances. En effet, il ne faut pas perdre de vue que nous travaillons ici sur un Raspberry.

6) Augmentation du nombre de variables prises en compte

Au cours de notre enquête auprès de cyclistes, nous avons relevé plusieurs thématiques intéressantes à prendre en compte pour affiner notre algorithme. Nous pouvons reprendre les thématiques proposées par la littérature scientifique et par le questionnaire, à savoir : les intersections, la largeur des voies empruntées, la mauvaise visibilité, les accotements, le mauvais éclairage, la mauvaise signalisation.
Parmi les thématiques que nous pouvons envisager d'intégrer à moyen terme, nous pouvons relever :

- la pente, disponible en licence ouverte sur le data Grand Lyon : https://data.grandlyon.com/jeux-de-donnees/image-relief-2018-metropole-lyon-format-tiff/ressources
- la largeur des voiries, disponible en licence ouverte partiellement sur le data Grand Lyon : https://data.grandlyon.com/jeux-de-donnees/chaussees-trottoirs-metropole-lyon/donnees

L'idée est de faire d'OSCO non pas une simple application pour connaître le risque de circuler sur un itinéraire précis, mais bien d'en faire une application à utiliser pour tous ses trajets en vélo.

7) Mise en place de commentaires sur les trajets proposés

Il serait intéressant de proposer un espace où les utilisateurs peuvent directement nous faire remonter leurs remarques concernant l'application et/ou les itinéraires proposés. L'utilisateur pourrait renseigner un champ texte par un objet dans un formulaire de contact en ligne, avec une demande/remarque et son adresse mail pour être recontacté.
En outre, sous un onglet "Communauté", les cyclistes pourraient commenter les tronçons pour informer la communauté cycliste d'éventuels passages à risque voire dangereux.

8) Intégrer la notion de temporalité

Nous pourrions également prendre en compte la notion de temporalité. Celle-ci permettra de prendre en compte l'état des trafics routier (DataGrandLyon) et cycliste (données du trafic cycliste disponible ici : http://www.eco-public.com/ParcPublic/?id=3902 et https://data.grandlyon.com/jeux-de-donnees/comptage-velo-metropole-lyon/donnees), les accidents, les travaux et la météo en temps réel et ajustera en fonction l'itinéraire proposé.

9) Ajouter les autres types de mobilités

Dans le cadre de ce projet, nous n'avons pas pu ajouter d'autres types de mobilités dans nos analyses et donc proposer des itinéraires adaptés à chaque mode de transport doux.
Ainsi, avant de choisir le type d'itinéraire souhaité, l'utilisateur serait amené à sélectionner son mode de transport : gyropode, monoroue, trotinette, vélo. Par conséquent, plusieurs paramètres seraient à adapter en fonction du mode de déplacement :

- L'indice de dangerosité en fonction de chaque type de déplacement,
- Le calcul du temps de trajet en fonction de la vitesse du mode doux sélectionné.

VII. Compétences acquises

a) Compétences développées

Le projet géonumérique nous a permis à tous de valoriser les apprentissages théoriques de l'année et de développer des compétences principalement ordonnées en deux grands types : techniques et gestion de projet. Pour réaliser une simple synthèse sur les aspects techniques, nous avons souhaité résumer cette partie à l'aide de la figure suivante. Celle-ci reprend les principales thématiques sur lesquels nous avons pu acquérir des connaissances par rapport à nos acquis antérieurs.

b) Retours personnels

Figure 57 : Graphiques retours compétences

Bénédicte

" Pour moi, ce projet se caractérise principalement par le transfert des savoir-faire et compétences techniques. Chaque membre OSCO a su se porter dans le travail afin de transmettre ses connaissances pour enrichir celles de ses partenaires. Ainsi, en se soutenant et en se renforçant intellectuellement, le projet OSCO a pu être mené à bien autour d'objectifs communs : le partage de connaissance et l'ambition de proposer une application finie, fonctionnellement ambitieuse et fédératrice.

Techniquement, nous nous sommes répartis le travail et les tâches tout en assurant une continuité dans le travail. Promouvoir l'information, comme cela a été effectué lors de ce projet, s'est articulé autour de la construction d'une problématique, d'une méthodologie adaptée, de choix de visualisation graphiquement adaptés aux utilisateurs visés et de rédactions d'analyses rigoureuses pour présenter les différents points projets. Cette recherche m'a également entraînée à croiser/partager mes connaissances avec différents profils plus informatiques et ainsi m'a appris à tirer profit de cette

pluridisciplinarité pour mener à bien un projet. Enfin le projet OSCO m'a confirmé que les savoir-être sont les clés de la réussite d'un projet.**"**

Pierre

"Ce projet géonumérique est pour moi la plus grande entreprise que nous ayons eu à mener au cours de ces deux années d'enseignements universitaires, en termes de production de résultats et de travail collaboratif, avec un groupe déterminé ! Devant concevoir un projet de sa problématisation initiale à son développement applicatif, nous avons su trouver les ressources nécessaires pour le mener à bon port, par la concertation continue autour de nos objectifs. Ces derniers, nombreux devant l'ampleur des domaines techniques à couvrir, nous ont permis à tous de se positionner avec grande responsabilité sur des tâches à accomplir. Que cela soit dans notre zone de confort ou non, suivant nos profils initiaux, nous avons tous contribué à son élaboration, pour terminer par deux semaines à haute intensité.

Personnellement, je tiens à cette notion de "zone de confort", qui témoigne très bien de notre volonté de s'en extirper pour ne pas nous limiter dans nos envies. Nous n'avons pas hésité à nous saisir en profondeur de technologies plus ou moins explorées, par exemple les technologies web en ce qui me concerne. Dernière pièce du puzzle, l'interface web cartographique m'a permis de contribuer au développement d'une application, pour laquelle il m'a fallu communiquer autant sur les choix ergonomiques et solutions logicielles avec Bénédicte qui l'a réalisée avec moi qu'avec Jérémy, Victor et Thomas sur la bonne transmission des données entre le "front" et le "back".

C'est une très bonne expérience de travail, avec une grande qualité des discussions quant aux orientations à donner au projet, grâce à une équipe soudée tant pendant qu'en dehors des cours.**"**

Jérémy

" Issu d'un thème cher à toute l'équipe, dont le vélo est un moyen de transport régulier, OSCO nous a permis de réaliser intégralement un projet, de l'idée lancée au cours d'un repas jusqu'au produit quasi-final. Cette expérience, à mi-chemin entre la réalité du monde professionnel et le hackathon, nous a tous permis de monter en compétence sur des thématiques variées : de la gestion de projet à la technique, de la réflexion UX à la représentation cartographique. Elle fut également l'occasion de clore ces deux années de master au sein d'une équipe très équilibrée, dans les compétences et humainement. Une ouverture constante à l'autre et une écoute de chacun ont permis d'une part de travailler dans la meilleure ambiance possible, mais aussi de partager régulièrement avancées et savoirs en cours de route, le tout malgré une échéance courte qui peut être vectrice de stress. Sur un plan à la fois plus technique et plus personnel, ce projet m'a permis d'entrer plus en profondeur dans le monde d'OpenStreetMap. Passionné depuis plusieurs années par le logiciel libre et la mouvance open data, comprendre plus en détail le fonctionnement technique et humain d'OSM fut une expérience très forte. En ce sens, je suis ravi de notre contribution à la communauté avec la libération d'OSCO qui, s'il ne reste qu'un projet étudiant, pourra je l'espère s'ouvrir à d'autres horizons ! **"**

Victor

" Ce projet m'a permis de développer des compétences en analyse et gestion de données puisque je me suis principalement concentré sur la création de la base de données et sur les fonctions d'itinéraire. Par la même occasion, j'ai pu apprendre beaucoup sur les technologies de routage, qui m'intéressent particulièrement.

A l'échelle du groupe, les rôles se sont répartis assez naturellement selon les capacités de chacun mais aussi selon les volontés d'apprendre de nouvelles choses. Avant cette année, je n'avais jamais travaillé sous la forme de workshop. Après celui de la semaine de l'anthropocène dans le cadre de notre cours de Smart City et le projet Géonum, j'apprécie assez cette façon de fonctionner. On apprend rapidement en peu de temps tout en développant un produit fini consultable par tous.

Je terminerai ces quelques mots par la plus-value des travaux de groupe pendant ce Master Géomatique mention Géographiques Numériques. Certes, le projet Géonum et l'atelier bureau

d'étude en sont les plus grands témoins, mais globalement le master s'organise autour d'un échange de compétences entre profil plus informatique, profil géographe, profil analyste de données. J'ai beaucoup appris grâce à cette façon de travailler et c'est finalement assez proche de la vision que je me faisais du "géomaticien". C'est à dire une haute compétence technique au service de thématiques transversales toutes plus intéressantes et prenantes les unes que les autres. **"**

Thomas

" Par ce projet, j'ai pu approfondir mes compétences en développement par la mise en place du serveur Go avec Jérémy. J'ai beaucoup apprécié les échanges que l'on a pu avoir dans le groupe et entre les groupes ce qui nous a permis d'élargir nos champs d'horizon. La clé de voûte de ce projet a été la communication. C'est par le partage que nous avons pu nous stimuler les uns les autres pour ce projet. L'adage est confirmé : seul on va plus vite, mais ensemble on va vraiment plus loin (et encore plus quand la bonne humeur est présente). **"**

Conclusion

Notre projet géonumérique s'achève par la réalisation d'une application de routage pour les néo-mobilités. OSCO propose d'obtenir des trajets plus sécurisés ou plus rapides, au choix de l'utilisateur ; un itinéraire personnalisé permettant même à tout un chacun de l'affiner selon ses propres attentes. Au-delà de proposer un trajet entre un point de départ et d'arrivée, OSCO se veut préventive, libre et fédératrice d'un réseau d'utilisateurs unis pour lutter contre la dangerosité quotidienne.

La mobilisation de données libres et le processus de collaboration mis en place par OSCO ont été choisis afin de favoriser l'accessibilité de l'information pour tous. Il s'agit d'une part de fédérer un réseau citoyen et également de parfaire les données OpenStreetMap : des données OSM de meilleure qualité fournissent des routages plus précis. Ce projet ouvre la voie à une coopération citoyenne afin de partager sur les méthodes utilisées pour constituer l'indice de dangerosité et sensibiliser ainsi la population à de nouvelles tendances émergentes de mobilité urbaine.

La procédure proposée n'est pas figée et tendra à évoluer pour être la plus exhaustive possible. Le but initial étant de proposer des itinéraires plus précis prenant en compte plus de facteurs de dangerosité, l'objectif est d'intégrer d'ici fin 2020 les données en temps réel des trafics routiers et cyclistes. Parmi les perspectives évoquées, l'intégration de nouvelles sources de données géographiques et l'optimisation des productions informatiques permet d'entrevoir encore de belles opportunités pour ce projet.

En filigrane de l'année universitaire et de l'ensemble des cours, ce projet nous a donc permis de mettre en commun l'intégralité des connaissances en cartographie, en géomatique et en informatique que nous avons cultivé ces deux dernières années, au service d'un projet d'intérêt général.

Bibliographie

À propos des mobilités :

Lévy, Jacques, et Michel Lussault. *Dictionnaire de la géographie et de l'espace des sociétés.* Éditions Belin, 2003.

Vincent-Geslin, Stéphanie, et Emmanuel Ravalet. « La mobilité dans tous ses états. Représentations, imaginaires et pratiques. Introduction du Dossier ». *SociologieS*, 2 novembre 2015. http://journals.openedition.org/sociologies/5134.

Géoconfluences - ENS de Lyon. « Mobilité ». Terme. Consulté le 10 novembre 2019. http://geoconfluences.ens-lyon.fr/glossaire/mobilite.

À propos de l'intégration législative :

Assemblée Nationale. Orientation des mobilités (LOM), Pub. L. No. 157 (2018). http://www.assemblee-nationale.fr/dyn/15/dossiers/loi_orientation_mobilites.

« Loi mobilités, lom, projet de loi d'orientation | Vie publique ». Consulté le 10 novembre 2019. https://www.vie-publique.fr/loi/20809-loi-mobilites-lom-projet-de-loi-dorientation.

« Loi_Mobilités_15_mesures_cles.pdf ». Consulté le 10 novembre 2019. https://www.ecologique-solidaire.gouv.fr/sites/default/files/18191_LOM_15_mesures_12P_Pour%20BAT.pdf.

À propos de l'informations géographique volontaire

Goodchild, Michael F. « Citizens as Sensors: The World of Volunteered Geography ». *GeoJournal* 69, nº 4 (1 août 2007): 211-21. https://doi.org/10.1007/s10708-007-9111-y.

À propos des engins de déplacements personnalisés :

Behrendt, Frauke. « Why Cycling Matters for Electric Mobility: Towards Diverse, Active and Sustainable ». *Mobilities* 13, nº 1 (13 juillet 2017): 64-80.

Brenac, Thierry. « Sécurité et nouvelles pratiques de l'espace public : le cas des trottinettes, skateboards et autres engins à roulettes », 1 janvier 2015. https://hal.archives-ouvertes.fr/hal-01213361.

Brutus, Stéphane, Roshan Javadian, et Alexandra Joelle Panaccio. « Cycling, Car, or Public Transit: A Study of Stress and Mood upon Arrival at Work ». *International Journal of Workplace Health Management*, 6 février 2017. https://doi.org/10.1108/IJWHM-10-2015-0059.

Bührmann, Sebastian. « Bicycles as Public-Individual Transport – European Developments ». Présenté à MEETBIKE - European Conference on Bicycle Transport and Networking, Dresde, 3 avril 2008. https://ecoplan.org/library/Meetbike_Buehrmann_PPT.pdf.

Fishman, Elliot. « Cycling as Transport ». *Transport Reviews* 36, n° 1 (18 décembre 2015): 1-8.

Fishman, Elliot, Simon Washington, et Narelle Haworth. « Understanding the Fear of Bicycle Riding in Australia ». *Journal of the Australasian College of Road Safety* 23, n° 3 (2012): 19-27.

Héran, Frédéric. « Les nouvelles formes de la mobilité : trottinettes électriques, hoverboards, bicyclettes électriques... » *Annales des Mines - Realites industrielles* Mai 2018, n° 2 (26 avril 2018): 36-40.

Hopkinson, P., et M. Wardman. « Evaluating the Demand for New Cycle Facilities ». *Transport Policy* 3, n° 4 (1 octobre 1996): 241-49. https://doi.org/10.1016/S0967-070X(96)00020-0.

Nikolaeva, Anna, Marco te Brömmelstroet, Rob Raven, et James Ranson. « Smart Cycling Futures: Charting a New Terrain and Moving towards a Research Agenda ». *Journal of Transport Geography* 79 (1 juillet 2019): 102486. https://doi.org/10.1016/j.jtrangeo.2019.102486.

Pucher, John, et Ralph Buehler. « Cycling towards a more sustainable transport future ». *Transport Reviews* 37, n° 6 (2017): 689-94.

Pucher, John, et Ralph Buehler. « Integrating Bicycling and Public Transport in North America ». *Journal of Public Transportation* 12, n° 3 (septembre 2009): 79-104. https://doi.org/10.5038/2375-0901.12.3.5.

Pucher, John, et Lewis Dijkstra. « Promoting Safe Walking and Cycling to Improve Public Health: Lessons From The Netherlands and Germany ». *American Journal of Public Health* 93, n° 9 (1 septembre 2003): 1509-16. https://doi.org/10.2105.

Rabaud, Mathieu, et Cyprien Richer. « (Micro)mobilités émergentes et intermodalités. L'irruption des « Engins de Déplacements Personnalisés » dans les chaînes de mobilité. », 2019. https://halshs.archives-ouvertes.fr/halshs-02318268.

À propos des algorithmes utilisés :

Dijkstra, E. W. « A Note on Two Problems in Connexion with Graphs ». *Numerische Mathematik* 1, n° 1 (1 décembre 1959): 269-71. https://doi.org/10.1007/BF01386390.

Distance de Levenshtein. Consulté le 26 février 2020. https://fr.wikipedia.org/wiki/Distance_de_Levenshtein

À propos des pratiques citoyennes :

Koussa, Victoria. « Vélo, trottinette, gyroroue : quelle alternative aux transports traditionnels ? » *Hashtag*. France Culture, 31 août 2018. https://www.franceculture.fr/emissions/hashtag/velo-trottinette-gyroroue-quelle-alternative-aux-transports-traditionnels.

Pueyo, Serge. « Municipales : cinq ans d'EELV à Grenoble, ça change quoi ? » *leparisien.fr*, 22 août 2019. http://www.leparisien.fr/elections/municipales/municipales-cinq-ans-d-eelv-a-grenoble-ca-change-quoi-22-08-2019-8137218.php.

Wendling, Ségolène. « Les déplacements doux sur la Communauté de Communes du Bassin de Lons-le-Saunier ». Rapport de stage, juillet 2011. http://veloquirit39000.fubicy.org/Sch%E9ma%20CCBL/m%E9moire%20final.pdf.

À propos des chiffres mobilisés :

Insee. « Partir de bon matin, à bicyclette... - Insee Première - 1629 », 17 janvier 2017. https://www.insee.fr/fr/statistiques/2557426.

« Bilan 2018 de la sécurité routière | Observatoire national interministériel de la sécurité routière », 18 septembre 2019. https://www.onisr.securite-routiere.interieur.gouv.fr/contenus/etat-de-l-insecurite-routiere/bilans-annuels-de-la-securite-routiere/bilan-2018-de-la-securite-routiere.

« Épidémiologie des accidents de vélo et stratégies de prévention pour les éviter », 14 mai 2019. https://www.santepubliquefrance.fr/docs/epidemiologie-des-accidents-de-velo-et-strategies-de-prevention-pour-les-eviter-synthese-bibliographique-en-france-et-dans-les-pays-de-developpem.

Rapports spécialisés :

Gailou, Yohan. « Évaluation multicritère et cartographies de la cyclabilité en milieu urbain ». Lyon, 2018.

Krenn, Patricia, Pekka Oja, et Sylvia Titze. « Development of a Bikeability Index to Assess the Bicycle-Friendliness of Urban Environments ». *Open Journal of Civil Engineering* 05 (1 janvier 2015): 451-59. https://doi.org/10.4236/ojce.2015.54045.

Wang, Y., C. K. Chau, W. Y. Ng, et T. M. Leung. « A Review on the Effects of Physical Built Environment Attributes on Enhancing Walking and Cycling Activity Levels within Residential Neighborhoods ». *Cities* 50 (1 février 2016): 1-15. https://doi.org/10.1016/j.cities.2015.08.004.

Winters, Meghan, Michael Brauer, Eleanor M Setton, et Kay Teschke. « Mapping Bikeability: A Spatial Tool to Support Sustainable Travel ». *Environment and Planning B: Planning and Design* 40, n° 5 (2013): 865-83. https://doi.org/10.1068/b38185.