Développement d'une application web pour les règles d'urbanisme

Eliot LAMBOROT

Janvier - Mai 2020

Résumé : Les décisionnaires du développement urbain ont aujourd'hui accès aux ressources des « villes intelligentes », qui peuvent être utilisées pour les aider dans leurs choix d'aménagement du territoire. Mais bien que les données soient facilement disponibles, les connaissances, notamment les règles urbaines, possèdent rarement une représentation informatique directement exploitable. Ce projet a pour but la création d'une application web de cartographie permettant d'avoir une meilleure vision de certaines règles d'urbanisme. Ainsi il nous sera possible de visualiser sur une carte les différents éléments qui respectent ou non les contraintes imposées par le PLU-H mais aussi d'ajouter ou modifier des éléments urbains afin de faire apparaître les nouvelles infractions aux règles que cela peut engendrer.

Mots-clés : ville intelligente, connaissances spatiales, règles urbaines, systèmes d'information géographique, intégration de données

1 Introduction

Ce projet s'inscrit dans le contexte de « ville intelligente », traduction de l'anglais Smart City, qui désigne une ville utilisant les technologies de l'information et de la communication pour améliorer la qualité des services urbains ou encore réduire ses coûts. Parmi le nombre grandissant de villes expérimentant ce concept, nous retrouvons en France le projet Lyon Smart City qui a notamment ouvert ses données au public. Cependant ces villes restent soumises aux nombreuses règles régissant le développement du territoire. Ces règles s'appliquent à plusieurs domaines (e.g, transport, construction,...) mais aussi à plusieurs niveaux administratifs. Par exemple au niveau national avec le code de l'urbanisme ou le code de l'environnement et au niveau communal ou intercommunal avec les Plans Locaux d'Urbanisme.

Actuellement, pour aider les décisionnaires, les règles urbaines sont visualisées sur des cartes déjà informatisées (analyse spatiale avec des systèmes d'information géographique, ou SIG). Mais ces cartes sont pour l'instant produites par des géomaticiens qui interprètent humainement des règles textuelles. Pour appliquer ces règles, il est nécessaire de disposer de données pertinentes (e.g., tracés des pistes cyclables, informations sur les bâtiments). Or ces données sont généralement stockées dans des sources hétérogènes (format, schéma et représentation des concepts différents) qu'il faut intégrer avant de pouvoir exploiter les règles urbaines. La formalisation et l'informatisation des règles (au sens « modélisation ») assisterait et aiderait à automatiser cette production ainsi que les vérifications qui en découlent[2]. Elle rendrait aussi possible l'inférence de nouvelles règles.

Ce POM est donc une première tentative pour mieux cerner cette problématique de formalisation et d'informatisation avec des cas concrètement implémentés. L'objectif final est de développer un prototype (application web) centré sur la métropole du Grand Lyon et qui permet de vérifier un ensemble de règles pré-définies. La réalisation de cette application s'est déroulée de février à mai 2020 sous la direction des professeurs Franck Favetta et Fabien Duchateau de l'équipe Base de Données du LIRIS qui travaillent déjà sur cette thématique[1]. La première étape du projet fut l'identification de règles pertinentes, qu'il est possible d'implémenter sur une application cartographique et pour lesquelles nous avons les données nécessaires. Vient ensuite l'étape de formalisation de la règle afin de pouvoir l'exploiter facilement. Troisièmement, il faut collecter et agréger les données nécessaires à l'application d'une règle. Enfin il faut implémenter cette règle dans l'application.

Dans la suite de ce rapport, nous décrivons les outils utilisés pendant le projet. Puis les fonctionnalités liées aux règles urbaines sont détaillées (description, intégration de données et implémentation). Enfin, la conclusion présente des perspectives d'amélioration sur le travail réalisé ainsi qu'un retour d'expérience.

2 Outils utilisés

Le besoin de traiter un assez grand nombre de données nous oblige à utiliser un SGBD. C'est PostgreSQL, un un système de gestion de base de données relationnelle et objet open source, qui a été choisi. Ce choix et motivé la présence de l'extension PostGIS qui nous permet de traiter facilement et efficacement les données spatiales qui sont les principales données que nous utiliserons. Notre application contient un serveur web qui utilise le framework Python Flask pour sa légèreté et sa simplicité. Le client web utilise quant à lui la bibliothèque Leaflet pour l'aspect cartographique et Bootstrap 4 pour le style. Les fonds de carte sont fournies par OpenStreetMap.

Ces outils nous serviront à implémenter les différentes règles urbaines puis à développer les fonctionnalités basées sur celles-ci. Nous décrirons en détails ces fonctionnalités dans la partie suivante.

3 Fonctionnalités basées sur les règles urbaines

Parmi les nombreuses règles qui figurent dans les différents règlements, nous nous sommes concentrés sur quatre règles qui nous semblaient implémentables et pour lesquelles nous étions sûrs d'avoir les données disponibles. Ces différentes règles seront présentées dans les sous-sections suivantes, de leur description jusqu'à leur implémentation sur la métropole du Grand Lyon. Dans les paragraphes suivants nous serons amenés à utiliser des variables pour représenter des éléments urbains. Si ces éléments ne sont pas explicitement redéfinis nous supposerons que B, B1, B2 représentent des bâtiments, Z une zone du PLU-H et T un terrain constructible de la métropole de Lyon.

3.1 Croisements sans noeud

3.1.1 Description et objectif

Pour la première implémentation nous avons choisi la règle affirmant que deux tronçons (de route) ne doivent pas se croiser sans l'existence d'un nœud entre ces tronçons (i.e., correspondant à un carrefour ou une intersection).

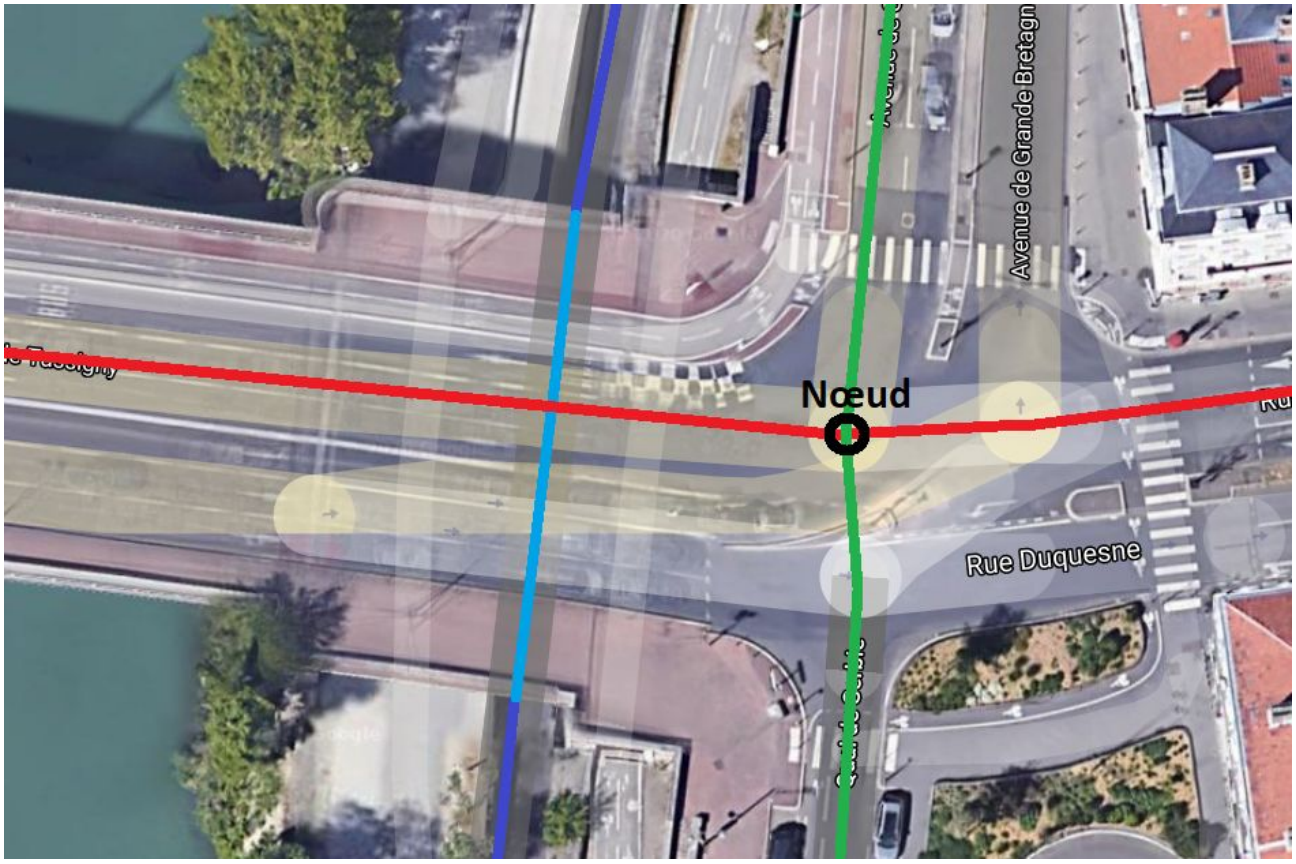


Figure 1. Vue aérienne exhibant différents cas d'intersections entre tronçons

La figure ci-dessus est un exemple tangible de cas où deux routes, ici rouge et bleue, peuvent se croiser sans l'existence d'un noeud placé à l'intersection. Ce cas est autorisé car les tronçons se croisent à une hauteur différente, la partie bleue claire étant situé en-dessous de la route rouge. A l'inverse, le croisement entre la route rouge et la route verte possède un noeud comme le demande la règle.

Nous avons choisi de commencer par cette règle car elle est simple à formaliser et nécessite peu de données. Soit T1 et T2 deux tronçons de la trame viaire, un prédicat *Intersects(X,Y)* dénotant l'existence d'une intersection entre X et Y, et un prédicat *Nœud(X,Y)* pour l'existence d'un nœud entre X et Y. On peut alors formaliser la règle sous la forme suivante :

Intersects (T1,T2) => Nœud (T1,T2)

L'objectif d'application de cette règle est de vérifier, pour tout tronçon qui en croise un autre, qu'il existe un nœud entre ces deux tronçons. Cela permet par exemple d'évaluer la qualité des données de la trame viaire.

3.1.2 Intégration des données

Pour appliquer la règle, nous allons avoir besoin des deux jeux de données représentant respectivement les tronçons et les noeuds de la trame viaire. Le réseau des voies de la métropole est représenté sous la forme d'un filaire de voies, correspondant à l'axe de la chaussée. Chaque voie est découpée en tronçons en fonction des intersections de voirie (noeuds). Les tronçons sont composés de données spatiales sous forme de ligne (suite de coordonnées géographiques) ainsi que d'un identifiant. Chaque nœud est lui représenté par la liste des identifiants de chaque tronçon lui appartenant sous la forme suivante : Idt_1 | Idt_2 | ... | Idt_n. L'intégration de données est ici limitée car les identifiant des tronçons passant par un noeud correspondent aux identifiants de tronçons.

3.1.3 Implémentation

La stratégie consiste à faire une jointure sur l'intersection de la géométrie de deux copies de la table des tronçons et ainsi détecter tous les tronçons s'inter-sectionnant. De ces tronçons nous récupérons les identifiants puis nous vérifions simplement dans la table des nœuds s'il en existe un nœud qui inclut les deux tronçons en question (i.e., soit ... | Idt_1 | ... | Idt_2 | ... soit ... | Idt_2 | ... |Idt_1 | ....).

Pour réaliser la jointure, la fonction st_intersects de PostGIS permet de vérifier si deux géométries possèdent une intersection. La vérification de l'existence d'un nœud se fait simplement avec l'opérateur SQL LIKE (comparaison approximative). Dès que l'on obtient la liste des tronçons ne respectant pas la règle, nous les envoyons sous forme de GeoJSON au client.

Le client s'occupe uniquement d'afficher les tronçons sur la carte Leaflet.

Pour la métropole de Lyon, les seuls tronçons que nous trouvons sont composés des trémies ou des tunnels, qui en effet recoupent d'autre tronçons lorsque que nous regardons en deux dimensions. Ces résultats étaient attendus et si nous possédions la valeur d'élévation de chaque tronçon, nous pourrions éliminer ces faux positifs. Sur Lyon, le jeu de données de la trame viaire est donc de bonne qualité.

3.2 Prescription sur la hauteur des façades

3.2.1 Description et objectif

Pour la deuxième implémentation, il s'agit toujours de vérifier l'application d'une règle assez simple. En effet, le PLU-H de la Métropole de Lyon réglemente la hauteur de façade pour les constructions. La ville et ses alentours sont ainsi divisés en secteurs géographiques, ayant chacune sa propre valeur de prescription de hauteur de construction. Le règlement établit que les constructions dans un secteur ne doivent pas posséder de façade d'une hauteur supérieure à la prescription du secteur. Supposons un secteur S et un bâtiment B, pour la formalisation nous aurons aussi besoin du prédicat *Contains(a,b)*, issu du modèle topologique DE-9IM[3][1], qui affirme que *a* contient *b*. Nous l'utilisons pour affirmer l'appartenance de B au secteur S. Nous avons aussi besoin de la valeur de hauteur de façade de B ainsi que la valeur de prescription de hauteur de Z. On peut alors formaliser la règle sous la forme suivante :

$$Contains(S,B) \Rightarrow B.hauteur \leq S.prescription$$

La règle s'applique également aux nouvelles constructions, mais ces dates n'étant généralement pas disponibles dans les jeux de données du Grand Lyon, notre objectif est de vérifier le respect de la règle pour tous les bâtiments de la métropole. Pour effectuer la vérification de cette règle nous utilisons deux jeux de données, décrits au paragraphe suivant, qui nous permettent de récupérer les différentes valeurs nécessaires.

3.2.2 Intégration des données

Le premier jeu contient la liste des bâtiments de la métropole sous forme de géographie de leur toiture ainsi qu'une liste de caractéristiques rattachées. De ces caractéristiques nous allons récupérer l'attribut de hauteur de façade. Le deuxième jeu contient les secteurs géographiques de prescription de hauteur décrite dans le PLU-H. Ici aussi ce sont des périmètres géographiques auxquels est attachée une valeur pour la hauteur maximale prescrite dans le secteur.

[1] Modèle topologique DE-9IM, https://en.wikipedia.org/wiki/DE-9IM

3.2.3 Implémentation

Afin de vérifier l'appartenance d'un bâtiment à un secteur, nous croisons les géométries des deux jeux pour faire correspondre chaque bâtiment avec son secteur de prescription. Une fois chaque bâtiment relié à son secteur, il nous suffit de comparer la hauteur de façade du bâtiment avec la hauteur prescrite dans son secteur. Nous vérifions comme cela le respect de la règle.

Pour implémenter cette stratégie nous faisons une jointure sur l'intersection (toujours avec la fonction st_intersects de PostGIS) des géométries des deux tables. Nous utilisons ensuite la commande SQL WHERE pour extraire les bâtiments respectant la condition affirmant que la hauteur de façade (+/- une marge) soit inférieure à la hauteur prescrite. Les données ainsi calculées par le serveur sont ensuite renvoyées au client.

A la réception de la liste des bâtiments, le client affiche les différentes géométries sur la carte avec une couleur représentant la différence entre la hauteur de façade et la prescription (cf. Figure 2). Par exemple, les bâtiments en rouge ne respectent pas du tout les prescriptions (sept mètres et plus) tandis que ceux en vert font une légère entorse au règlement (moins d'un mètre).

Sur l'ensemble de la métropole, nous constatons que la majorité des bâtiments respectent la norme ou alors la dépassent de peu (moins d'un mètre).

Figure 2. Carte représentative des bâtiments qui respectent ou dépassent leur prescription de hauteur

3.3 Création de pistes cyclables entre deux noeuds

3.3.1 Description et objectif

La création de voie cyclable est devenue un enjeu majeur des stratégies d'urbanisation actuelles. En effet le nombre de pistes cyclables ne fait qu'augmenter dans les grandes villes et il devient important d'établir une stratégie pour leur création. Nous avons donc regardé si nous pouvions utiliser les données offertes par le Grand Lyon pour aider à la décision d'aménagement de ces pistes.

Le but ici est d'aider un urbaniste qui souhaiterait créer une piste entre deux points. Nous souhaitons donc pouvoir choisir deux nœuds de la trame et calculer les pistes cyclables intéressantes qui pourraient être construites entre ces deux points. Les deux paramètres que nous avons jugés pertinents pour le trajet d'une nouvelle piste sont sa longueur et la réutilisation de pistes existantes.

3.3.3 Intégration des données

Nous possédons plusieurs jeux de données qui peuvent nous être utiles en fonction de la stratégie. Le premier jeu qui nous intéresse est celui des nœuds de la trame viaire. Ce jeu contient les coordonnées des croisements de routes ou chemins qui seront les points de départ et d'arrivée de nos futures pistes cyclables. Le deuxième jeu de données contient les tronçons de la trame viaire que nous avons déjà utilisés (cf. 3.1.2). Ce sont ces tronçons qui forment la future piste. Le dernier jeu que nous utilisons représente la géométrie des tronçons cyclables déjà présents sur la métropole. C'est avec ce jeu que nous pourrons prioriser les nouvelles pistes utilisant des anciennes. Par contre, les tronçons de route et de pistes n'ont aucun lien entre eux, et il n'est pas possible de connaître directement les pistes qui sont sur ou à côté d'une route. Pire, les géométries des routes et des pistes ne se superposent pas, et nous décrivons dans la partie suivante comment résoudre ce problème d'hétérogénéité.

3.3.4 Implémentation

Le premier point consiste à calculer les plus courts chemin entre les deux noeuds sous la forme d'une liste de tronçons. Ensuite nous comparons ces tronçons avec ceux des pistes existantes pour calculer, pour chaque plus court chemin, quelle portion est déjà cyclable. La fonctionnalité nous permet donc de choisir deux noeuds de la trame viaire et de calculer les K plus courts chemins en favorisant ou non la réutilisation de voie cyclable.

L'implémentation de cette stratégie se fait d'abord par la création d'une table temporaire contenant les K plus courts chemins entre les deux noeuds choisis. Pour cela nous utilisons l'algorithme KSP de Postgis (une implémentation de l'algorithme de Yen). Cependant un problème survient lorsqu'il nous faut vérifier si les différents tronçons comportent une bande cyclable. En effet les géométries des deux jeux sont assez hétérogènes, i.e. pour une même route la géométrie des tronçons cyclables et non-cyclable sont différents et nous ne possédons aucun identificateur qui pourrait les mettre en relation.

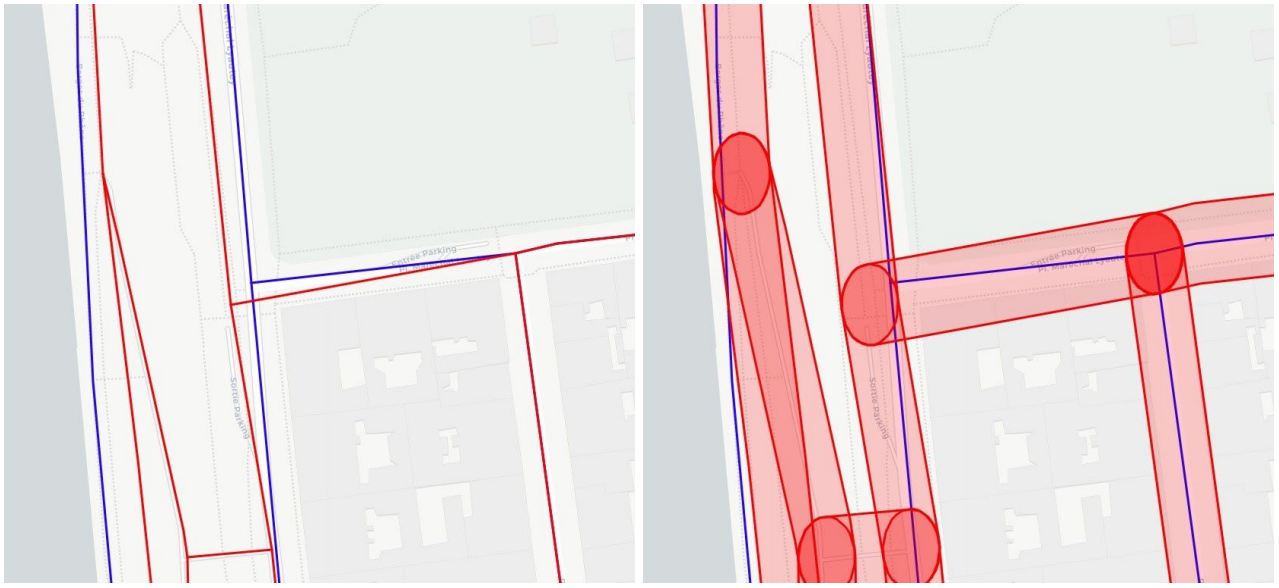
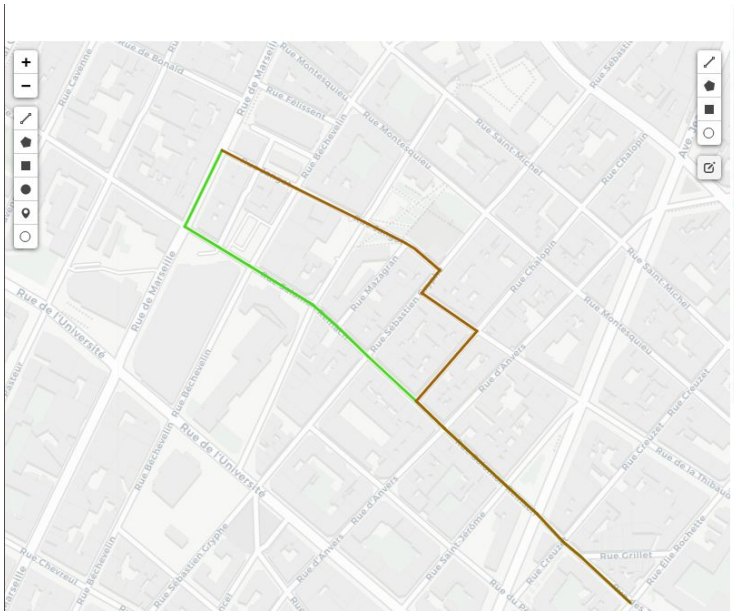


Figure 3. Superposition des tronçons de rue (bleus) et des tronçons de piste (rouges). À gauche, sans le buffer (peu de superposition) et à droite, avec augmentation artificielle de la taille des tronçons cyclables

L'exemple de la figure 3 (gauche) montre les tronçons cyclables en rouge décalés par rapport aux tronçons de rues en bleu. Pour palier à ce problème, nous augmentons (virtuellement) la taille des tronçons cyclables (opération spatiale de *buffer*) afin qu'ils englobent les tronçons de rues comme sur l'exemple (figure 3, droite) et ainsi nous pouvons faire une jointure entre les deux géométries avec la fonction st_contains de PostGIS.

Une fois les différents tronçons reçus, le client web s'occupe de trier chaque tronçon en fonction de l'ID du chemin et affiche chaque chemin sur la carte ainsi que le tableau comparatif de longueur et de pourcentage de piste réutilisée. La figure 4 illustre le résultat d'une recherche d'aménagement entre la rue Elie Rochette et la rue de Marseille. Seules deux propositions sont affichées : le trajet numéro 7 (orange), qui mesure 673 mètres et réutilise 83% de pistes existantes, et le trajet numéro 0 (vert) qui est plus court mais ne réutilise que 70% de pistes.



Longueur	Pourcentage deja piste	N°	Visible
673	83	7	☑
684	83	11	☐
696	83	8	☐
703	83	9	☐
635	82	3	☐
662	82	6	☐
663	82	5	☐
618	81	1	☐
628	81	2	☐
615	70	0	☑
658	68	4	☐
693	61	10	☐

Figure 4. Interface affichant les propositions de pistes cyclables

3.4 Création d'un projet de construction

3.4.1 Description et objectif

Dans le but de vérifier la conformité d'un nouveau projet de construction, nous avons pensé à une fonctionnalité permettant à l'utilisateur de l'aider dans la création d'un bâtiment. Il existe plusieurs règles dans le PLU-H pour la création de nouveau bâtiment, nous en avons déjà vu une concernant la prescription de hauteur de façade, qui pourra être réutilisée ici. En complément, nous avons choisi un échantillon de quatre règles à implémenter pour cette fonctionnalité, que nous détaillons ci-après :

A. La surface maximale du bâtiment dépend de sa fonction

Soit un bâtiment B que l'on souhaite construire. Avec le prédicat *Contains* (cf. 3.2.1) et la propriété Z.surfaceMax(F) qui correspond à la surface plafond d'un bâtiment de fonction F (e.g. habitation, bureau, commerce) pour la zone Z, nous pouvons donner la formalisation suivante pour la règle :

$$Contains(Z,B) \Rightarrow B.surface \leq Z.surfaceMax(B.fonction)$$

B. Certaines fonctions de bâtiments sont interdites

Chaque zone possède une liste de fonctions de bâtiments qui sont interdites à la construction. Avec cette propriété nous pouvons simplement formaliser la règle ci-dessus telle que :

$$Contains(Z,B) \Rightarrow B.fonction \notin Z.interdits$$

C. Les bâtiments doivent être construits dans la Bande de Constructibilité Principale (BCP)

La BCP représente une bande sur la bordure de chaque terrain. C'est dans cette bande que les bâtiments doivent être construits et sa taille dépend du type de zone. On suppose que *BCP(Z)* est une propriété de chaque terrain qui représente la géométrie de sa BCP en fonction de la zone Z. Il nous faut aussi une variable T qui représente un terrain. Avec cela nous pouvons alors formuler la règle sous la forme suivante :

$$Contains(Z,T) \wedge Contains(T,B) \Rightarrow Contains(T.BCP(Z),B)$$

D. Une distance entre les constructions sur un même terrain doit être respectée.

SI plusieurs constructions sont présentes sur un même terrain, alors elles doivent respecter une distance entre elles qui dépend de la zone dans laquelle elles sont construites. Chaque zone possède une propriété *distanceMin* qui représente la distance minimale prescrite entre deux constructions sur un même terrain. On notera [B1, B2] la distance entre les bâtiments B1 et B2. Ceci nous permet de formuler la règle ainsi :

$$Contains(Z,T) \wedge Contains(T,B1) \wedge Contains(T,B2) \Rightarrow [B1, B2] \leq Z.distanceMin$$

E. La hauteur maximale de façade est prescrite.

Cette règle a déjà été décrite et formalisée dans la partie 3.2.1.

L'objectif est donc de permettre à l'utilisateur de tracer un nouveau bâtiment sur la carte et que l'application vérifie parmi le respect des cinq règles décrites.

3.4.3 Intégration des données

Les règles étant pour la plupart textuelles, nous aurons besoin des zones décrites par le PLU-H ainsi que les zones de prescriptions de hauteurs vue précédemment (c.f. 3.3.2). Le jeu de données des zones comporte des périmètres géographiques avec un attribut de type de zone. Nous aurons aussi besoin du jeu de données contenant la surface des terrains de la métropole.

3.4.4 Implémentation

Il faut d'abord que le client envoie les caractéristiques de la nouvelle construction au serveur (i.e. géométrie et attributs hauteur et fonction). Le serveur doit ensuite retrouver la zone associée puis en déduire quelles règles appliquer. On peut ensuite vérifier les règles et renvoyer au client le résultat.

Pour la règle A, il nous faut récupérer sa superficie qui est calculée par Leaflet ainsi que la fonction et la hauteur du bâtiment créé par l'utilisateur. Nous pouvons ensuite faire une jointure, avec st_contains, sur les géométries des zones et celle du nouveau bâtiment afin de trouver le type de zone associé à la nouvelle construction. La zone ayant un attribut de surface maximale nous pouvons vérifier si le bâtiment respecte la règle. Pour la règle B, même principe, après avoir trouvé le type de zone correspondant au bâtiment nous pouvons vérifier que la fonction du bâtiment est autorisée dans sa zone. Pour la règle C, la taille de la Bande de Constructibilité Principale est donnée par le type de zone. Pour vérifier si un bâtiment est construit dans la BCP nous allons avoir besoin de la géométrie de la BCP. Les données pour la BCP n'existant pas nous allons devoir les calculer à partir des données géographiques des limites de terrains. Pour cela nous calculons la différence entre la géométrie de la propriété et sa même géométrie mais réduite de X mètres, X étant la taille de la BCP. Une fois la BCP calculée il suffit de vérifier si sa géométrie contient le bâtiment. Pour vérifier si la distance entre deux constructions sur un même terrain est respectée (règle D), il nous faut dans un premier temps croiser les géométries des deux nouveaux bâtiments avec la géométrie des propriétés pour vérifier que les deux bâtiments sont bien sur la même propriété. Nous récupérons le type de zone de la propriété (même procédé que pour les autres règles) afin de pouvoir déduire la distance minimale prescrite entre les deux constructions. Enfin nous vérifions que les deux propriétés respectent la distance minimale. L'implémentation pour la prescription de hauteur de façade à déjà été décrite (c.f. 3.3), la seule différence est qu'il n'y a qu'un seul bâtiment dans ce cas précis.

Coté client la création du bâtiment se fait grâce au plugin Draw de Leaflet qui permet à l'utilisateur de créer des formes géométriques sur la carte à partir desquelles nous récupérons les données géographiques. On demande aussi à l'utilisateur de renseigner, via un formulaire, les différentes caractéristiques (hauteur, fonction du bâtiment) nécessaires à la vérification des règles.

D'autres règles auraient été intéressantes lors d'un projet de nouvelle construction, comme celle qui impose à un bâtiment de toucher certaines limites de la propriété. Par exemple, dans les zones UCe1 (zones urbaines, périphériques du centre ancien), il est précisé que "les constructions sont implantées sur les deux limites séparatives latérales". Cependant, l'implémentation de cette règle nécessite des informations précises sur les limites de propriétés (i.e., lesquelles sont frontales ou latérales) que nous ne possédons pas pour l'instant.

4 Conclusion

Malgré le développement récent des "smart cities", les règles urbaines qui régissent nos villes et territoires sont encore peu informatisées. L'**objectif de mon projet POM** consistait à développer une "proof of concept" afin de démontrer deux points. Premièrement mettre en évidence la faisabilité quant à la modélisation et à la représentation de ces règles urbaines et spatiales. Deuxièmement exploiter ces règles pour vérifier le respect des normes ou pour valider un nouveau projet d'aménagement. Notre application implémente quatres fonctionnalités basées sur ces règles, ce qui illustre la possibilité d'automatiser, au moins en partie, ces connaissances urbaines.

Bien que l'application développée soit concluante par rapport aux objectifs initiaux, plusieurs **perspectives d'amélioration** sont envisagées. Tout d'abord, les règles spatiales ont été codées en dur dans l'application, ce qui ne facilite pas leur partage et réutilisation. Une solution idéale serait d'être capable d'extraire ces règles directement à partir des documents textuels (e.g. PLU-H) dans lesquels elles apparaissent. Une représentation formelle de ces règles, comme mentionné dans [Laurini et al. 2017],

faciliterait leur exploitation ainsi que leur intégration. Une autre perspective concerne bien sûr l'ajout de nouvelles règles, ce qui implique, en plus de leur définition formelle, d'identifier les jeux de données pertinents et de résoudre les problèmes d'intégration pouvant survenir. L'exemple des tronçons de rue et de pistes qui ne sont ni reliés ni superposables montre à quel point ce verrou reste difficilement automatisable. Ce problème se pose d'autant plus si l'on souhaite mettre à disposition l'application pour d'autres villes. Enfin, des échanges avec des urbanistes et des décideurs territoriaux devraient faire émerger de nouveaux besoins applicatifs.

D'un **point de vue personnel**, j'ai apprécié participer à la création d'une application qui n'a pour l'instant pas ou peu d'équivalent, il faut innover et chaque problème rencontré est inédit. Ce fut aussi valorisant de travailler sur un problème moderne sous la direction de personnes impliquées sur la thématique de par leur travaux de recherche. Durant ce projet j'ai découvert la géomatique et j'ai aussi pu approfondir mes connaissances particulièrement dans le traitement des données. C'est ce domaine qui m'intéresse le plus dans ma poursuite d'étude et je ressors, de cette expérience, conforté dans mes choix.

5 Références

[1] Franck Favetta and Fabien Duchateau Detecting Conflicts and Recommending Actions for Urban Planning. Regional Knowledge workshop, Lyon, 2020

[2] R. Laurini and F. Favetta. About External Geographic Information and Knowledge in Smart Cities. Smart Data and Smart Cities, 42, 2017

[3] Clementini, E., Di Felice, P., Van Oosterom, P. : A Small Set of Formal Topological Relationships Suitable for the end-User Interaction, in proceedings of Third International symposium (SSD'93) Advances in Spatial Databases, ed. Abel, D. and Ooi, B. C., Lecture Notes in Computer Science, Springer Verlag, Singapore, 1993, pp. 277-295