



PROJET DÉVELOPPEMENT INFORMATIQUE  
PDI9 : PLUGIN QGIS DE RECONSTRUCTION 3D DE BÂTIMENT

---

## Rapport d'Analyse

---

***Élèves :***

MARÉCHAL Mathéo  
BREILLAD Arnaud  
BROSSARD Maud  
STEINMETZ Louis

***Commanditaire :***

MUGUET Thomas  
VINCENT Picavet  
BERTRAND Parpoil

## Table des matières

<b>1</b>	<b>Contexte</b>	<b>2</b>
1.1	Origine . . . . .	2
1.2	Présentation de l'équipe . . . . .	3
1.3	Enjeux . . . . .	3
<b>2</b>	<b>Objectif de l'étude et reformulation du besoin</b>	<b>3</b>
2.1	Les objectifs de l'étude . . . . .	3
2.2	Les contraintes . . . . .	4
2.3	Le recueil du besoin - les acteurs . . . . .	4
2.4	La réponse apportée au besoin . . . . .	4
<b>3</b>	<b>Analyse fonctionnelle - Solutions proposées</b>	<b>4</b>
<b>4</b>	<b>Étude technique : Choix des logiciels et langages - Architecture</b>	<b>6</b>
4.1	Les outils utilisés . . . . .	6
4.2	Fonctionnement de la reconstruction 3D . . . . .	8
4.3	Réalisation technique . . . . .	9
4.4	Problèmes rencontrés . . . . .	9
4.5	Résultats obtenus . . . . .	11
4.6	Choix du logo et du nom du plugin QGIS . . . . .	13
4.7	Perspectives d'évolutions . . . . .	13
<b>5</b>	<b>Réalisation et suivi de projet</b>	<b>14</b>
5.1	Les risques . . . . .	14
5.2	Planning prévisionnel . . . . .	16
5.2.1	Diagramme de Pert . . . . .	16
5.2.2	Diagramme de Gantt . . . . .	17
5.3	Organisation . . . . .	17
5.4	Retour sur les attentes du commanditaire . . . . .	18

# Introduction

Le programme national LiDAR HD a favorisé la production et la diffusion d'un grand jeu de données du sol et du sursol en France. Cela a permis d'accéder à une nouvelle dimension. La demande dans la thématique de la 3D a donc fortement augmenté et est maintenant la réponse apportée à divers besoins (collectivités, gestionnaires, acteurs privés, etc.). Oslandia cherche à répondre à cette demande en développant un outil open source sur QGIS permettant de reconstruire des bâtiments en 3D.

## 1 Contexte

### 1.1 Origine

Oslandia est une société spécialisée en architecture SIG. Ils fournissent des prestations de services autour de logiciels open source. Ils sont aussi experts des applications de webmapping au travers des outils *Leaflet* et *OpenLayers* et utilisent cette expertise dans des applications métier dans divers domaines (eau, assainissement, etc.).

Maintenant qu'il existe les données de LiDAR HD, Oslandia souhaiterait améliorer l'intégration de la 3D dans QGIS. En effet, la demande a augmenté subitement alors que les outils de QGIS ne sont pas encore suffisamment développés. Oslandia cherche donc de manière générale à améliorer l'intégration de la 3D au cœur de QGIS et par un plugin dans le cadre du projet développement informatique. Dans ce contexte, l'entreprise aimerait bien développer un nouveau plugin QGIS, permettant de faire une reconstruction 3D de bâtiments dans une zone urbaine dont l'étendue est indéfinie.

3D Bag est un visionneur web qui met à disposition une maquette 3D des Pays-Bas à partir de données produites par l'AHN (la carte numérique des altitudes aux Pays-Bas) et des données de BAG ; BAG étant le registre des bâtiments et des adresses le plus détaillé et disponible en open source. Pour visualiser ces bâtiments sur 3D Bag, il a fallu reconstruire un à un ces bâtiments, ce qui est possible grâce au logiciel nommé GeoFlow, qui a notamment été développé par la même équipe : le groupe de recherche sur la géoinformation de l'université de Delft. Leur travail est disponible sur ce lien Git : *geoflow-bundle*

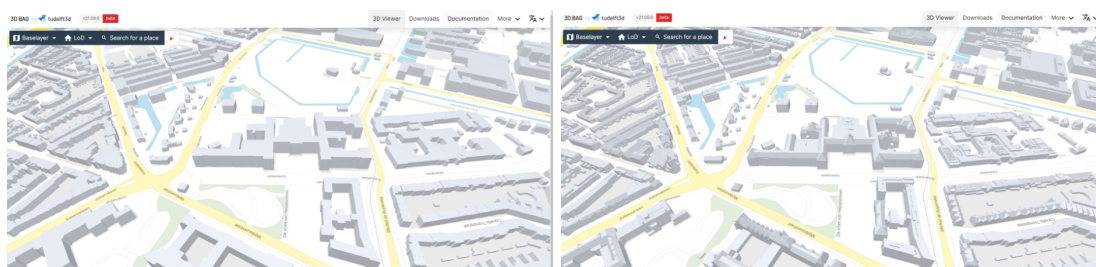


FIGURE 2 – Captures d'écran de 3D Bag aux différents niveaux de détails

Suite aux recherches menées par Oslandia dans le domaine de la reconstruction 3D, 3D Bag et GeoFlow a été identifié comme le duo le plus abouti en modélisation urbaine à partir de nuage de points. De plus, les logiciels et les données utilisées sont open source, ce qui est en adéquation avec les valeurs de l'entreprise. En outre, il convient de souligner que GeoFlow utilise

principalement la brique logicielle CGAL (bibliothèque logicielle de calcul géométrique), qui est également utilisée par Oslandia avec SFCGAL (extension de PostGIS basées sur CGAL). Cette similitude d'outils laisse la possibilité à Oslandia d'intégrer cette reconstruction 3D directement dans PostGIS à l'avenir.

Ainsi, l'enjeu est ici de récupérer le logiciel GeoFlow et de réussir à l'adapter afin qu'il soit fonctionnel avec nos propres sources de données. Et enfin, de créer un plugin QGIS à partir de ce code.

## 1.2 Présentation de l'équipe

Ce projet est commandité par Oslandia, par l'intermédiaire de Bertrand Parpoil, Manager de Projet qui est à l'origine de cette étude.

L'équipe qui développe ce projet est composée de 4 étudiants de l'ENSG : trois étudiants en ingénieur 2ème année (Louis Steinmetz, Mathéo Maréchal, Arnaud Breillad) et une étudiante en M1 Géomatique (Maud Brossard). Thomas Muguet, ingénieur SIG à Oslandia, nous supervise sur ce projet. Il nous apporte ses connaissances techniques et toutes les recherches qui ont été faites en amont.

Concernant l'apport de chacun, tous les étudiants ont des connaissances en matière de programmation, de plus, plusieurs étudiants possèdent des compétences particulières dans la manipulation de nuages de points LiDAR. Mathéo Maréchal sera le chef de projet.

## 1.3 Enjeux

Ce plugin 3D s'intègre avec d'autres projets d'Oslandia, néanmoins la demande n'est pas impérative. Si l'outil de reconstruction 3D est prêt à l'emploi, cela serait un avantage non négligeable pour l'entreprise. Il n'y a donc pas de conséquences négatives majeures.

# 2 Objectif de l'étude et reformulation du besoin

## 2.1 Les objectifs de l'étude

- **Les fonctionnalités** : La fonctionnalité principale qu'offrira notre application est la suivante : à partir des données de notre utilisateur en entrée (il faut bien évidemment que ces données soient compatibles avec notre programme), notre plugin sortira les bâtiments reconstruits en 3D.

- **Performance imposée** : Le commanditaire souhaiterait que les bâtiments sortis en 3D par notre logiciel aient un niveau de détail correspondant à un LoD (Level of Detail) 1.2, 1.3 ou celui d'un LoD 2.2. De plus, il souhaiterait que le plugin soit développé en open source et obligatoirement sur QGIS et qu'une documentation utilisateur explicite, claire et bien renseignée accompagne le plugin.

- **Interopérabilité** : Aucune interopérabilité n'est prévue dans ce contexte pour notre logiciel

- **Scénarios de diffusion de l'outil** : Le plugin étant développé en Open Source donc public, un scénario de diffusion de l'outil n'est pas prévu. Cependant, si à la fin du projet, le plugin est pleinement opérationnel, on peut supposer qu'il pourra servir à de nombreuses personnes dans le

cadre d'études urbaines par exemple (étude sur la pollution sonore, les réseaux routiers en zones urbaines, etc...). De plus, l'outil va être diffusé de manière interne au sein même d'Oslandia car le besoin provient en premier lieu de ses services.

## 2.2 Les contraintes

Peu de contraintes nous sont imposées par notre commanditaire et le cahier des charges est somme toute assez léger. Néanmoins, les contraintes majeures sont la création d'un jeu de données test (accessible sur GitLab), qui accompagnera le code open source de notre plugin QGIS (également disponible sur GitLab) ainsi qu'une documentation utilisateur bien renseignée et la plus claire possible afin qu'il soit le plus facilement compréhensible par tous les utilisateurs. Les contraintes imposées par l'ENSG sont le rapport d'analyse final et le document de présentation. Il faut néanmoins développer le plugin avant le rendu du projet qui sera le 17 mai 2023.

## 2.3 Le recueil du besoin - les acteurs

La demande pour des modèles 3D est en constante augmentation, notamment chez le grand public. De plus, au sein d'Oslandia, un de leurs services exprime le besoin d'avoir des modèles 3D à disposition. Dans ce contexte, notre projet a pour objectif de développer un plugin fonctionnel qui permettra de valider la faisabilité d'utiliser GeoFlow sur QGIS. Nous mettons donc l'accent sur la livraison d'un plugin fonctionnel dans les délais impartis. Dans ce cadre, nous avons décidé de ne pas réaliser d'enquêtes utilisateurs.

Cependant, les utilisateurs finaux seront tous les utilisateurs de QGIS qui auront besoin de ce plugin. Le code sera open source. Il est nécessaire de créer une documentation utilisateur précise et explicite qui aidera les utilisateurs quel que soit leur niveau dans l'utilisation et la compréhension de ce plugin QGIS. Un jeu de données test est aussi prévu afin d'aider à la compréhension du fonctionnement du plugin. Il serait pertinent dans ce contexte de rédiger un protocole afin d'aider les usagers à créer des jeux de données initiaux qui seront pris en charge par le plugin.

## 2.4 La réponse apportée au besoin

Oslandia nous a orienté à développer un plugin QGIS. Il était alors nécessaire de vérifier la compatibilité entre QGIS et la visualisation 3D de bâtiment modélisé. Néanmoins Oslandia n'était pas fermé sur ce choix de plugin QGIS et, en cas de besoin, aurait bien accepté un développement sur un autre logiciel sous réserve qu'il soit open source.

# 3 Analyse fonctionnelle - Solutions proposées

Les fonctionnalités attendues ont évolué entre le début et la fin du projet. Au début nous imaginions que l'utilisateur devrait choisir un niveau de détail (LoD) à reconstruire. Finalement GeoFlow crée directement un fichier résultat comprenant tous les LoD. L'utilisateur choisira simplement sur la vue 3D de QGIS quel LoD il souhaite représenter.

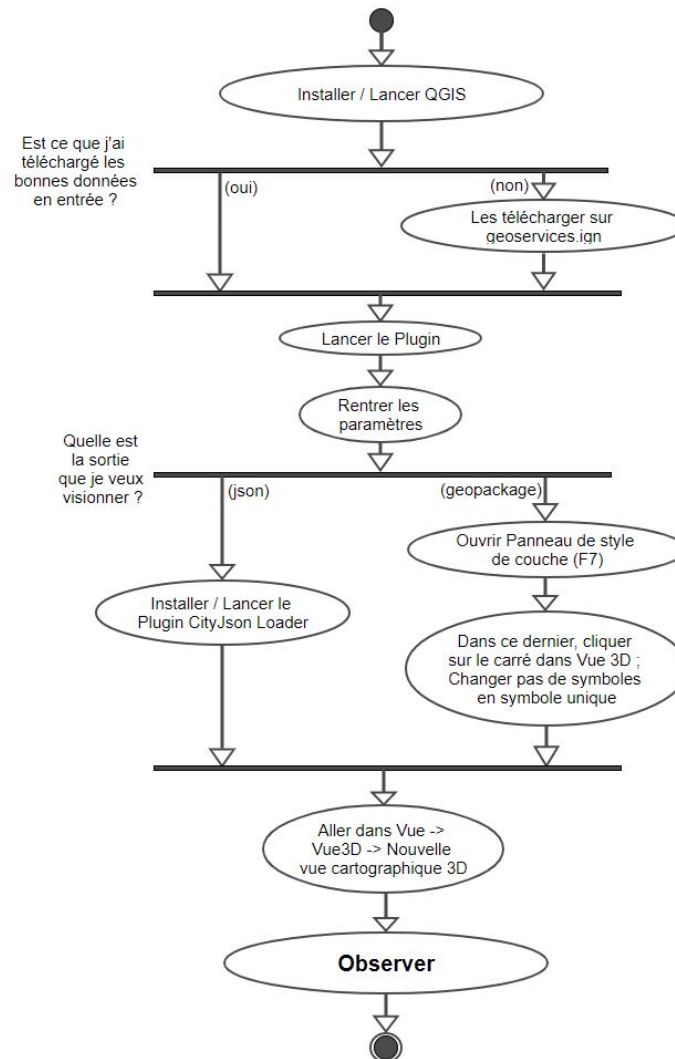


FIGURE 3 – Diagramme d'activités

Le diagramme d'activités présente la procédure qu'un utilisateur devra suivre pour faire fonctionner notre plugin. Par rapport aux attentes initiales, nous pouvons modifier ce diagramme en retirant la branche concernant le fichier Geopackage car notre plugin n'est fonctionnel qu'avec le fichier JSON produit par GeoFlow.

Ainsi, l'utilisateur devra installer puis lancer QGIS comportant notre plugin. Il devra trouver les bonnes données pour les rentrer en paramètres. Ensuite, l'utilisateur devra installer un autre plugin pour observer le fichier en sortie de GeoFlow dans une *nouvelle vue cartographique 3D* de QGIS.

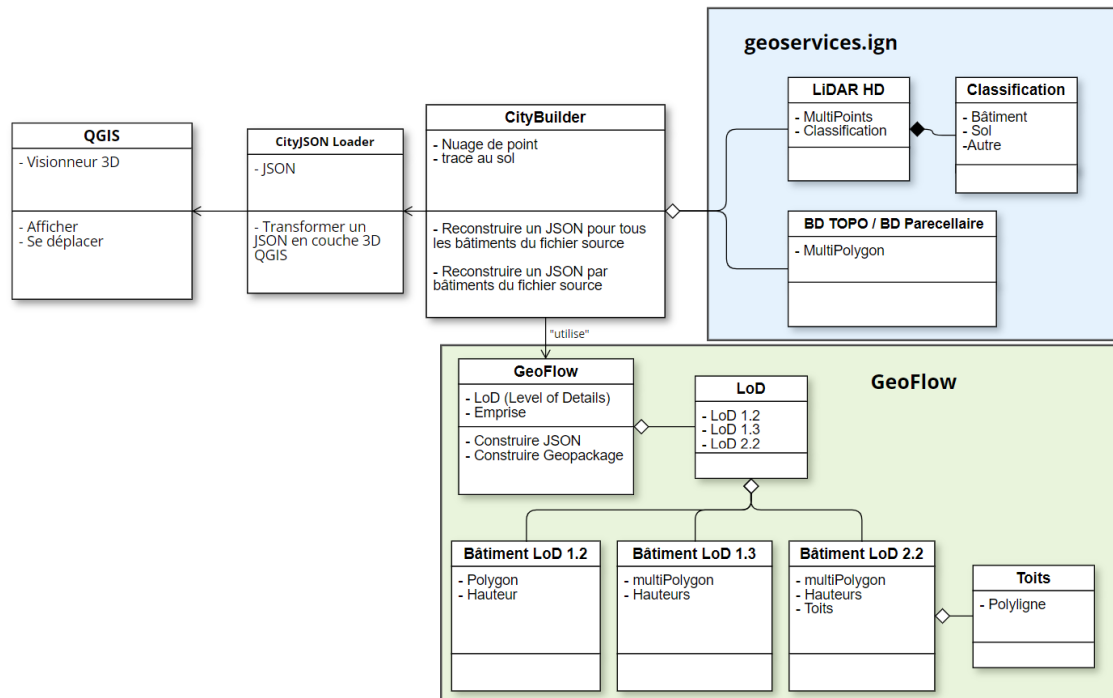


FIGURE 4 – Diagramme de classes

Notre plugin, *CityBuilder* fonctionne avec des données en entrée provenant des *geoservices.ign*. À partir de ces données téléchargées, le plugin appellera *GeoFlow* pour calculer la reconstruction des bâtiments. Enfin, cette reconstruction pourra être visualisée sur QGIS par l'utilisateur en utilisant le plugin *CityJSON Loader*.

## 4 Étude technique : Choix des logiciels et langages - Architecture

### 4.1 Les outils utilisés

Comme développé précédemment, nous utiliserons *GeoFlow* pour reconstruire les bâtiments. *GeoFlow* produit deux fichiers en sortie, un geopackage et un JSON. Pour visualiser sur QGIS le geopackage, nous utilisons l'outil *vue 3D* intégrée directement dans QGIS. Pour le fichier JSON, une étape supplémentaire est nécessaire. En effet, QGIS ne prend pas en charge les fichiers JSON. Après quelques recherches, nous avons décidé d'utiliser *CityJSON-Loader* qui est un plugin directement accessible sur QGIS : *CityJSON-loader*.

Lorsque nous avons essayé d'utiliser *GeoFlow* avec un jeu de données français, nous devions visualiser et comprendre la construction des fichiers de sortie. Alors, nous nous sommes tournés vers la visionneuse de geoJSON *CityJSON Ninja* qui est pratique pour sa simplicité et sa rapidité d'utilisation. Nous avons pu comprendre certains problèmes sur ces fichiers JSON. La lisibilité d'un fichier JSON a été très pratique pour modifier simplement le fichier afin de corriger les erreurs d'écriture en sortie de *GeoFlow*. En effet, un simple éditeur de texte suffit pour modifier le fichier au format JSON. Cela est bien différent pour le fichier geopackage. Nous ne pouvons pas modifier aussi facilement les données du fichier. Par la suite, nous avons décidé de nous concentrer sur le fichier JSON et de délaissier le fichier geopackage.

Pour développer notre plugin, nous avons utilisé deux plugins de QGIS. Le premier : *Plugin*

*Builder* pour nous donner un modèle de scripte python de plugin. Le second, *Plugin Reloader*, pour faciliter le développement du scripte python. L'interface graphique a été développée sur : *QTCreator*. Ces outils nous ont permis d'obtenir rapidement un premier prototype de plugin.

Le tableau suivant présente les différents outils et logiciels que nous utilisons pour utiliser notre plugin.

Outils principaux utilisés pour la création du plugin				
Geoflow		Jeux de données utilisées pour le plugin.		CityJSON Loader
Nuage de point (.las)	Footprint (.shp ou .gpkg)	Lidar	Cadastre	Le plugin CityJSON Loader de QGIS permet de visualiser, d'analyser et de manipuler des données CityJSON dans l'interface utilisateur de QGIS.
Fournit le nuage de point. C'est un format .las (équivalent de notre Lidar)	Fournit la trace au sol. Ces données sont utilisées en format géo package (.gpkg). C'est l'équivalent de notre cadastre	Le Lidar hd va nous fournir le nuage de point. Les données fournies sont en format .las.	Le cadastre nous donne la trace au sol de nos bâtiments. Les données utilisées ont au format shapefile (shp).	

FIGURE 5 – Présentation des outils utilisés

Sur QGIS, les deux langages les plus communément utilisés pour développer un plugin sont le *C++* et *Python*. *C++* a l'avantage d'être plus adaptée aux appels de lignes de commandes et potentiellement un peu plus rapide que *Python*. Cependant, ce langage est moins adapté pour le développement rapide d'un plugin. En effet, il est nécessaire de compiler le code en tant qu'exécutable en *C++*. La console python intégrée dans QGIS permet de charger, d'exécuter et de déboguer facilement le scripte. Enfin, nous avons été formés au langage *Python* contrairement au *C++*. Pour un projet de courte durée tel que le Projet Développement Informatique, nous préférons ne pas utiliser trop de temps à nous former sur un langage aussi complexe que le *C++*. Pour ces raisons nous avons préféré poursuivre avec *Python*.



## 4.2 Fonctionnement de la reconstruction 3D

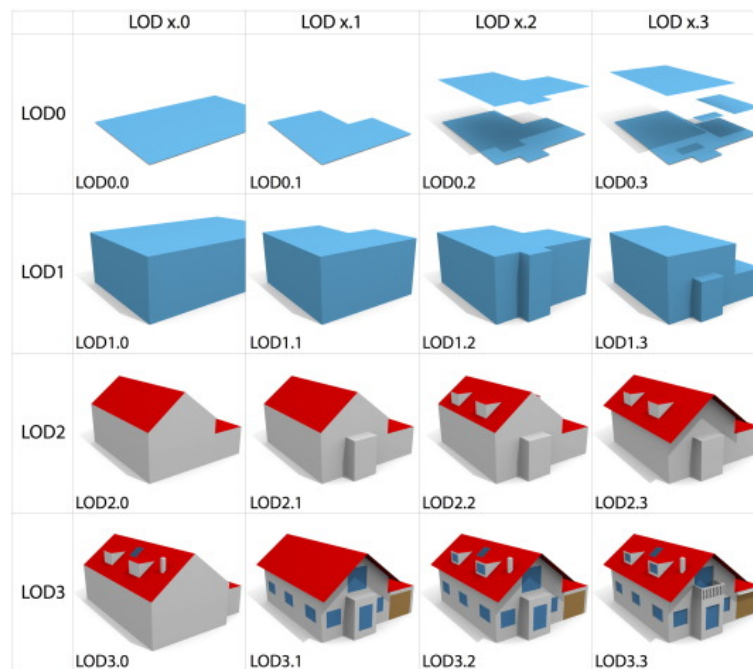


FIGURE 6 – Matrices des LoD *An improved LOD specification for 3D building models*

Pour comprendre le fonctionnement de la reconstruction 3D il est d'abord intéressant de s'intéresser aux différents LoD et se poser la question de quels LoD représenter dans notre cadre d'utilisation. L'image ci-dessus représente une grande majorité des niveaux de détails disponibles. On remarque que plus on descend dans notre tableau, plus les formes représentées sont complexes : d'un simple plan, nous passons à un pavé pour ensuite rajouter des triangles et enfin des cheminées/gouttières etc en LOD3. Aussi, plus on se déplace vers la droite, plus on ajoute des plans à notre modèle et donc on prend en compte les différences de niveaux.

Dans notre plugin nous avons choisi trois niveaux de détails : le LoD 1.2, 1.3 et 2.2. Les deux premiers sont intéressants car ils sont plus faciles à reconstruire ainsi, le côté algorithmique est maîtrisé (il en existe plusieurs et sont performants). De plus, ils suffisent à toutes les utilisations consistantes à la mesure de propagation des ondes en ville par exemple (propagation de la chaleur, du son, etc).

Le LoD 2.2 est plus utilisé pour une utilisation *ludique*, où l'objectif est d'avoir une représentation la plus réaliste possible. C'est, à l'heure actuelle, le niveau de détail maximal que l'on peut générer de manière *automatisé*. Pour aller à un niveau de détail supérieur (LoD 3 par exemple), il faut réaliser un traitement spécifique sur le bâtiment, comme un relevé photogrammétrique par exemple.

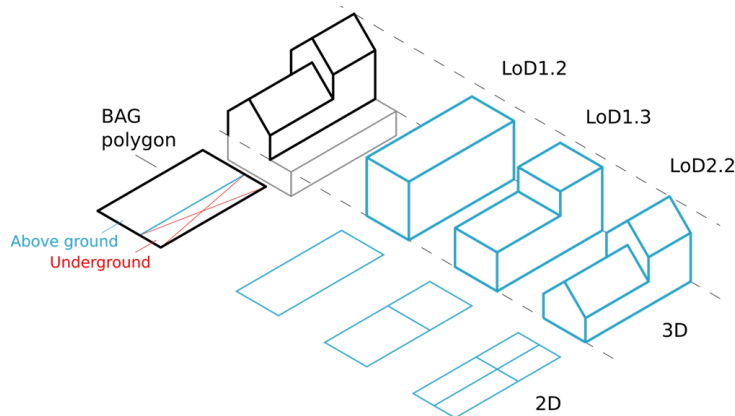


FIGURE 7 – Construction de la partition permettant la modélisation 3D *Schéma de Delft University of Technology, Automated 3D reconstruction of LoD2 and LoD1 models for all 10 million buildings of the Netherlands*

Puis, pour expliquer de manière simplifiée le principe de reconstruction 3D, dans un premier temps GeoFlow isole les points du nuage appartenant à un bâtiment à partir du polygone contenu dans le fichier shapefile, en appliquant un buffer (zone tampon) afin de prendre aussi en compte les points environnants du bâtiment. Ensuite, il va effectuer une détection des arêtes pour ensuite les projetées sur le polygone, et ainsi construire ce qu'on appelle une partition (géométrie 2D représentés en bleu ciel devant les modèles). Et c'est à partir de cette partition que le logiciel va reconstruire le modèle 3D. On note que plus le niveau de détail est élevé, plus la partition est complexe, et plus le nombre d'arêtes représenté est élevé.

### 4.3 Réalisation technique

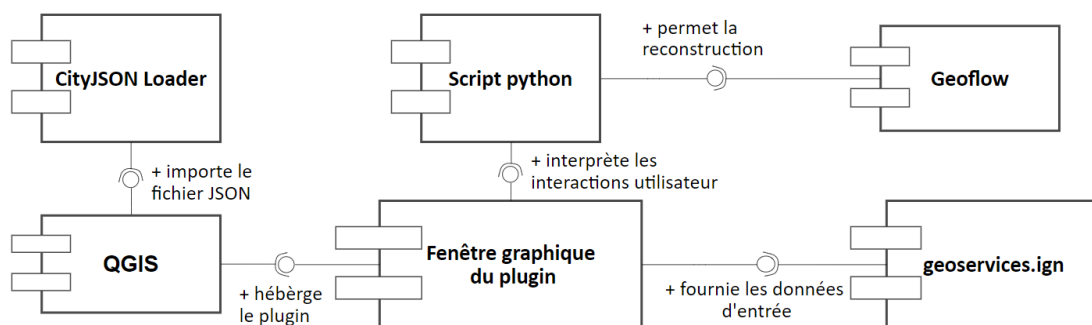


FIGURE 8 – Diagramme de composants

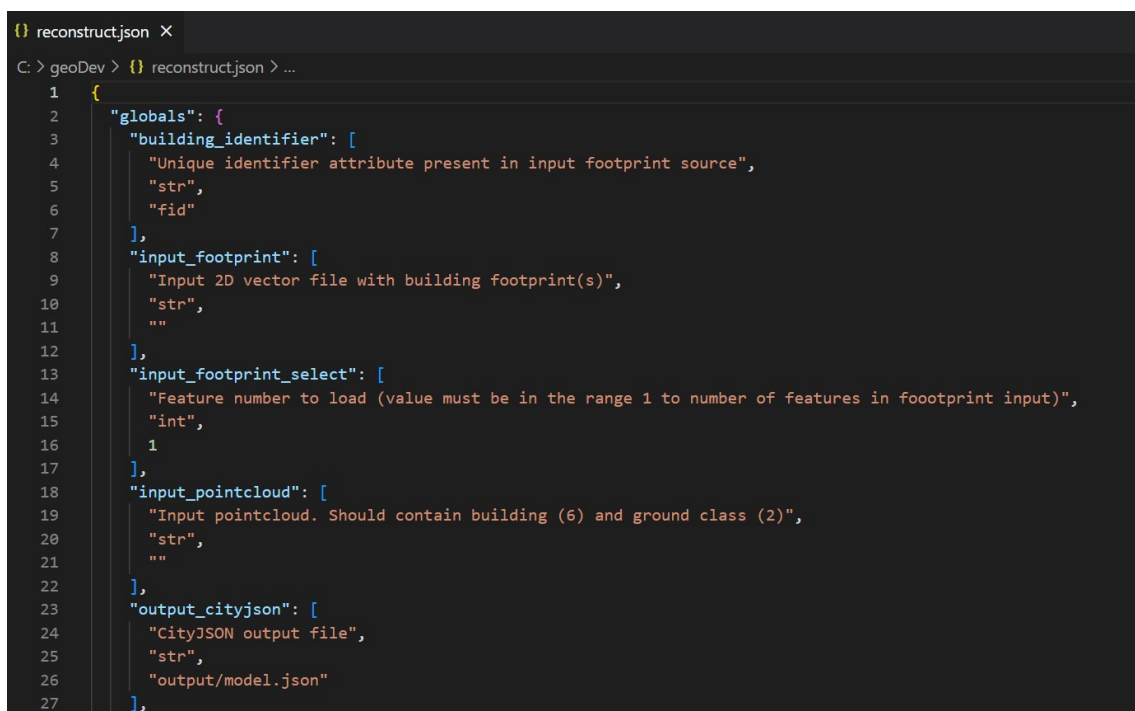
Le diagramme de composants permet de mettre en lumière les fonctionnements de notre plugin dans son environnement. Les données proviennent de *geoservices.ign*, elles sont transmises à GeoFlow grâce à notre plugin. Le résultat obtenu est un JSON que l'on doit importer sur QGIS grâce au plugin CityJSON Loader.

### 4.4 Problèmes rencontrés

Lors du développement du Plugin, nous avons rencontré plusieurs problèmes.

Le premier concernait notre compréhension du fonctionnement de GeoFlow, qui utilise des invites de commandes. Nous avons donc dû les comprendre afin de pouvoir reconstruire d'abord

un bâtiment puis les bâtiments. Maintenant, pour faire simple, l'invite de commande que l'on utilise ici se divise toujours en deux parties : l'appel à l'exécutable de Geoflow et ensuite les paramètres souhaités ; et c'est dans cette deuxième partie que l'on peut vraiment les manier pour reconstruire le bâtiment cherché. Pour cela, Geoflow a besoin de ce qu'on appelle une *flowchart*, un fichier JSON qui contient tous les paramètres fixés pour les entrées et les sorties puis les paramètres concernant directement la reconstruction. Ainsi, selon si l'on veut reconstruire un quartier ou un seul bâtiment, la *flowchart* utilisée n'est pas la même et lorsque l'on vient rajouter les différents paramètres dans l'invite de commande, c'est comme si l'on récrivait les paramètres de ce fichier. Voici à quoi ressemble la *flowchart* qui reconstruit un seul bâtiment mais en LoD 2.2.



```

1 {
2   "globals": {
3     "building_identfier": [
4       "Unique identifier attribute present in input footprint source",
5       "str",
6       "fid"
7     ],
8     "input_footprint": [
9       "Input 2D vector file with building footprint(s)",
10      "str",
11      ""
12    ],
13    "input_footprint_select": [
14      "Feature number to load (value must be in the range 1 to number of features in footprint input)",
15      "int",
16      1
17    ],
18    "input_pointcloud": [
19      "Input pointcloud. Should contain building (6) and ground class (2)",
20      "str",
21      ""
22    ],
23    "output_cityjson": [
24      "CityJSON output file",
25      "str",
26      "output/model.json"
27    ]
28  }
29 }

```

FIGURE 9 – Capture d'écran d'une *flowchart* pour mieux comprendre

Et voici comment s'organise alors l'invite de commande basique avec cette *flowchart* :

#### Invite de commandes

```
geof reconstruct.json -input_footprint=... -input_pointcloud=... -output_cityjson=...
```

Le deuxième problème majeur était lié à la création de notre jeu de données. Nous devons choisir entre deux sources de données pour notre empreinte au sol (footprint) : les données de la BD Topo de l'IGN ou les données du cadastre français (BD parcellaire). Après avoir comparé ces deux sources, nous avons constaté qu'il pouvait y avoir un décalage d'environ deux mètres entre la trace au sol et le nuage de points LiDAR dans le cadastre. Cependant, cela n'affecte pas le fonctionnement de GeoFlow qui reconstruit les bâtiments en utilisant une zone tampon de quatre mètres autour de la trace au sol. La BD Topo est plus précise, mais elle utilise un système de coordonnées EPSG différent de notre nuage de points LiDAR, contrairement aux données du cadastre. Par souci de simplicité, nous avons donc décidé d'utiliser les données du cadastre. Comme le cadastre de notre jeu de données pour l'exemple était encore d'actualité, cela ne posait pas véritablement de problème. Par contre, de manière générale, il est préférable d'utiliser la librairie *proj* de python pour projeter la couche vecteur dans le bon EPSG.

Le troisième problème concernait la visualisation en 3D des bâtiments reconstruits dans QGIS. Pour chaque invite de commande lancée, GeoFlow prend en entrée la trace au sol (.shp) et le nuage de points LiDAR (.las) pour générer un fichier CityJSON correspondant au bâtiment reconstruit. Cependant, lorsque nous essayions de visualiser la vue 3D sur QGIS, nous observions, à première vue, une fenêtre blanche sans donnée. Nous avons alors utilisé la visionneuse *CityJSON Ninja* et avons constaté que le bâtiment était bien reconstruit en 3D, mais à une hauteur bien plus élevée que la trace au sol. La caméra de QGIS pour la visualisation 3D était fixée sur la trace au sol, ce qui nécessitait une importante prise de recul sur la vue ("dézoomer") pour apercevoir le bâtiment. Pour résoudre ce problème d'altitude, nous avons créé un script Python qui modifiait automatiquement le fichier JSON pour recentrer le bâtiment à la bonne hauteur.

Enfin, nous avons aussi rencontré une complication pour faire fonctionner notre script depuis QGIS mais en utilisant la console Python de QGIS, on a pu globalement voir ce qu'il se déroulait. Le problème concernait surtout le chemin de la *flowchart* pour reconstruire tout un quartier. Un problème que l'on a surpassé en comprenant que l'invite de commande pour cela doit être faite depuis un certain endroit pour qu'il fonctionne. Un endroit où il y a aussi le fichier *reconstruct\_one.json*.

Problèmes techniques		Solutions trouvées
Compréhension de Geoflow	<ul style="list-style-type: none"> <li>- Comprendre comment fonctionnent les invites de commandes de Geoflow</li> </ul>	<ul style="list-style-type: none"> <li>- Lecture et compréhension du Git de Geoflow.</li> <li>- Création de plusieurs jeux de données test.</li> </ul>
Jeux de données	<ul style="list-style-type: none"> <li>- EPSG qui ne coïncidait pas</li> <li>- Problèmes de formats de données liés au cadastre (difficultés à trouver les données directement en format shapefile).</li> </ul>	<ul style="list-style-type: none"> <li>- Nous avons choisi le cadastre pour régler les problèmes d'EPSG qui ne coïncidait pas.</li> <li>- Au début conversion des données du cadastre qui étaient au format JSON au format shapefile.</li> <li>- Nous avons ensuite trouvé ces données directement au format Shapefile.</li> </ul>
Problème de visualisation des données	<ul style="list-style-type: none"> <li>- Problème de positionnement dans l'espace des bâtiments avec la génération du JSON brut.</li> <li>- Problème d'agrégation des bâtiments en Lod 2.2</li> </ul>	<ul style="list-style-type: none"> <li>- Script Python pour modifier directement le JSON généré et régler les problèmes.</li> <li>- Création d'un script python différent pour régler les problèmes d'agrégation de bâtiments en Lod 2.2.</li> <li>- Visualisation du JSON sur Ninja.</li> </ul>
Développement du plugin dans Qgis	<ul style="list-style-type: none"> <li>- Difficultés à faire fonctionner le script python depuis Qgis.</li> </ul>	<ul style="list-style-type: none"> <li>- Utiliser la console Python Qgis.</li> </ul>

FIGURE 10 – Tableau récapitulatif des problèmes rencontrés

## 4.5 Résultats obtenus

Les onglets *Footprint* et *Point cloud* servent d'entrées pour charger respectivement la trace au sol (footprint) et le nuage de points nécessaires à GeoFlow pour exécuter la ligne de commande. La case à sélectionner *All in one File* permet de ne produire qu'un seul fichier en sortie. Alors tous les bâtiments de la zone en entrée seront modélisés et enregistrés dans un JSON en LoD 0. 1.2 et 1.3. Si la case n'est pas sélectionnée, un fichier par bâtiment sera produit. Cette reconstruction permettra d'intégrer en plus le LoD 2.2. En effet, GeoFlow ne supporte pas la reconstruction de plusieurs bâtiments en un seul fichier. Ensuite *Output* permet de choisir le dossier où le fichier JSON produit par GeoFlow sera enregistré. Lorsque l'utilisateur clique sur *Run*, la commande est exécutée. (*voir fig. 11*)

Nous avons construit nos jeux de données de la manière suivante : pour la trace au sol, nous avons téléchargé les données du cadastre au format *Shapefile*, ainsi que les données du LiDAR HD (déjà classifiées) disponibles sur le site de l'IGN. Ensuite, nous avons découpé la tuile LiDAR en fonction de la ville, du quartier ou du bâtiment qui nous intéressait. Nous avons également sélectionné les bâtiments du cadastre que nous souhaitions modéliser. Les résultats que vous pouvez voir ci-dessous ont été obtenus à partir d'un jeu de données de la ville de Manosque.

Enfin, la documentation utilisateur, les jeux de données test ainsi que le code du plugin commenté sont disponibles en open source sur ce lien GitLab : [GitLab.com/CityBuilder](https://gitlab.com/CityBuilder)

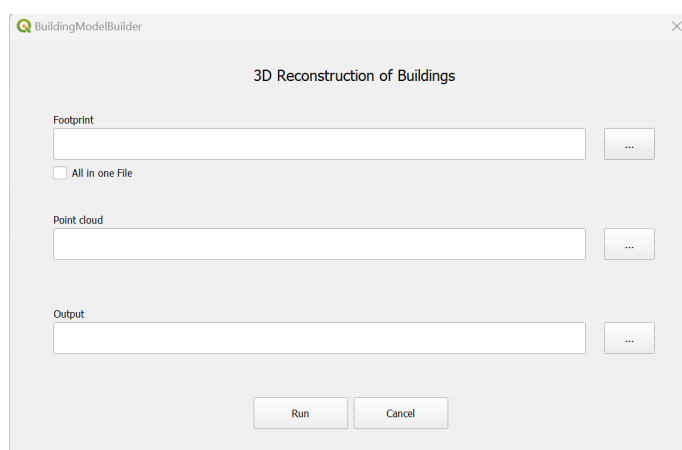


FIGURE 11 – Interface graphique du plugin

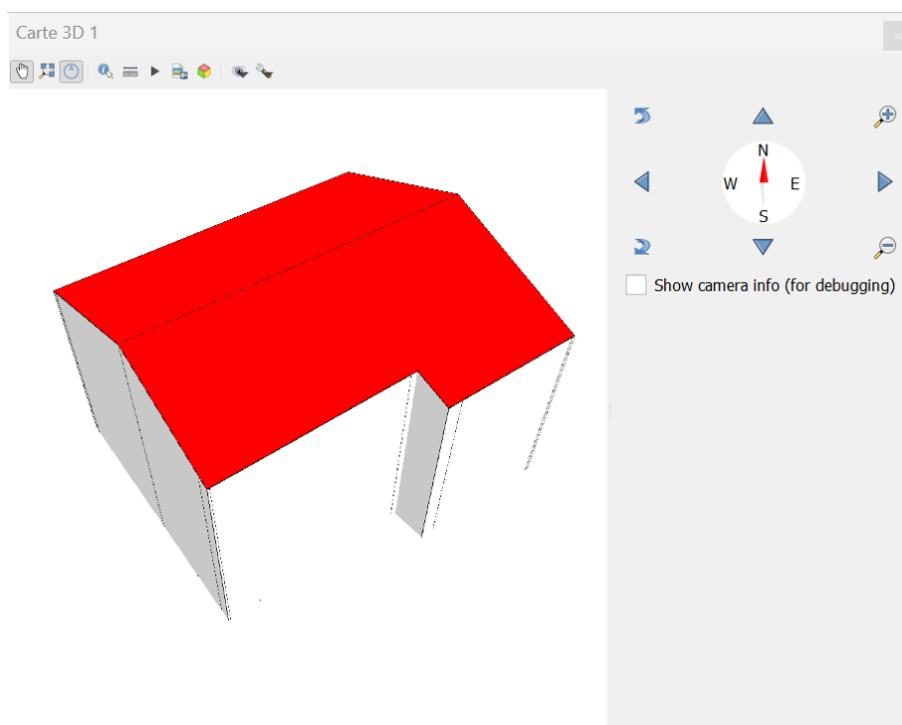


FIGURE 12 – Visualisation sous QGIS de la reconstruction 3D d'un bâtiment simple de Manosque à partir de notre jeu de données

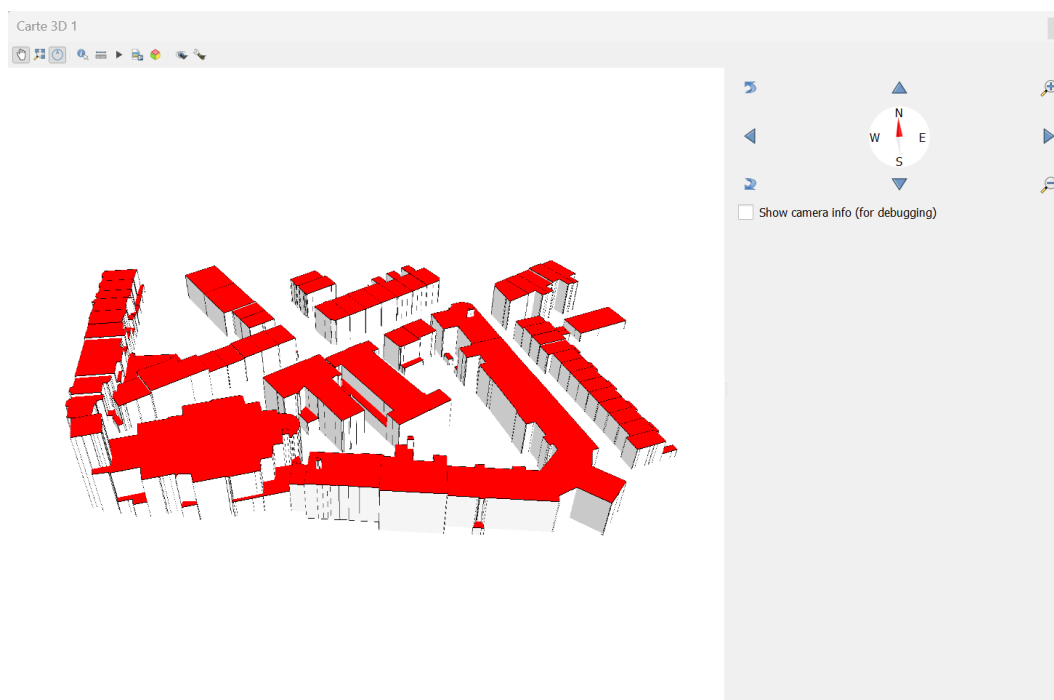


FIGURE 13 – Visualisation sous QGIS de la reconstruction 3D d'un quartier

#### 4.6 Choix du logo et du nom du plugin QGIS

Avec l'avis de notre commanditaire Thomas Muguet, nous avons décidé d'appeler notre plugin *CityBuilder*. Nous avons ensuite dessiné un logo aux couleurs d'Oslandia (bleu et blanc) et de l'ENSG (vert et blanc). Pour plus de contraste, nous avons utilisé un fond noir. Les bâtiments dessinés sont construits à partir de formes géométriques "grossières" pour conserver le style graphique du plugin CityJSON Loader que nous devons utiliser pour importer le résultat produit par GeoFlow.



FIGURE 14 – Logo de notre plugin CityBuilder

#### 4.7 Perspectives d'évolutions

Durant le PDI, nous avons développé principalement une première version du plugin CityBuilder permettant de valider l'étape "preuve du concept" attendu par notre commanditaire. Nous pourrions apporter dans le futur quelques évolutions pour compléter ce premier prototype.

Tout d'abord, nous pourrions retravailler sur l'esthétique de la fenêtre graphique du plugin,

mais aussi sur son ergonomie en ajoutant des fonctionnalités comme le choix du niveau de détail souhaité (LoD) ou le choix du système de référence (EPSG).

Ensuite, nous avons abandonné le développement d'une fonctionnalité au cours du projet par manque de temps. Il aurait été intéressant d'intégrer notre plugin dans la boîte à outil de QGIS et de donner la possibilité d'appliquer un traitement QGIS qui permet de reconstruire un bâtiment d'une couche cadastrale en le sélectionnant directement par son polygone sous QGIS.

Avec plus de temps, nous aurions aimé intégrer le plugin City JSON Loader directement dans notre scripte python. Cela aurait permis d'obtenir directement le fichier JSON en sortie de GeoFlow dans QGIS sans manipulation de l'utilisateur et sans le téléchargement d'un plugin annexe. Aussi, pour la reconstruction d'un quartier entier en LoD 2.2, notre scripte ouvre un terminal par bâtiment à reconstruire. Une évolution possible serait de reconstruire chaque bâtiment dans le même terminal pour éviter l'ouverture de nombreuses fenêtres en simultanée.

## 5 Réalisation et suivi de projet







### 5.1 Les risques

Ci-dessous notre matrice des risques mise à jour tout au long du projet. La version incluse dans le document correspond à la dernière matrice des risques disponible à ce jour.

Risques	Ordre de priorité				Remédiation
	Probabilité	Impact	Score	Evolution	
Mauvaise communication entre les membres de l'équipe	1	4	Moyen : 4	=	<ul style="list-style-type: none"> <li>Planifier des réunions régulières pour discuter de l'avancement du projet et des problèmes éventuels.</li> <li>Utiliser des outils de collaboration en ligne pour partager les informations et les mises à jour.</li> <li>Définir des canaux de communication clairs et des responsabilités pour les membres de l'équipe.</li> </ul>
Désaccords au sein de l'équipe concernant la stratégie à adopter	1	3	Faible : 3	↘	<ul style="list-style-type: none"> <li>Impliquer tous les membres de l'équipe dans la planification du projet dès le début.</li> <li>Établir des objectifs clairs et convenir des critères d'évaluation pour chaque étape du projet.</li> </ul>
Incompatibilité des emplois du temps des membres de l'équipe	1	5	Moyen : 5	↗	<ul style="list-style-type: none"> <li>Planifier un calendrier de travail détaillé qui prend en compte les disponibilités de chacun.</li> <li>Anticiper les absences éventuelles et prévoir des mesures pour faire face à ces situations.</li> </ul>
Dépendance à l'égard d'un membre de l'équipe	3	2	Modéré : 6	↘	<ul style="list-style-type: none"> <li>Assurer la formation de tous les membres de l'équipe pour qu'ils soient polyvalents.</li> <li>Prévoir des mesures de secours si un membre de l'équipe est absent ou indisponible.</li> </ul>
Retard dans la livraison des livrables	2	5	Elevé : 10	=	<ul style="list-style-type: none"> <li>Établir des délais réalistes pour chaque livrable.</li> <li>Mettre en place des mécanismes de suivi et de contrôle pour surveiller l'avancement du projet.</li> <li>Planifier des marges de temps supplémentaires pour faire face aux imprévus.</li> </ul>
Mauvaise qualité des livrables	2	4	Modéré : 6	=	<ul style="list-style-type: none"> <li>Établir des critères de qualité clairs pour chaque livrable.</li> <li>Mettre en place des processus de contrôle qualité pour vérifier la qualité des livrables avant leur livraison.</li> </ul>



Risques	Ordre de priorité				Remédiation
	Probabilité	Impact	Score	Evolution	
Délai trop court pour réaliser les tâches	3	4	Elevé : 8		<ul style="list-style-type: none"> <li>Établir des priorités claires pour chaque tâche.</li> <li>Répartir les tâches en fonction des compétences de chacun.</li> <li>Mettre en place un calendrier de travail réaliste qui prend en compte les délais nécessaires pour chaque tâche.</li> </ul>
Insuffisance des compétences techniques des membres de l'équipe	2	3	Elevé : 9		<ul style="list-style-type: none"> <li>Établir une liste des compétences requises pour le projet et s'assurer que chaque membre de l'équipe possède ces compétences.</li> <li>Prévoir des formations ou des tutoriels pour les membres de l'équipe qui ont besoin de renforcer leurs compétences.</li> </ul>
Manque de ressources pour réaliser les tâches	1	1	Faible : 3		<ul style="list-style-type: none"> <li>Planifier les ressources nécessaires pour chaque tâche.</li> <li>Identifier les ressources disponibles et les ressources supplémentaires qui peuvent être nécessaires.</li> <li>Établir un budget réaliste pour le projet.</li> </ul>
Changement de la portée du projet	1	1	Très faible: 4		<ul style="list-style-type: none"> <li>Établir une portée claire pour le projet dès le début.</li> <li>Planifier des mécanismes de contrôle et de suivi pour détecter tout changement dans la portée du projet.</li> <li>Évaluer l'impact de tout changement de portée sur les délais, les ressources et les coûts.</li> </ul>
Difficulté à obtenir le feedback du commanditaire	3	3	Elevé : 9		<ul style="list-style-type: none"> <li>Établir une communication régulière avec le commanditaire, en lui envoyant des mises à jour fréquentes sur l'avancement du projet et en sollicitant son feedback.</li> <li>S'assurer que le commanditaire ait toutes les informations nécessaires pour donner son feedback, en lui fournissant une documentation claire et précise.</li> </ul>
Conflits d'intérêts entre les membres de l'équipe	1	2	Très faible : 2		<ul style="list-style-type: none"> <li>Identifier les domaines de conflit potentiel dès le début du projet et travailler à les résoudre rapidement.</li> <li>Établir des rôles clairs pour chaque membre de l'équipe et s'assurer que chacun sait ce qu'on attend de lui.</li> <li>Encourager une communication ouverte et honnête au sein de l'équipe pour éviter les malentendus.</li> </ul>

Risques	Ordre de priorité				Remédiation
	Probabilité	Impact	Score	Evolution	
Absence ou départ d'un membre de l'équipe en cours de projet	1	2	Très Faible : 2		<ul style="list-style-type: none"> <li>Identifier les domaines de risque où la contribution de ce membre de l'équipe est essentielle et planifier des actions pour pallier son absence.</li> <li>Établir un plan de communication clair pour informer le commanditaire de toute absence ou départ de membre de l'équipe.</li> </ul>
Perte de motivation ou d'engagement des membres de l'équipe	2	3	Moyen : 6		<ul style="list-style-type: none"> <li>Établir des objectifs clairs pour chaque membre de l'équipe et s'assurer qu'ils soient pertinents et atteignables.</li> <li>Organiser des réunions régulières pour suivre l'avancement du projet et pour encourager l'échange d'idées et le feedback constructif.</li> </ul>
Manque de transparence dans la gestion du projet	1	3	Faible : 3		<ul style="list-style-type: none"> <li>Établir une communication claire et ouverte avec tous les membres de l'équipe en fournissant des mises à jour régulières sur l'avancement du projet, les changements de portée, etc.</li> <li>Encourager la communication ouverte en permettant à tous les membres de l'équipe de poser des questions et d'exprimer leurs préoccupations.</li> </ul>
Ne pas réussir à faire fonctionner Geoflow avec nos jeux de données français	1	4	Faible : 4		<ul style="list-style-type: none"> <li>Plusieurs solutions</li> </ul>
Le bâtiment reconstruit sur Qgis se trouve bien plus haut que sa trace au sol.	1	3	Faible : 3		<ul style="list-style-type: none"> <li>Modifier directement dans le JSON créé avec les données française la hauteur.</li> <li>Création d'un script python qui modifie automatiquement la hauteur du bâtiment dans le JSON.</li> </ul>
Réussir à intégrer le bon EPSG au JSON du bâtiment français.	1	4	Faible : 4		<ul style="list-style-type: none"> <li>Modifier directement dans le JSON créé avec le bon EPSG.</li> <li>Création d'un script python qui modifie automatiquement l'EPSG du bâtiment dans le JSON.</li> </ul>

Risques	Ordre de priorité				Remédiation
	Probabilité	Impact	Score	Evolution	
Ne pas réussir à faire fonctionner le script python en intégrant le plugin cityjson loader	3	4	élevé : 12		<ul style="list-style-type: none"> <li>Essayer de trouver un code Open source de City Json loader et l'intégrer à notre script python</li> </ul>
Ne pas réussir à faire fonctionner le Lod 2.2 pour plusieurs bâtiments avec notre script python.	2	2	Faible : 4		<ul style="list-style-type: none"> <li>Abandon du LOD 2.2</li> <li>Création d'un script python intégré au script final spécialement conçu pour faire fonctionner le cas particulier du lod 2.2</li> </ul>
Ne pas réussir à intégrer les différents script python ensemble au plugin Qgis.	2	5	Elevé : 10		<ul style="list-style-type: none"> <li>Trouver un moyen de déboguer le script pour l'intégrer au plugin.</li> <li>Demander de l'aide aux personnes compétentes.</li> </ul>
Rester bloquer sur l'orienté objet (pour intégrer le script python au plugin).	1	5	Moyen : 5		<ul style="list-style-type: none"> <li>Demander aux personnes compétentes</li> <li>Montée en compétence de l'équipe par auto-instruction.</li> </ul>
Risque qu'à la fin du projet, le plugin ne soit compatible qu'avec le type d'ordinateur avec lequel il a été conçu.	5	3	Très élevé : 15		<ul style="list-style-type: none"> <li>Informier l'utilisateur dans le README de GitLab et la documentation utilisateur</li> </ul>
Risque que ce plugin ne soit compatible qu'avec des versions de QGIS supérieure à la 3.20 et inférieure à la 3.30	5	2	Elevé : 10		<ul style="list-style-type: none"> <li>Informier l'utilisateur dans le README de GitLab et la documentation utilisateur</li> </ul>



## 5.2 Planning prévisionnel

Ce planning prévisionnel détaillé résume les grandes étapes de la gestion de notre projet ainsi que notre planification du temps nécessaire pour chaque étape. Il nous permet de visualiser de manière claire et organisée les différentes phases du projet, depuis la phase d'analyse et de conception jusqu'à la phase de développement, de test et de déploiement. Ce planning nous aide à coordonner efficacement les activités, à définir les délais et les échéances, et à prendre les mesures nécessaires pour respecter les objectifs fixés. Il constitue un outil essentiel pour assurer une gestion efficace du projet et optimiser notre gestion du temps.

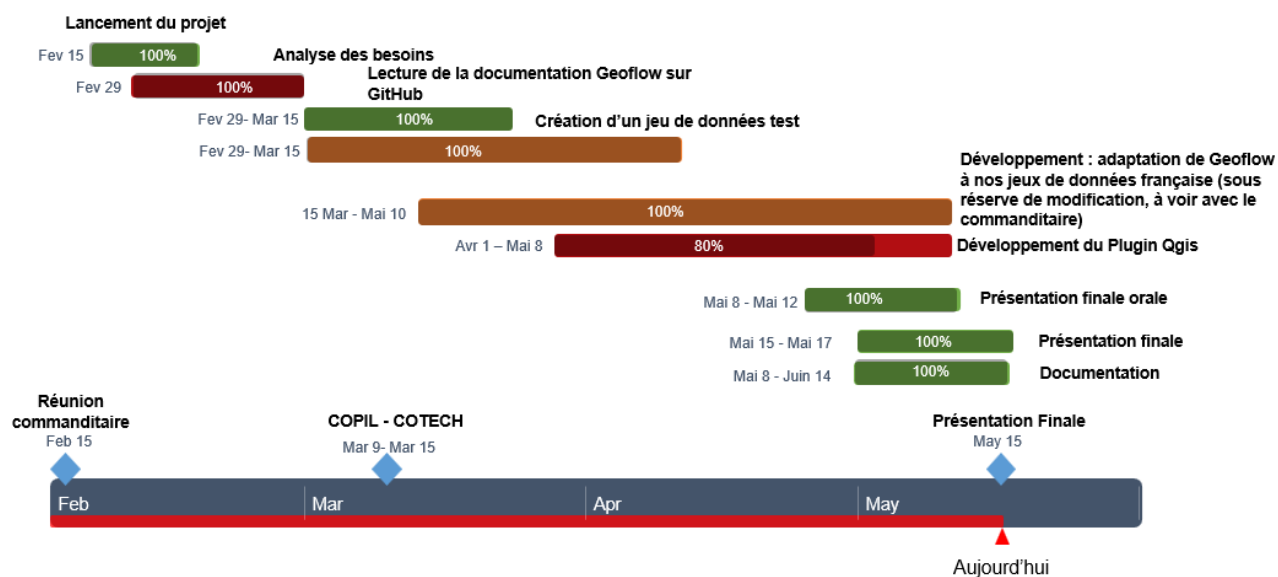


FIGURE 15 – Planning prévisionnel

Pour mener à bien ce projet, nous avons dans un premier temps mis en place les outils de gestion de projet tel que le GitHub ou encore le tchat Matrix. Ensuite nous avons vu avec notre commanditaire les objectifs à atteindre en en faisant des tickets sur le Git. Puis, nous avons réalisé une première phase de documentation sur les travaux des chercheurs hollandais ainsi que sur GeoFlow. Ce n'est donc qu'à partir de mi-mars que nous avons commencé la phase de développement par la construction de plusieurs jeux de données test simulant différents cas d'utilisation : d'abord un bâtiment simple, puis un quartier entier, et ceux dans deux endroits différents. Alors nous avons pu commencer à adapter GeoFlow avec nos données françaises pour enfin s'atteler au développement du plugin QGIS.

### 5.2.1 Diagramme de Pert

Le diagramme de Pert ci-dessous nous présente les grandes tâches du projet et définit les dépendances entre les tâches. Les flèches en rouge montrent le chemin critique.

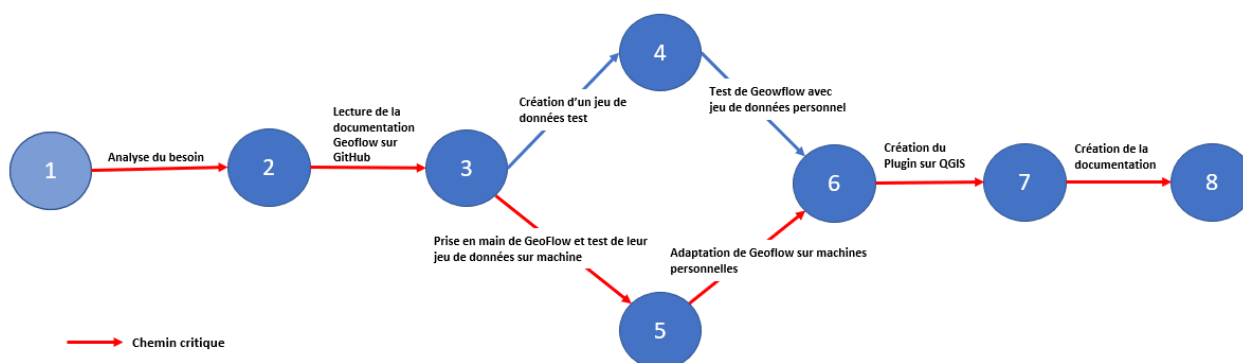


FIGURE 16 – Diagramme de Pert

### 5.2.2 Diagramme de Gantt

Le diagramme de Gant (cf. fig.17), a été réalisé au début du projet.

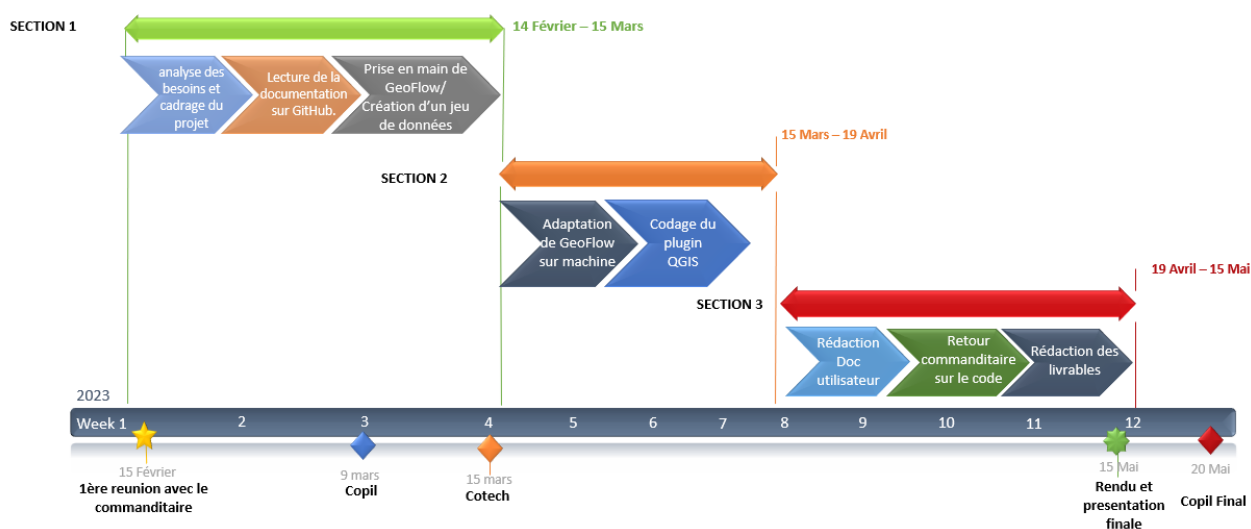


FIGURE 17 – Diagramme de Gantt

### 5.3 Organisation

Nous avons décidé de fonctionner avec un système de tickets sur *GitHub* afin de se séparer les tâches. Chaque ticket ayant un état d'avancement (en cours de traitement, fini, à faire, en attente, etc), cela nous a permis d'avoir tout au long du projet un certain recul sur notre avancée, et ainsi nous a permis de mieux gérer notre travail. Aussi ces derniers nous ont permis de discuter de manière formelle avec notre commanditaire, et c'est ainsi que nous avons pris toutes les informations importantes. Tous les travaux, de recherche, de synthèse ou de développement de code ont été déposés sur ce Git. Par ailleurs, une conversation a été créée sur *Matrix* via *Element* afin de pouvoir joindre rapidement notre commanditaire. Des visioconférences ont été organisées en fonction de nos besoins. En effet, Oslandia est une entreprise orienté télétravail ce qui permet de mettre facilement en place tous ces outils de communication à distance.

Nous avons choisi de développer en méthode agile pour travailler de manière itérative pour livrer rapidement une première version à notre commanditaire et l'adapter selon ses retours.

Cependant, la première version fonctionnelle de notre plugin est arrivée sur la fin du projet. Notre commanditaire a ainsi testé de son côté le plugin pour le valider en guise de recette. Même si nous n'avons pas développé de manière itérative plusieurs versions du plugin, la méthode agile a permis de mettre en place une communication régulière avec notre commanditaire, ce qui était parfaitement adapté pour le PDI.

## 5.4 Retour sur les attentes du commanditaire

En revenant sur nos objectifs du début, nous pouvons constater que les grands objectifs de base ont été atteints :

- Permettre de reconstruire tous les bâtiments du jeu de données en entrée.
- Possibilité de reconstruire uniquement les bâtiments sélectionnés.
- Possibilité de choisir le niveau de détail souhaité.

Après une discussion avec notre commanditaire, ces divers objectifs ont été classés et chacun s'est vu attribuer une note reflétant son niveau d'importance.

Cependant, certains objectifs étaient considérés comme moins prioritaires, n'ont pas été atteints en raison des contraintes de temps.

	Objectif	Détail	Priorité (/10)	Etat
Preuve du concept	Preuve de concept de reconstruction 3d <b>LoD 1.3 via Geoflow</b>	<ul style="list-style-type: none"> <li>• Création d'un jeu de donnée test</li> <li>• Créer un fichier GeoJSON/Geopackage à partir de notre jeu de données grâce à Geoflow</li> <li>• Visualiser le nuage de points test</li> </ul>	10	Atteint
	Preuve de concept de reconstruction 3d <b>LoD 2.2 via Geoflow</b>		8	Atteint
	Preuve de concept <b>de visualisation de bâtiments dans QGIS 3D</b>	Investiguer l'affichage des bâtiments dans QGIS	8	Atteint
	<b>Reconstruction d'un bâtiment à partir de script python dans QGIS</b> avec des données de test	Reconstruction d'un bâtiment depuis la console Python QGIS avec des fichiers source déjà existants	8	Atteint

	Objectif	Détail	Priorité (/10)	Etat
Développement du Plugin	Plugin de reconstruction <b>d'un</b> bâtiment depuis un plugin QGIS via une boîte de dialogue pour <b>sélectionner les fichiers</b> d'entrées et de sortie	Reconstruction d'un bâtiment depuis la console Python QGIS avec des fichiers source déjà existants	6	Atteint
	Plugin de reconstruction <b>d'un</b> bâtiment depuis un plugin QGIS via une boîte de dialogue pour <b>sélectionner les couches QGIS</b> et <b>afficher le résultat</b> comme nouvelle couche	Affichage de bâtiment reconstruit depuis la console Python QGIS	5	Abandonné
	Plugin de reconstruction d'un bâtiment depuis un plugin QGIS via un traitement QGIS	Création d'un traitement QGIS (accessible depuis la toolbox) pour reconstruire un bâtiment depuis la console Python QGIS avec des fichiers source déjà existants	1	Abandonné
	Plugin de reconstruction <b>de plusieurs</b> bâtiments depuis un plugin QGIS via une boîte de dialogue pour sélectionner les fichiers d'entrées et de sortie	Reconstruction de plusieurs bâtiments depuis la console Python QGIS avec des fichiers source déjà existants	6	Atteint

	Objectif	Détail	Priorité (/10)	Etat
Livrables	Produire la documentation utilisateur (et technique si besoin)	Production d'un documentation pour permettre aux utilisateurs d'être autonomes sur tout le processus	8	En cours
	Rendre le code disponible en open-source sur gitlab.com		10	En cours
	Article sur le blog Oslandia		3	Abandonné