

RAPPORT DE STAGE PFA

2025

Mise en place d'une chaîne de production open source pour la modélisation 3D cityGML avec intégration d'une Base de données 3DcityDB sous DOCKER.



RÉALISÉ PAR:

- SAMNE PENGDWENDE BORIS

ENCADRÉ PAR:

- Mme Fatine Choujae
- M. Jalil BEN ADDOU IDRISI
- Mme Fatima-Zahra BAMESSAOUD
- Pr. YAZIDI ALAOUI Otmane

DATE

Du 01 juil.
au 09 sept.
2025

Remerciements

Je souhaite exprimer ma profonde gratitude à **M. Jalil BEN ADDOU IDRISI**, Directeur de la BU SIG – DSi, pour m'avoir offert cette précieuse opportunité de stage au sein d'ETAFAT et pour la confiance accordée tout au long de cette expérience. Mes remerciements les plus sincères vont à **Mme Fatine Choujae**, Cheffe de département 3DRC-BIM – BU 3D MAPPING (mon encadrante pendant le stage), ainsi qu'à **Mme Fatima-Zahra BAMESSAOUD**, pour leur encadrement attentif, leurs conseils avisés et leur disponibilité constante. Leur soutien et leur expertise ont été déterminants dans la réalisation de ce projet et ont permis un apprentissage riche sur le plan technique et professionnel. Je tiens également à remercier chaleureusement **l'ensemble de mes professeurs** de la filière Géoinformation à la **Faculté des Sciences et Techniques de Tanger**. Leur enseignement rigoureux, leur accompagnement patient et leur passion pour la discipline ont constitué un socle solide, indispensable à la réussite de ce stage. Enfin, je remercie **ma famille**, pour son soutien inconditionnel et son accompagnement constant tout au long de mes études. Leur encouragement, leur patience et leurs conseils ont été une source précieuse de motivation et de confiance. Ces remerciements s'adressent aussi à tous ceux qui, de près ou de loin, ont contribué à cette expérience enrichissante et formatrice.

Résumé

Ce stage s'inscrit dans le domaine de la modélisation urbaine 3D et de l'utilisation de standards ouverts tel que CityGML. Il a porté sur la mise en place d'une chaîne de production open source permettant de transformer des données géospatiales (shapefiles, nuages de points, géopackages) en maquettes numériques 3D géoréférencées, stockées dans une base de données 3D (3DCityDB) et exploitables pour visualisation et analyse. La méthodologie adoptée a combiné préparation et nettoyage des données, conception UML, édition manuelle de fichiers CityGML, importation dans PostgreSQL/PostGIS/3DCityDB via Docker, puis exploitation et visualisation à l'aide d'outils spécialisés comme KIT Model Viewer. Le travail a permis de valider la faisabilité de cette approche, d'illustrer les différents niveaux de détail (LOD) des bâtiments et de démontrer le potentiel des solutions open source pour la gestion urbaine et l'aménagement du territoire. Le stage met également en évidence les limites de l'édition manuelle pour des projets à grande échelle et ouvre des perspectives d'automatisation, de pipelines hybrides et d'intégration dans des workflows urbains plus vastes. Il constitue une expérience enrichissante en termes de compétences techniques et professionnelles, tout en apportant un intérêt concret pour l'entreprise dans la valorisation des données 3D.

Table des matières

Remerciements	2
Résumé	3
Table des matières	4
Liste des sigles et acronymes	6
Liste des figures	8
1. Introduction	10
2. Présentation de l'entreprise d'accueil	11
2.1 Identification et situation	11
2.2 Domaines d'activité et expertise	11
2.3 Positionnement sur le marché et projets récents	11
2.4 Organisation interne et rôle du département SIG	12
3. Contexte et état de l'art	13
3.1. La modélisation urbaine 3D et ses enjeux	13
3.2. Le standard CityGML	13
3.3. Les données sources : shapefiles des bâtiments, geopackage, nuage de points	14
3.4. Outils et logiciels utilisés pour la chaîne de production	14
3.5. État de l'art des approches de production CityGML	17
4. Méthodologie suivie	18
4.1 Analyse et préparation des données sources	18
4.2 Conception du schéma UML	23
4.3 Édition manuelle du fichier CityGML	24
4.4 Stockage et exploitation de la maquette 3D	25
4.5 Limites de l'approche	28
4.6 Résumé du workflow	29
5. Résultats obtenus	31
5.1 Bâtiment de référence étudié	31
5.2 Illustration des niveaux de détails (LOD-Level Of Details)	34

5.3 Exemple d'une ville entière.....	35
6. Discussion et limites	36
 6.1 Comparaison avec d'autres méthodes	37
 6.2 Limites techniques	37
 6.3 Perspectives d'amélioration.....	37
7. Conclusion	39
8. Bibliographie - webographie	40
9. Annexes.....	41

Liste des sigles et acronymes

N°	Sigle	Signification	Description
1	3DCityDB	3D City Database	Base de données relationnelle open source pour le stockage et la gestion des modèles CityGML.
2	BIM	Building Information Modeling	Modélisation des informations du bâtiment.
3	BU	Business Unit	Unité d'affaires.
4	DB	DataBase	Base de données.
5	DSI	Direction des Systèmes d'Information	Service en charge de la gestion informatique d'une organisation.
6	EPSG	European Petroleum Survey Group	Système de codification des systèmes de coordonnées géographiques.
7	ETAFAT	Bureau d'Études Techniques d'Ingénierie et de Topographie (Maroc)	Société marocaine spécialisée en topographie, ingénierie et cartographie.
8	GDAL/OGR	Geospatial Data Abstraction Library / OGR Simple Features Library	Bibliothèque open source pour la conversion et le traitement des formats géospatiaux.
9	GML	Geography Markup Language	Langage XML pour la représentation d'objets géographiques.
10	IF	Identifiant Fiscal	Identifiant unique attribué aux entreprises au Maroc.
11	ICE	Identifiant Commun de l'Entreprise	Numéro d'immatriculation attribué aux entreprises au Maroc.
12	KML	Keyhole Markup Language	Format XML pour la visualisation de données géographiques (ex. Google Earth).
13	LOD	Level of Detail	Niveau de détail dans les modèles CityGML (LOD0 à LOD4).
14	LiDAR	Light Detection and Ranging	Technique de télédétection produisant un nuage de points 3D.
15	OGC	Open Geospatial Consortium	Organisme international de normalisation pour les standards géospatiaux.
16	PGAdmin	PostgreSQL Administration Tool	Outil graphique d'administration de PostgreSQL.
17	PostGIS	PostgreSQL Geographic Information System	Extension spatiale de PostgreSQL permettant de stocker et traiter des données géographiques.
18	QGIS	Quantum GIS	Système d'information géographique open source.

19	RC	Registre de Commerce	Numéro d'enregistrement légal des entreprises au Maroc.
20	SARL	Société à Responsabilité Limitée	Forme juridique d'entreprise.
21	SIG	Système d'Information Géographique	Ensemble d'outils pour gérer et analyser des données spatiales.
22	SQL	Structured Query Language	Langage de requête structuré pour bases de données relationnelles.
23	UML	Unified Modeling Language	Langage de modélisation unifié (schémas conceptuels).
24	UTM	Universal Transverse Mercator	Système de projection cartographique universel transverse de Mercator.
25	VS Code	Visual Studio Code	Environnement de développement open source de Microsoft.
26	WGS84	World Geodetic System 1984	Système géodésique mondial de référence.
27	XML	Extensible Markup Language	Langage de balisage extensible utilisé pour structurer des données.
28	PFA	Projet de Fin d'année.	En deuxième(2 ^e) année du cyce Ingénieur

Liste des figures

Figure 1: logo de docker.....	15
Figure 2: Logo de postgreSQL.....	15
Figure 3: Logo de 3DCityDB.....	15
Figure 4:Logo de 3DCityDB Importer/Exporter	15
Figure 5: Logo de QGIS	16
Figure 6: logo de GDAL	16
Figure 7: Logo de Kit Model viewer.....	16
Figure 8: Logo de Cloud compare	16
Figure 9: Logo de VS code	16
Figure 10: Logo de Rhino city.....	16
Figure 11: Nuage de point en entier (avec image satellite en haut)	19
Figure 12: Decoupage du nuage de points des bâtiments concernés	19
Figure 13: Shapefile des bâtiments de la zone d'étude.	20
Figure 14: Shapefile des dimensions des bâtiments	21
Figure 15: Application web open source en ligne pour le filtrage du nuage de points. ...	22
Figure 16: traitement du nuage de points avec cloud compare.	23
Figure 17: UML du projet/ fichier CityGML et Base de donnees	24
Figure 18: Extrait du code GML sous VS code	25
Figure 19: Services nécessaires activés sous docker	26
Figure 20: importation du fichier dans la base en ligne de commande.....	27
Figure 21: consultation des données de la base avec pgadmin.....	28
Figure 22: nombre de lignes du fichier GML.....	29
Figure 23: : Worflow mise en place.	30
Figure 24: Vue entiere du bâtiment.	31
Figure 25: Vue perspectives du bâtiment	32
Figure 26: Arriere du Bâtiment.....	32
Figure 27: Autre Face du Bâtiment.....	33
Figure 28: Vue du bas du Bâtiment.	33
Figure 29: Vue de la charpente du bâtiment.	34
Figure 30: Illustration allégée des niveaux de détails.....	35
Figure 31: Exemple du cityGML appliquée à une ville.	36
Figure 32: Organisation temporelle du stage/ diagramme de gantt	41
Figure 33: Resumé de la presentation de l'entreprise.....	42
Figure 34: page de téléchargement de QGIS.....	43
Figure 35: Interface de QGIS	44
Figure 36: page de telechargement de cloud compare	44
Figure 37: Interface de cloud compare	45
Figure 38: téléchargement de Visual Studio Code.....	45

Figure 39: Interface de Vs code	46
Figure 40: Téléchargement de docker	47
Figure 41: Interface de docker.....	47
Figure 42: Téléchargement de 3DcityDb sur Github	48
Figure 43: ouverture du dossier dans un terminal	48
Figure 44: extrait du contenu du fichier docker-compse.yml	49
Figure 45: Interface de Docker avec les services lancés.....	49
Figure 46: Interface de connexion à PG admin	50
Figure 47: Connexion à la base de données 3D city DB	51
Figure 48: Consultation de la BD qui contient bien les schéma adéquats.	51
Figure 49: Telechargement de Kit Model Viewer.....	52
Figure 50: Ouverture d'un fichier CityGML sur KIT model viewer	52
Figure 51: Fichier Ouvert sur Kit Model Viewer	53

1. Introduction

Le développement urbain et l'aménagement du territoire reposent de plus en plus sur l'intégration de données géospatiales précises et sur la modélisation tridimensionnelle des villes. Les maquettes numériques 3D géoréférencées se présentent comme des outils essentiels pour la planification urbaine, la simulation environnementale, l'analyse spatiale et la gestion des infrastructures. Elles permettent de représenter de manière réaliste le bâti, les infrastructures et les objets urbains, offrant ainsi un support précieux aux décideurs et aux professionnels du secteur.

Dans ce contexte, l'objectif de ce stage est de mettre en place une chaîne de production open source permettant de transformer des données géographiques brutes (shapefiles, nuages de points, geopackages) en modèles 3D conformes au standard **CityGML**, stockés dans une base de données spatiale 3D (3DCityDB) et exploitables pour visualisation et analyse. Cette démarche inclut le nettoyage et la préparation des données, la conception UML pour structurer les informations, l'édition et la validation des fichiers CityGML, l'importation dans la base de données, ainsi que l'exploitation des modèles à travers des outils de visualisation 3D.

Le stage s'inscrit ainsi dans une perspective de valorisation des solutions open source pour l'ingénierie urbaine et la gestion géospatiale. Il vise à explorer les possibilités offertes par les standards ouverts et les environnements modulaires, tout en évaluant les limites et les besoins en automatisation pour des projets à plus grande échelle.

Ce rapport est structuré de manière à présenter :

- L'entreprise d'accueil, ses missions et son positionnement dans le domaine de l'ingénierie géospatiale ;
- L'état de l'art des standards et outils de modélisation urbaine 3D ;
- La méthodologie suivie, les outils utilisés et les étapes du processus de création de la maquette 3D ;
- Les résultats obtenus, leur discussion et les limites identifiées ;
- La conclusion et les perspectives de développement pour l'intégration des maquettes 3D dans les projets urbains.

2. Présentation de l'entreprise d'accueil

2.1 Identification et situation

- **Nom et statut juridique :** ETAFAT – Bureau d’Études Techniques d’Ingénierie et de Topographie, SARL au capital de 3 000 000,00 DH.
- **Siège social :** Lot 57, Lottissement Salaj, Aïn Diab, Casablanca, Maroc.
- **Contacts :** Tél. 05 22 79 87 00 / 05 22 79 87 01 ; Fax 05 22 79 81 09.
- **Informations légales :** ICE 001526391000027, RC 042807, IF 0103.
- **Date de création :** 1983, avec une longue expérience dans le domaine de l’ingénierie géospatiale.

2.2 Domaines d’activité et expertise

ETAFAT est spécialisée dans plusieurs domaines complémentaires de l’ingénierie territoriale et géospatiale :

1. **Foncier et aménagement du territoire**
 - a. Études topographiques et foncières pour la planification urbaine et rurale.
 - b. Gestion et valorisation du patrimoine foncier.
2. **Systèmes d’information géographique (SIG) et ingénierie de données**
 - a. Collecte, traitement et analyse de données géospatiales.
 - b. Création de bases de données et modélisation 3D.
3. **Topographie et cartographie avancée**
 - a. Acquisition de données via drones, scanners laser, GPS et autres technologies de relevé.
 - b. Cartographie numérique et modélisation 3D des infrastructures et du territoire.
4. **Conseil en ingénierie et expertise digitale**
 - a. Assistance technique et stratégie pour les projets territoriaux.
 - b. Intégration de solutions open source et innovantes dans les workflows professionnels.

2.3 Positionnement sur le marché et projets récents

- **Positionnement :** Acteur majeur de l’ingénierie géospatiale au Maroc, reconnu pour l’expertise technique et l’utilisation de technologies avancées.
- **Projets nationaux :** Participation à des infrastructures majeures, telles que les études préliminaires topographiques du gazoduc Nigeria-Maroc.

- **Projets internationaux** : Présence dans certains pays africains, avec des filiales ou collaborations en Côte d'Ivoire et au Sénégal.
- **Technologies employées** : Drones, scanners laser, SIG, CityGML et autres solutions numériques pour garantir des prestations de qualité et conformes aux standards internationaux.

2.4 Organisation interne et rôle du département SIG

1. Organisation générale :

- a. Plusieurs pôles d'expertise couvrant l'ingénierie, le conseil, la cartographie, le foncier et les données spatiales.
- b. Une équipe d'environ 170 collaborateurs composée d'ingénieurs et de techniciens spécialisés.

2. Département SIG : Le département **Systèmes d'Information Géographique (SIG)** constitue un pilier central dans l'activité d'ETAFAT. Il met son savoir-faire au service de nombreux donneurs d'ordres publics et privés, au Maroc comme en Afrique : collectivités locales, organismes publics, gestionnaires de réseaux, bureaux d'études ou encore structures de gestion urbaine. Ses principales missions sont :

- **Collecte, traitement et modélisation des données géospatiales**, en 2D et 3D, avec intégration dans des bases spécialisées comme **3DCityDB**.
- **Élaboration et développement de solutions SIG complètes**, allant de l'étude des besoins jusqu'à la mise en service et la maintenance des outils.
- **Applications multiples** : administration foncière et gestion urbaine, diagnostic territorial, protection du patrimoine, gestion des risques naturels, gestion des actifs imposables, etc.
- **Accompagnement des utilisateurs** grâce à des prestations sur-mesure ou des packs clé-en-main, adaptés aux spécificités de chaque métier.

3. Rôle stratégique du département SIG :

Au-delà de son expertise technique, le département SIG occupe une place **stratégique** au sein de l'entreprise :

- **Innovation et modernisation** : il intègre des solutions numériques et open source (QGIS, PostGIS, CityGML, 3DCityDB, ...) dans les workflows professionnels, réduisant les coûts et augmentant la compétitivité.
- **Garantie de qualité et d'interopérabilité** : il assure la cohérence, la fiabilité et la compatibilité des données géospatiales, condition essentielle pour sécuriser les projets d'aménagement et d'ingénierie.

- **Création de valeur pour les clients** : en maîtrisant toute la chaîne (étude, conception, intégration, maintenance), il offre des solutions durables et adaptées, renforçant la confiance et la fidélité des partenaires.
- **Positionnement stratégique sur le marché africain** : en accompagnant divers acteurs publics et privés, ETAFAT consolide son rôle de **référence régionale en ingénierie géospatiale et SIG**.

3. Contexte et état de l'art

3.1. La modélisation urbaine 3D et ses enjeux

La modélisation 3D des villes s'impose progressivement comme un outil indispensable pour les acteurs de l'aménagement du territoire, de l'urbanisme, de l'ingénierie et de la gestion urbaine. Une maquette numérique urbaine permet de représenter le bâti, les infrastructures et les objets urbains de manière réaliste et géoréférencée, facilitant ainsi :

- La planification urbaine et le suivi de l'expansion des villes,
- La simulation environnementale (ensoleillement, bruit, pollution, consommation énergétique),
- La gestion patrimoniale des infrastructures,
- L'analyse de risques (inondations, séismes),
- Le développement de services innovants dans le cadre des smart cities.

Ces besoins exigent des données fiables, des standards ouverts et des chaînes de production reproductible. Dans ce contexte, **CityGML** s'est imposé comme le standard de référence pour l'échange et la représentation des modèles urbains 3D.

Par ailleurs, certains départements, comme le **département SIG** de l'entreprise, utilisent des logiciels propriétaires tels que **Rhinocity** pour la génération de modèles 3D urbains. Ce logiciel, bien que non open source, permet de produire des maquettes précises adaptées aux exigences locales. Cependant, il n'a **pas été utilisé dans le cadre de ce stage**, qui s'appuie exclusivement sur des outils open source.

3.2. Le standard CityGML

CityGML est une norme internationale développée par l'OGC (Open Geospatial Consortium). Elle définit un schéma conceptuel et un encodage XML/GML pour la représentation de modèles urbains tridimensionnels.

Principales caractéristiques :

- **Hiérarchie thématique** : bâtiments, terrains, infrastructures, végétation, mobilier urbain, etc.
- **Niveaux de détail (LOD)** : du LOD0 (empreintes au sol) au LOD4 (détails intérieurs des bâtiments).
- **Sémantique enrichie** : chaque objet 3D porte des attributs (fonction, hauteur, date de construction, matériaux, etc.) en plus de sa géométrie.
- **Interopérabilité** : format ouvert compatible avec de nombreux SIG et bases de données.

Dans le cadre de ce stage, l'accent a été mis sur le **LOD3** (Bâtiment illustratif avec détail architecturaux) et sur la structuration dans un fichier **CityGML** conforme.

3.3. Les données sources : shapefiles des bâtiments, geopackage, nuage de points

La modélisation CityGML repose sur des données géographiques de base. Pour la ville de **Laâyoune**, les données initiales provenaient principalement de **shapefiles** représentant l'empreinte des bâtiments.

Caractéristiques des données sources :

- Format **ESRI Shapefile (.shp)**, largement utilisé dans le SIG.
- Contenu : polygones représentant les empreintes au sol des bâtiments.
- Système de coordonnées initial : vérifié et reprojeté en **EPSG :32629 (UTM zone 29N, WGS84)** et **EPSG :26194(Merchich nord sahara)** pour garantir un géoréférencement correct.
- Limites : absence d'altitude et de métadonnées sémantiques détaillées (usage du bâtiment, matériaux, etc.).

Ces données brutes nécessitaient donc un **prétraitement** avant leur conversion vers CityGML.

3.4. Outils et logiciels utilisés pour la chaîne de production

Différents logiciels et outils open source ont été utilisés pour assurer la conversion, le traitement et la gestion des données géospatiales :

Docker

Docker est un outil de virtualisation légère qui permet d'exécuter des applications dans des conteneurs isolés, avec toutes leurs dépendances. Il a été utilisé pour déployer **PostgreSQL/PostGIS** et **3DCityDB** dans un environnement stable et reproductible, sans conflits de configuration. Concrètement Docker nous invite l'installation et la configuration locales des logiciels.



Figure 1: logo de docker

PostgreSQL/PostGIS (sous Docker)

PostgreSQL est un système de gestion de base de données relationnelle, et PostGIS est son extension spatiale qui permet de gérer des données géographiques. Dans ce stage, ces outils ont été utilisés pour stocker les données préparées (shapefiles, geopackages) et pour effectuer des requêtes spatiales nécessaires à l'organisation et à la validation des données avant leur intégration dans 3DCityDB.



Figure 2: Logo de postgresSQL

3DCityDB (sous Docker)

3DCityDB est une base de données relationnelle open source conçue pour stocker et gérer des modèles **CityGML**. Elle permet de structurer les objets 3D en tables normalisées et de conserver les niveaux de détail (LOD). L'utilisation de 3DCityDB sous Docker assure un environnement stable et facilite l'importation et l'exportation des fichiers CityGML via l'outil 3DCityDB Importer/Exporter.



Figure 3: Logo de 3DCityDB

3DCityDB Importer/Exporter ou citydb-tool (version récente) dans ce cas

Outil Java fournissant une interface graphique pour importer un fichier CityGML dans la base 3DCityDB et Exporter le modèle final vers différents formats (CityGML, KML, COLLADA ou glTF). Il a été utilisé pour valider la structure des modèles et produire des fichiers exploitables pour la visualisation.



Figure 4:Logo de 3DCityDB Importer/Exporter

QGIS

Logiciel SIG open source utilisé pour le prétraitement des données : visualisation et correction des shapefiles, reprojection des données, export vers d'autres formats.



Figure 5: Logo de QGIS

GDAL/OGR

Bibliothèque open source essentielle pour la conversion et la manipulation des formats géospatiaux, utilisée pour automatiser certaines transformations de données.



Figure 6: logo de GDAL

Kit Model Viewer (FZK Viewer)

Logiciel de visualisation 3D permettant de contrôler la qualité des modèles, vérifier la cohérence des données et générer des rendus exploitables.



Figure 7: Logo de Kit Model viewer

CloudCompare

Outil open source pour le traitement des nuages de points et des modèles 3D, permettant le nettoyage et la comparaison des données avant intégration.

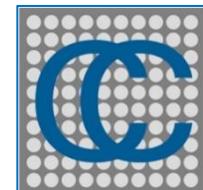


Figure 8: Logo de Cloud compare

VS Code

Environnement de développement utilisé pour l'édition de scripts et le traitement de fichiers intermédiaires (GML, GeoJSON), facilitant l'automatisation de certaines étapes.



Figure 9: Logo de VS code

Rhinocity

Logiciel propriétaire utilisé par l'entreprise pour la génération de modèles 3D urbains. Il n'a **pas été utilisé dans le cadre de ce stage**, mais son existence illustre les pratiques courantes de l'entreprise pour la mise en place des modèles 3D.



Figure 10: Logo de Rhino city

3.5. État de l'art des approches de production CityGML

Deux approches principales permettent de générer un modèle **CityGML**, chacune présentant des avantages et des limites :

Approche automatisée (scripts Python, GeoPandas, lxml, ... - outils propriétaires) :

- Cette méthode repose sur l'extraction et la transformation automatique des données géospatiales en fichiers CityGML.
- Elle est rapide et peut traiter de grands volumes de données en peu de temps.
- Cependant, elle nécessite une expertise technique avancée pour garantir la conformité au standard CityGML, notamment dans la structuration XML/GML.
- Les outils utilisés dans cette approche sont souvent propriétaires ou non open source, ce qui limite leur flexibilité et leur reproductibilité dans un contexte académique ou open source.

Approche manuelle structurée (édition GML et importation dans 3DCityDB via outils open source) :

- Cette méthode consiste à préparer et structurer les fichiers GML manuellement ou semi-automatiquement avant leur importation dans une **base de données 3DCityDB**.
- L'utilisation d'outils open source tels que **QGIS, GDAL/OGR, PostgreSQL/PostGIS et 3DCityDB Importer/Exporter** permet un contrôle rigoureux des données et assure leur conformité avec les standards CityGML.
- Cette approche garantit la production d'un modèle **robuste, précis et conforme**, notamment pour les niveaux de détail LOD1 utilisés dans ce stage.
- La préférence pour cette approche s'explique par l'exigence de reproductibilité et d'ouverture des logiciels, ainsi que par la nécessité de s'assurer que chaque objet 3D est correctement structuré et géoréférencé.

Dans ce stage, la **seconde approche** a été adoptée afin de garantir la **qualité et la conformité** du modèle produit. L'ensemble de la chaîne de production reposait sur des logiciels **open source**, ce qui a permis un contrôle complet sur le traitement des données. **Rhino city**, bien qu'utilisé par le département SIG de l'entreprise pour la génération de modèles 3D, n'a pas été employé dans le cadre de ce stage et reste hors de cette chaîne de production.

4. Méthodologie suivie

La méthodologie adoptée dans le cadre de ce stage repose sur la mise en place d'un pipeline **open source** permettant la création d'une maquette 3D cityGML géoréférencée à partir de données existantes. Étant donné que les solutions open source de transformation automatique des données vers le format **CityGML** ne sont pas open source, une approche **manuelle** qui est gratuite a été retenue. Celle-ci, bien que complexe et chronophage, a permis de construire un prototype sur un bâtiment unique, choisi à titre d'exemple. La démarche se décompose en quatre grandes étapes principales :

4.1 Analyse et préparation des données sources

La première étape du processus a consisté à analyser et préparer les données disponibles afin de disposer d'un jeu cohérent et adapté à la zone d'étude. Cette zone correspond à un **petit quartier situé à proximité du stade Moulay Rachid de Laâyoune**. Les données initiales étaient constituées :

- D'un **nuage de points LiDAR (modèle numérique de surface et de terrain)**, riche en informations tridimensionnelles,
- D'un **géopackage** contenant différents shapefiles (bâtiments, routes, limites administratives, etc.).
- D'une image satellite de la zone d'étude (image marquant les limites de la zone d'étude considérée)

Ces données ont fait l'objet de plusieurs traitements successifs décrits ci-après.

4.1.1 Découpage de la zone d'étude

Afin de restreindre le travail au seul quartier choisi, un **découpage spatial** a été réalisé :

- Dans **QGIS**, la zone d'étude a été délimitée par un polygone englobant le quartier ciblé.
- Les couches du geopackage ont ensuite été **découpées (clip)** à l'aide de ce polygone, de manière à ne conserver que les entités pertinentes.
- De la même manière, le **nuage de points** a été découpé pour ne conserver que la partie correspondant à ce quartier.

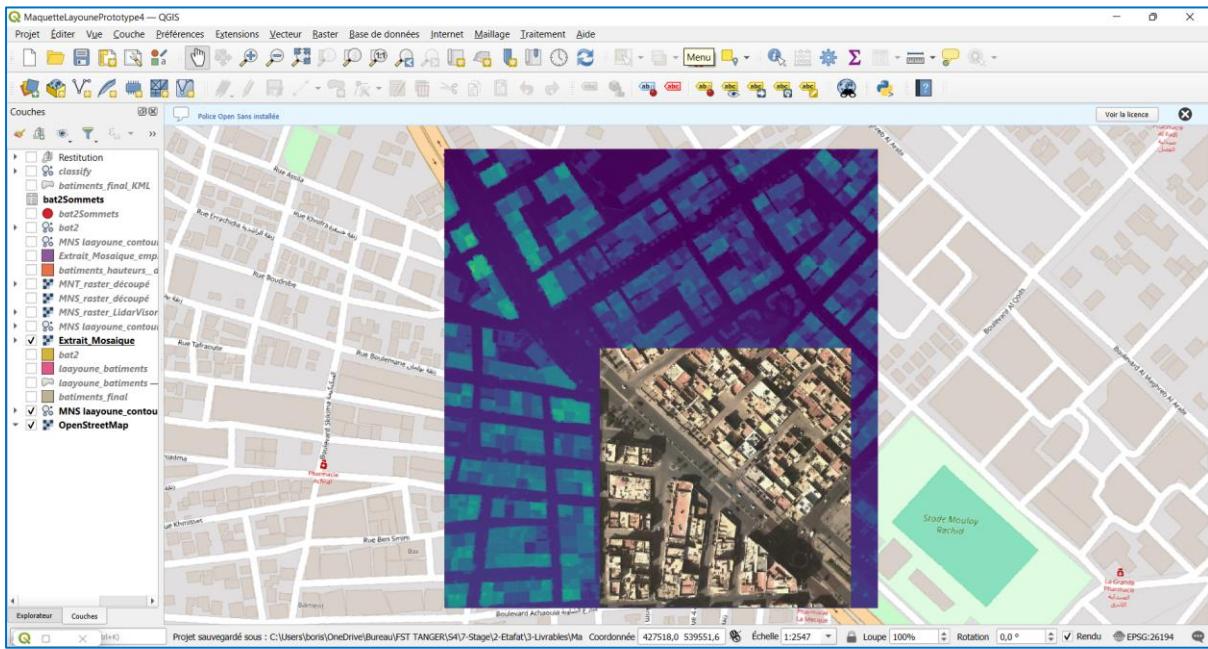


Figure 11: Nuage de point en entier (avec image satellite en haut)

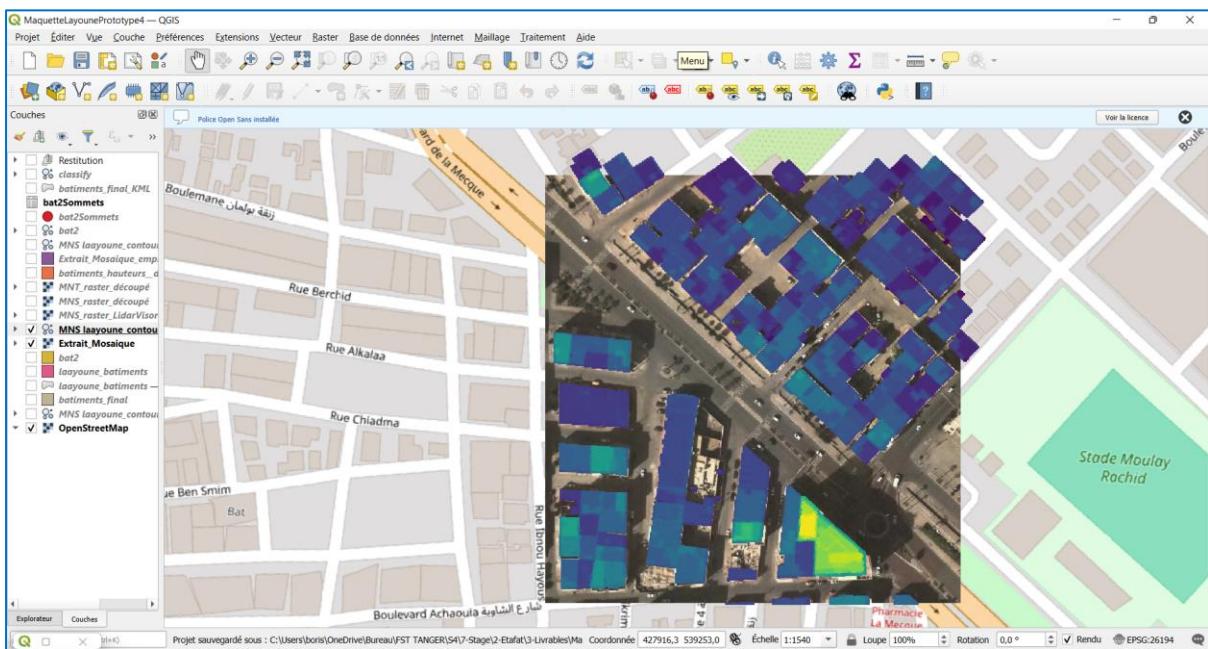


Figure 12: Découpage du nuage de points des bâtiments concernés

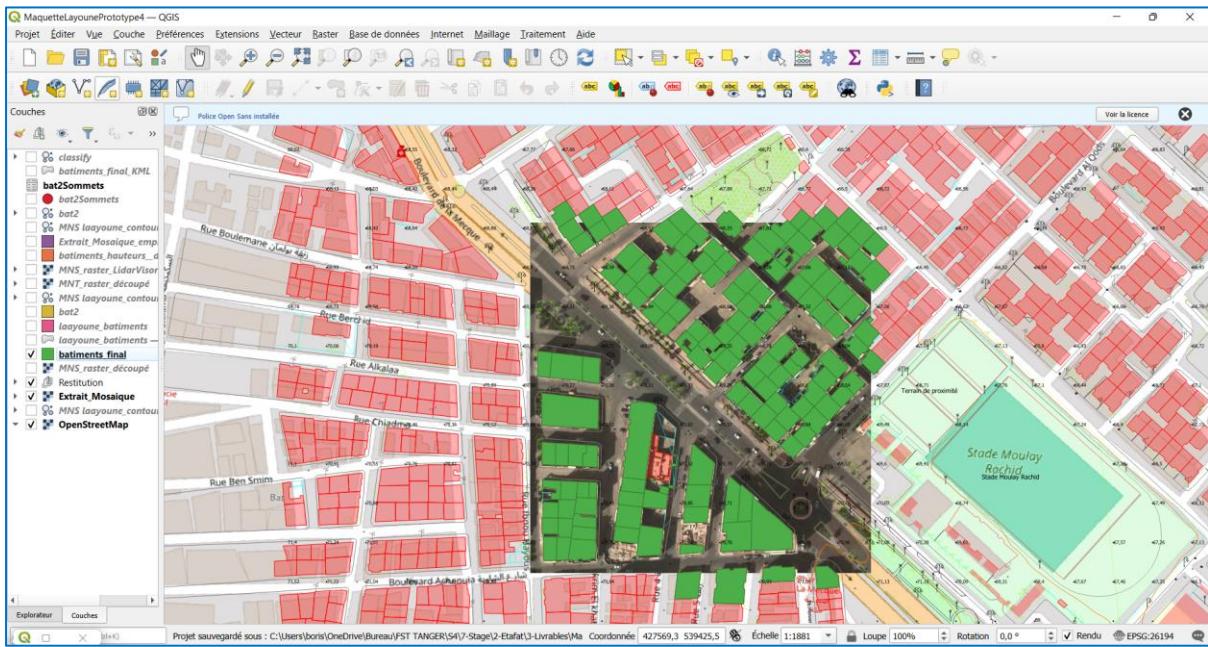


Figure 13: Shapefile des bâtiments de la zone d'étude.

4.1.2 Génération d'un shapefile des dimensions des bâtiments

Une étape importante a ensuite consisté à créer un shapefile contenant les **dimensions approximatives des bâtiments** :

- Les contours des bâtiments ont été extraits à partir du geopackage.
- Le **nuage de points** a servi à estimer les hauteurs en identifiant les points les plus élevés sur chaque bâtiment.
- Ces informations (largeur, longueur, hauteur) ont été intégrées dans un nouveau shapefile décrivant les volumes bâtis du quartier.

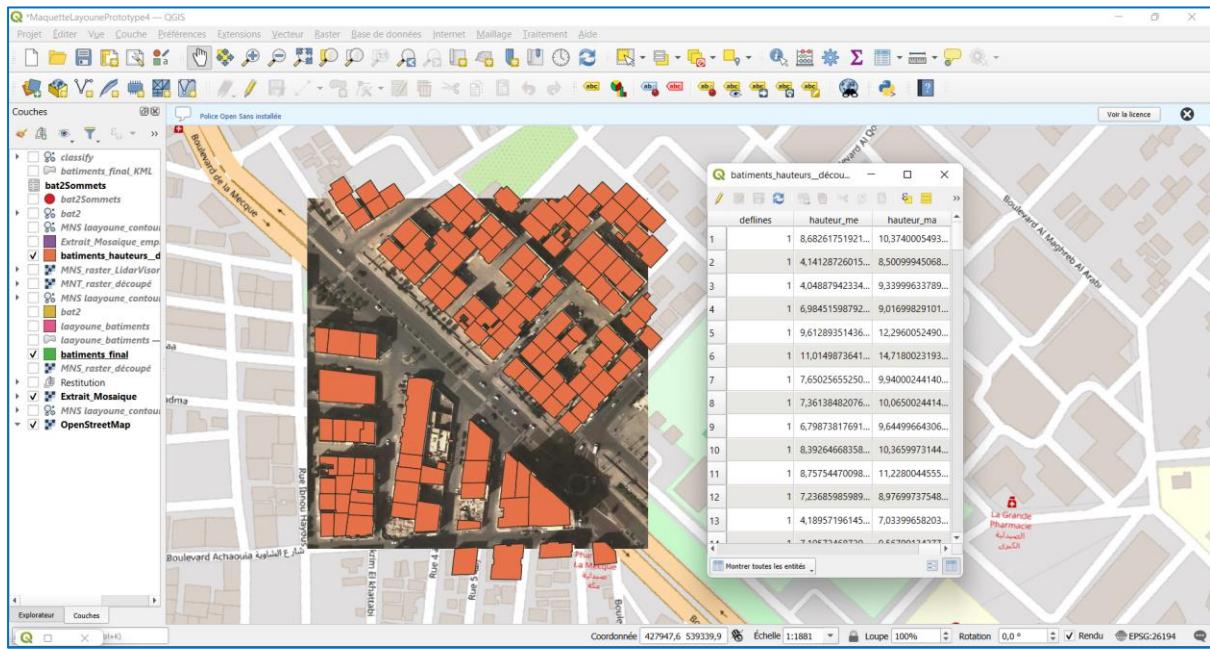


Figure 14: Shapefile des dimensions des bâtiments

Ce shapefile constitue une donnée de référence qui servira à la modélisation CityGML.

4.1.3 Classification du nuage de points

Pour améliorer l’exploitation du nuage de points, une **classification thématique** a été effectuée à l’aide d’un logiciel web externe spécialisé (<https://lidarvisor.com/>). Cette opération a permis d’identifier différentes classes telles que :

- **Bâtiments** ;
- **Sol** ;
- **Végétation** ;
- **Objets divers (mobilier urbain, véhicules, etc.).**

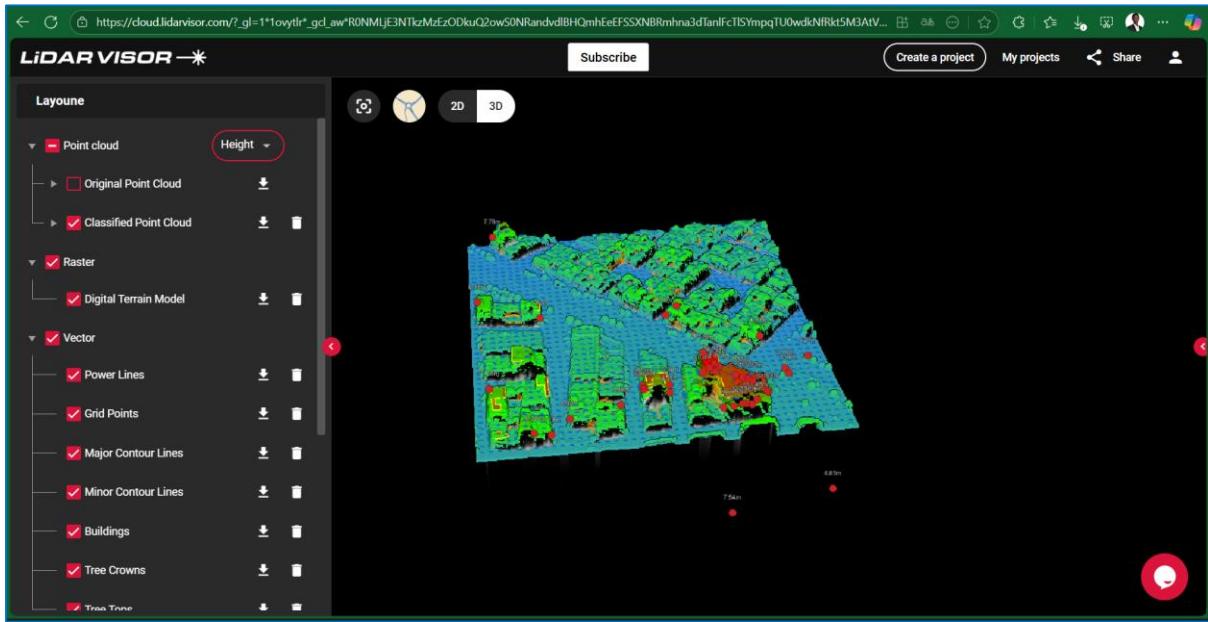


Figure 15: Application web open source en ligne pour le filtrage du nuage de points.

Cette classification facilite le filtrage ultérieur et la sélection des points appartenant uniquement aux structures bâties.

4.1.4 Visualisation et triangulation avec CloudCompare

Enfin, le logiciel **CloudCompare** a été mobilisé pour deux usages principaux :

1. **Visualisation 3D** : inspection qualitative du nuage de points, vérification des découpages et de la classification.
2. **Triangulation** : génération de maillages 3D à partir de sous-ensembles du nuage, afin de mieux comprendre la morphologie du bâtiment étudié.

La triangulation ne vise pas ici à produire un modèle surfacique complet, mais à **aider à la lecture des volumes** et à compléter les mesures géométriques réalisées.

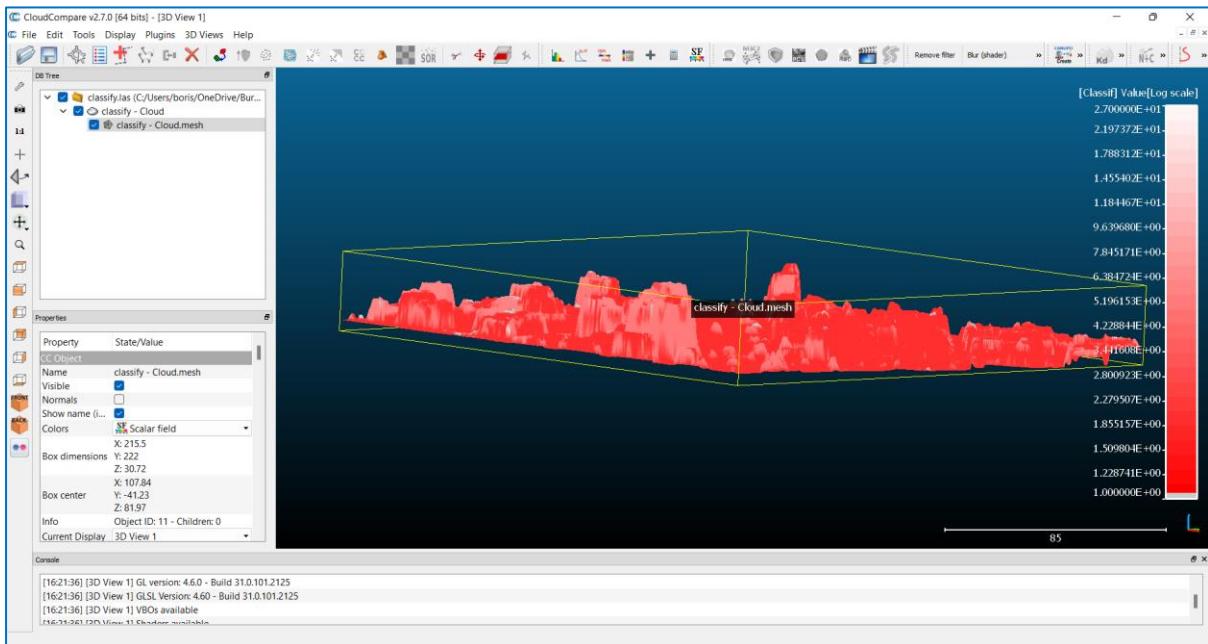


Figure 16: traitement du nuage de points avec cloud compare.

Ainsi, cette étape de préparation des données a permis de disposer :

- D'un nuage de points nettoyé, découpé et classifié ;
- D'un geopackage restreint à la zone d'étude ;
- D'un shapefile détaillant les dimensions des bâtiments ;
- De maillages 3D illustratifs facilitant l'interprétation.

Ces données constituent le socle sur lequel repose la modélisation manuelle ultérieure en CityGML.

4.2 Conception du schéma UML

Une fois les données préparées, la deuxième étape a consisté à concevoir un **diagramme UML** représentant la structure du bâtiment et ses principales entités.

- L'UML a été élaboré à l'aide de Lucidchart, logiciel de modélisation, en tenant compte de la structure attendue dans une base **3DCityDB**.
- Ce diagramme a servi de passerelle conceptuelle entre les données sources (points, vecteurs) et leur traduction dans un modèle **CityGML**, garantissant ainsi la conformité avec les standards de modélisation urbaine.

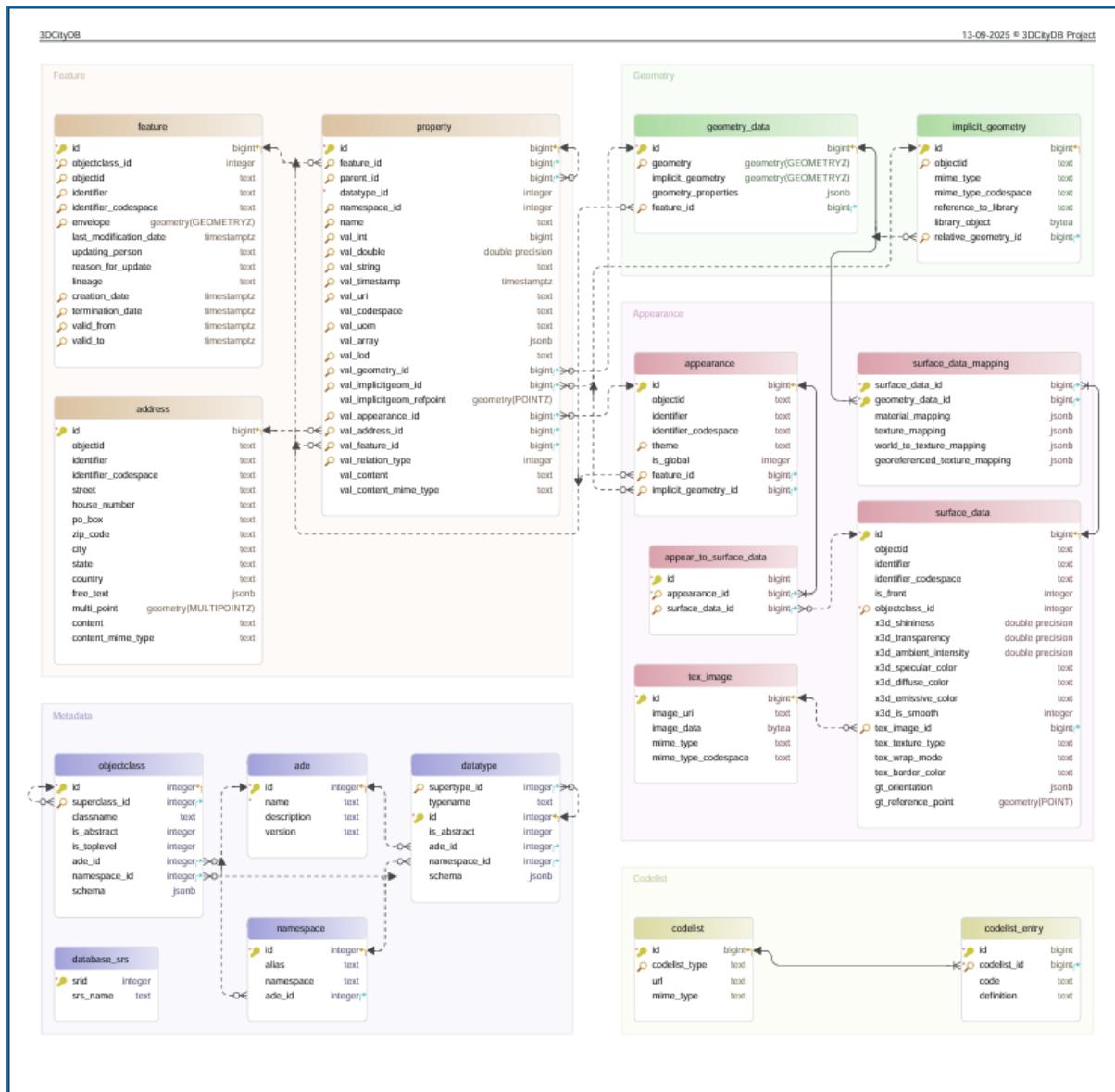


Figure 17: UML du projet/ fichier CityGML et Base de données

4.3 Édition manuelle du fichier CityGML

La troisième étape, et la plus critique, a été la création manuelle du fichier **CityGML** décrivant le bâtiment.

- L'édition a été effectuée avec **Visual Studio Code**, permettant de rédiger directement en **XML** les différentes entités (Building, GroundSurface, WallSurface, RoofSurface, etc.).
- Cette étape nécessite une **compréhension approfondie de la norme CityGML** (structures hiérarchiques, attributs obligatoires, SRS, relations géométriques).
- Chaque dimension et chaque coordonnée mesurée au préalable a été renseignée afin d'assurer un rendu géoréférencé fidèle.

Ce travail fastidieux a mis en évidence la complexité d'une telle approche, qui ne peut raisonnablement être envisagée qu'à titre d'expérimentation ou sur un petit échantillon.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CityModel xmlns="http://www.opengis.net/citygml/3.0" xmlns:con="http://www.opengis.net/citygml/construction/3.0"
  xmlns:bldg="http://www.opengis.net/citygml/building/3.0" xmlns:gml="http://www.opengis.net/gml/3.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.w3.org/1999/xlink"
  xsi:schemaLocation="http://www.opengis.net/citygml/building/3.0 ../../../../../../standard/schema/building.xsd">
  <gml:boundedBy>
    <gml:Envelope srslname="EPSG:25832" srsDimension="3">
      <gml:lowerCorner>514001.2484 5403710.4177 231.8482</gml:lowerCorner>
      <gml:upperCorner>514024.4216 5403731.0224 260.3435</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <bldg:Building gml:id="NeckarStr88">
      <gml:name>Justizausbildungszentrum</gml:name>
      <boundary>
        <con:wallSurface gml:id="NeckarStr88_ws-37">
          <lod0MultiSurface>
            <gml:MultiSurface gml:id="CityDoctor_1596720477835_3">
              <gml:surfaceMember>
                <gml:Polygon gml:id="UUID_1eb35cb5-663a-4c7c-9a83-2bcfa8ec20ad">
                  <gml:exterior>
                    <gml:LinearRing gml:id="CityDoctor_1596720477834_2">
                      <gml:posList srsDimension="3">514024.21905866443 5403717.0265079 257.2635 514020.904458681 5403713.878607884 257.2635 514024.21905866766 5403717.061107894 257.2635
                      257.2635 514024.18295866443 5403717.0265079 257.2635 514020.904458681 5403713.878607884 257.2635 514024.21905866766 5403717.061107894 257.2635
                      </gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:MultiSurface>
          </lod0MultiSurface>
        </con:wallSurface>
      <boundary>
        <con:wallSurface gml:id="NeckarStr88_ws-35">
          <lod0MultiSurface>
            <gml:MultiSurface gml:id="CityDoctor_1596720477835_5">

```

Figure 18: Extrait du code GML sous VS code

4.4 Stockage et exploitation de la maquette 3D

Une fois le fichier **CityGML** finalisé, il est nécessaire de l'intégrer dans une **base de données spatiale** afin de faciliter sa **consultation, son exploitation et son interopérabilité avec différents outils**.

Mise en place de l'environnement avec Docker

Pour ce projet, la base de données a été déployée à l'aide de **Docker**.

- L'utilisation de Docker présente plusieurs avantages :
 - Elle évite d'installer directement sur l'ordinateur des logiciels complexes comme **PostgreSQL/PostGIS** ou **3DCityDB**, qui nécessitent de nombreuses dépendances.
 - Elle garantit un environnement stable, reproductible et facile à partager, quel que soit le système d'exploitation de la machine utilisée.
 - Elle permet d'exécuter plusieurs services isolés (base de données, outils de gestion) à l'intérieur de **conteneurs**.

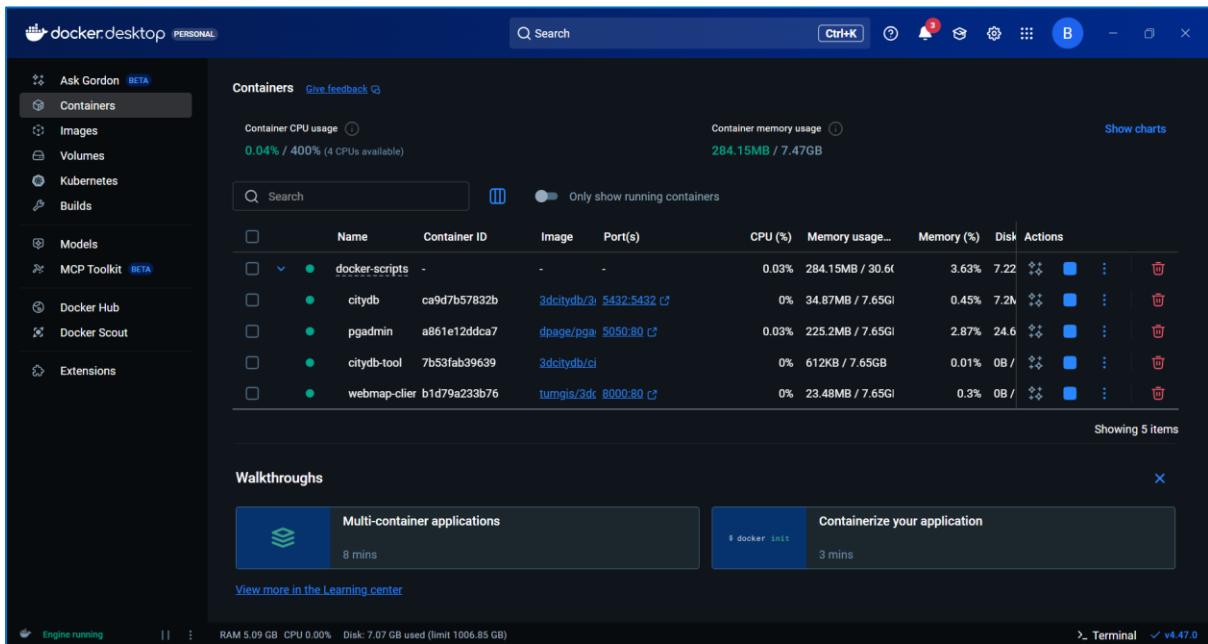


Figure 19: Services nécessaires activés sous docker

Le fichier **docker-compose.yml** joue ici un rôle central. (Retrouvez le contenu de ce fichier dans le manuel d'installation des logiciels dans la dernière page de ce rapport)

- Ce fichier décrit les différents services nécessaires au projet, par exemple :
 - Un conteneur pour **PostgreSQL/PostGIS**,
 - Un conteneur pour l'outil **citydb-tool**, utilisé pour gérer les imports/exports.
- Il définit également la configuration de chaque service (nom, ports utilisés, volumes de données partagés, etc.).
- Grâce à docker-compose, un simple appel de commande (docker compose up -d) lance automatiquement l'ensemble des services nécessaires au projet, sans configuration manuelle complexe.

Import du fichier CityGML

Une fois l'environnement en place, le **fichier CityGML** a été chargé dans la base via l'outil **citydb-tool**, exécuté en ligne de commande.

- L'import se fait en précisant la connexion à la base (hôte, port, utilisateur, mot de passe, base de données) et le chemin du fichier CityGML.
- Par exemple :

```

PS C:\Users\boris\OneDrive\Bureau\FST TANGER\S4\7-Stage\2-Etafat\5-Logiciel\Rapport\3dcitydb-master\postgresql\docker-scripts> docker exec -it citydb-tool citydb import citygml /data/CityGML_3.gml
[10:02:58 INFO] Starting citydb-tool, version 1.1.0.
[10:02:58 INFO] Loading plugins...
[10:02:58 INFO] Executing 'import citygml' command.
[10:03:00 INFO] Found 1 file(s) at /data/CityGML_3.gml.
[10:03:00 INFO] Connecting to database postgres@db:5432/citydb.
[10:03:02 INFO] 3D City Database: 5.1.0
[10:03:02 INFO] DBMS: PostgreSQL 17.5 (Debian 17.5-1.pgdg110+1) (PostGIS 3.5.2, SFCGAL n/a)
[10:03:02 INFO] Connection: postgres@db:5432/citydb
[10:03:02 INFO] Schema: citydb
[10:03:02 INFO] SRID: 4326
[10:03:02 INFO] SRS name: WGS 84
[10:03:02 INFO] SRS URI: urn:ogc:def:crs:EPSG::4326
[10:03:02 INFO] Changelog: disabled
[10:03:02 INFO] Database indexes are on.
[10:03:02 INFO] [111] Importing file /data/CityGML_3.gml.
[10:03:08 INFO] Import summary:
[10:03:08 INFO] bldg:Building: 1
[10:03:08 INFO] bldg:BuildingInstallation: 1
[10:03:08 INFO] con:DoorSurface: 3
[10:03:08 INFO] con:GroundSurface: 1
[10:03:08 INFO] con:OuterCeilingSurface: 4
[10:03:08 INFO] con:OuterFloorSurface: 1
[10:03:08 INFO] con:RoofSurface: 1
[10:03:08 INFO] con:WallSurface: 45
[10:03:08 INFO] con:WindowSurface: 77
[10:03:08 INFO] Total execution time: 11 s.
[10:03:08 INFO] citydb successfully completed.
PS C:\Users\boris\OneDrive\Bureau\FST TANGER\S4\7-Stage\2-Etafat\5-Logiciel\Rapport\3dcitydb-master\postgresql\docker-scripts> |

```

Figure 20: importation du fichier dans la base en ligne de commande

Cette commande demande au conteneur **citydb-tool** d'accéder au fichier CityGML_3.gml (placé dans le volume partagé /data) et de l'intégrer dans la base **3DCityDB**.

Exploitation du modèle 3D

Une fois importé, le modèle est stocké dans le schéma **3DCityDB** de PostgreSQL/PostGIS et peut être exploité de différentes manières :

- **Consultation directe en base** : par des requêtes SQL ou via l'interface **pgAdmin**.
- **Export et visualisation** : grâce à des applications externes comme **KIT Model Viewer**, qui permettent d'explorer la maquette en 3D.
- **Interopérabilité** : la base peut aussi être connectée à des outils SIG (comme QGIS) ou à des applications personnalisées pour extraire, analyser ou réutiliser les données.

The screenshot shows the PgAdmin interface with the following details:

- Object Explorer:** Shows a tree view of database objects including Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (17).
- Query History:** Displays the query: `SELECT * FROM citydb.feature ORDER BY id ASC`.
- Data Output:** A table showing the results of the query. The columns are: id [PK] bigint, objectclass_id integer, objectid text, identifier text, identifier_codespace text, envelope geometry.
- Table Data:** The table contains 402 rows. The first few rows are as follows:

id [PK] bigint	objectclass_id integer	objectid text	identifier text	identifier_codespace text	envelope geometry
1	1	901 NeckarStr88	[null]	[null]	01030000AE61000001000000500000003086FE445F1F41C62B859A0F9D5441F38E537424FB6
2	2	709 NeckarStr88_ws-37	[null]	[null]	01030000AE6100000100000050000000413B4D79935F1F41A1390476109D544123DB97E6A0F
3	3	709 NeckarStr88_ws-35	[null]	[null]	01030000AE61000001000000500000003086FE445F1F41C92B859A0F9D544123DB97E6A0F
4	4	709 NeckarStr88_ws-9	[null]	[null]	01030000AE610000010000005000000023C6F767CSF1F4199B5D49109D544161283C0CA37

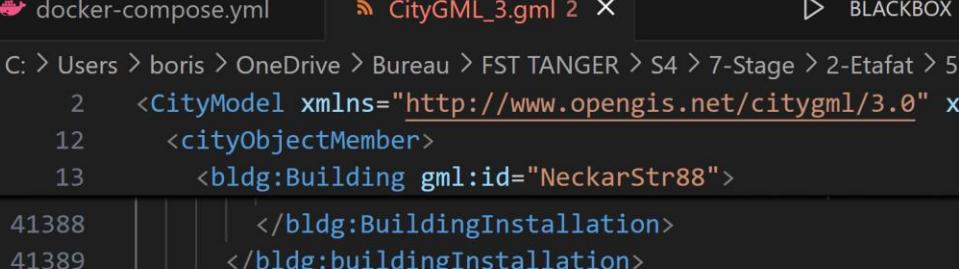
Figure 21: consultation des données de la base avec pgadmin

Cette approche permet donc de disposer d'une infrastructure **modulaire, portable et robuste** pour stocker et exploiter les données CityGML, tout en simplifiant la gestion technique grâce à Docker.

4.5 Limites de l'approche

Il convient de souligner que cette méthodologie, bien que fonctionnelle, présente certaines limites :

- Le **processus manuel** est extrêmement chronophage et peu reproductible à grande échelle (à titre illustratif, le bâtiment présenté contient 41393 lignes de code GML).



```
C: > Users > boris > OneDrive > Bureau > FST TANGER > S4 > 7-Stage > 2-Etage > 5-Logiciel >
    2   <CityModel xmlns="http://www.opengis.net/citygml/3.0" xm
    12     <cityObjectMember>
    13       <bldg:Building gml:id="NeckarStr88">
41388         </bldg:BuildingInstallation>
41389           </bldg:buildingInstallation>
41390             </bldg:Building>
41391               </cityObjectMember>
41392     </CityModel>
41393
```

Figure 22: nombre de lignes du fichier GML.

- Les compétences requises en **CityGML et bases 3D** sont élevées, ce qui limite la facilité d'adoption.
 - Elle a néanmoins permis de **valider la faisabilité technique** et de démontrer le potentiel des solutions open source dans un cadre expérimental.

4.6 Résumé du workflow

1. Traitement des données : La première étape consiste à préparer et nettoyer les données de base. Les fichiers sources (plans, ortho photos, relevés topographiques, etc.) sont vérifiés afin de garantir leur qualité et leur cohérence. On procède à des opérations de mise en forme (projection, harmonisation des formats, correction des erreurs) afin d'obtenir un jeu de données homogène et exploitable pour la suite du projet.

2. Conception UML : Une fois les données prêtes, on passe à la modélisation conceptuelle. À travers un diagramme UML, on définit les entités, leurs relations et leurs attributs. Cette étape permet de représenter la structure logique du futur modèle 3D et de s'assurer que celui-ci respecte les standards CityGML. C'est une phase clé pour garantir la cohérence et l'interopérabilité.

3. Édition du XML/GML : À partir du modèle UML, les objets géographiques sont traduits dans un langage structuré conforme aux normes de l'OGC. On produit des fichiers XML/GML qui décrivent les géométries, les attributs et la hiérarchie des objets. Cette étape transforme la conception théorique en un format lisible par les systèmes informatiques et compatible avec les bases de données spatiales.

4. Importation dans la base de données : Les fichiers CityGML obtenus sont ensuite importés dans la base de données relationnelle (PostgreSQL/PostGIS/3DCityDB) via un outil dédié comme 3DCityDB importer-exporter. Cette intégration permet de stocker efficacement les données 3D, d'assurer leur structuration et d'optimiser leur interrogation. C'est le passage d'un simple fichier statique vers un système dynamique de gestion de l'information géospatiale.

5. Exploitation : Enfin, les données intégrées peuvent être exploitées selon différents besoins : visualisation 3D, analyses spatiales, génération de cartes thématiques ou encore interopérabilité avec d'autres applications SIG et BIM. Cette étape démontre la valeur ajoutée du processus : transformer des données brutes en un outil décisionnel puissant pour l'aménagement et la gestion urbaine. Ici l'exploitation consiste en l'exécution de requêtes SQL sur les données du bâtiment.

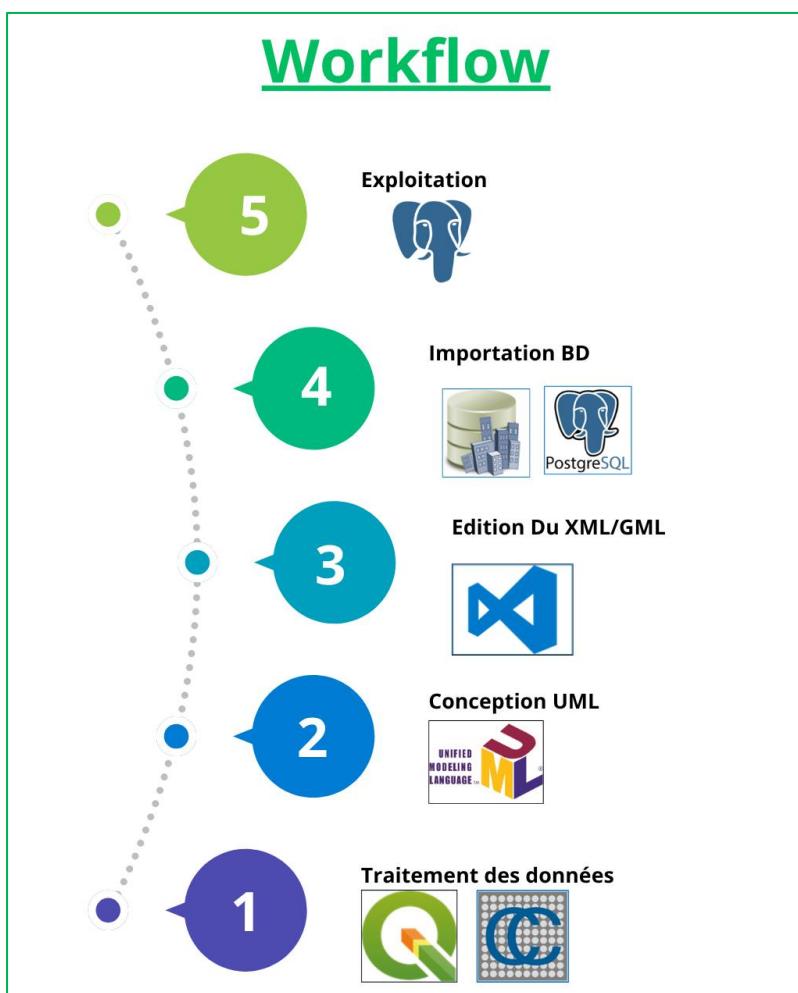


Figure 23: Workflow mise en place.

5. Résultats obtenus

5.1 Bâtiment de référence étudié

Un premier résultat consiste en la modélisation d'un bâtiment choisi comme exemple et disponible en ligne. Ce bâtiment sert de cas d'étude pour illustrer le processus manuel d'édition du fichier CityGML. Sa géométrie est construite en respectant les dimensions, la structure et la position géographique, puis intégrée dans un fichier conforme à la logique 3DCityDB. Ce premier essai montre concrètement la faisabilité de la démarche et met en évidence les étapes nécessaires pour obtenir une maquette 3D cohérente.

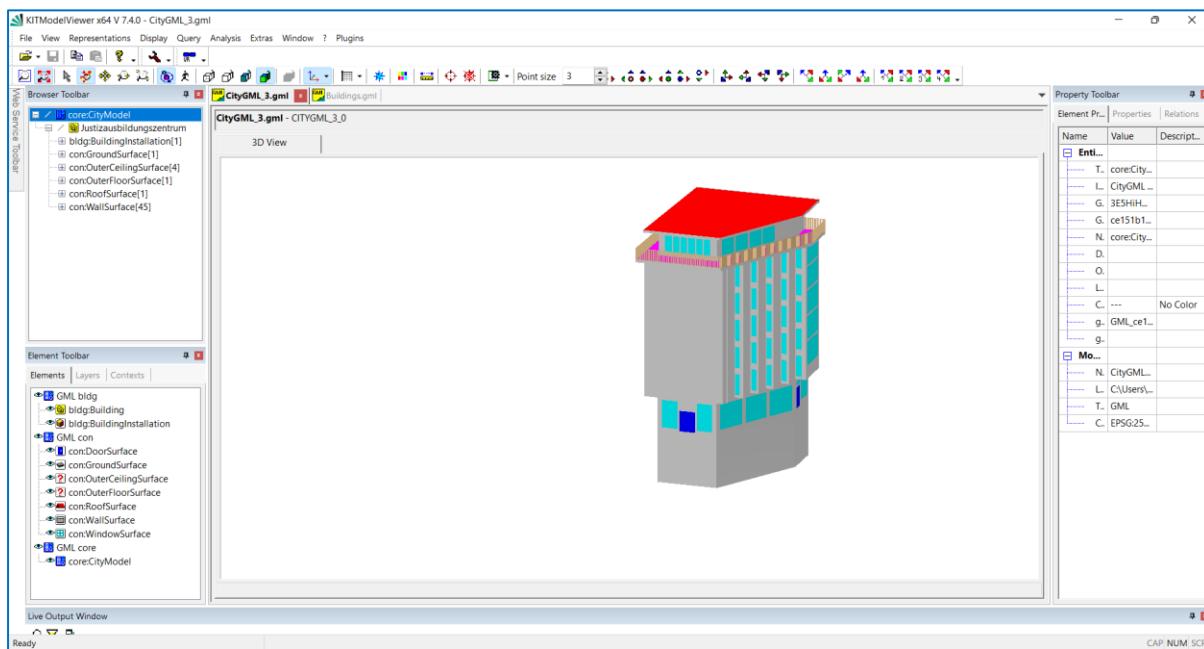


Figure 24: Vue entière du bâtiment.

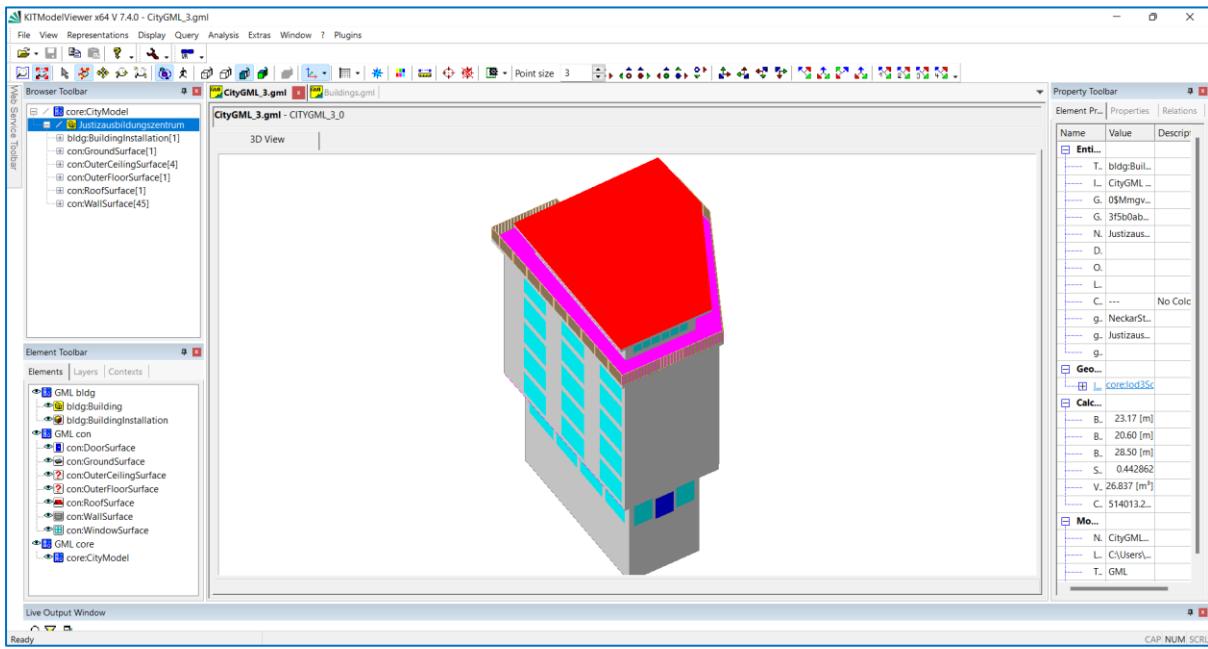


Figure 25: Vue perspectives du bâtiment

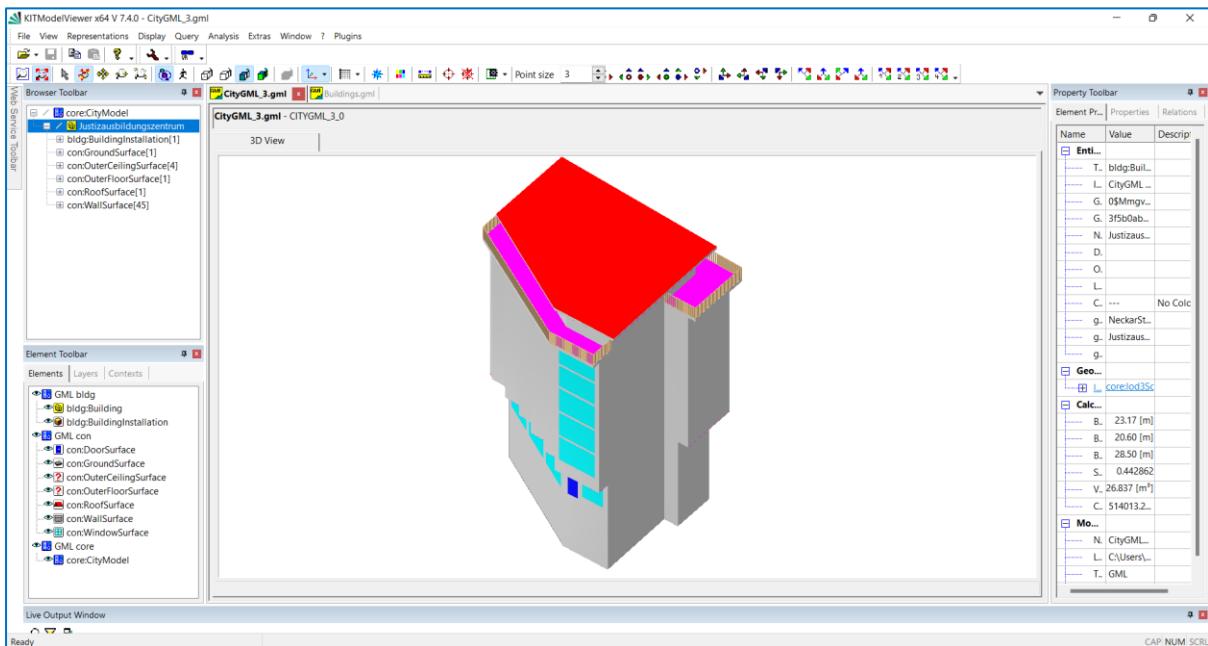


Figure 26: Arriere du Bâtiment

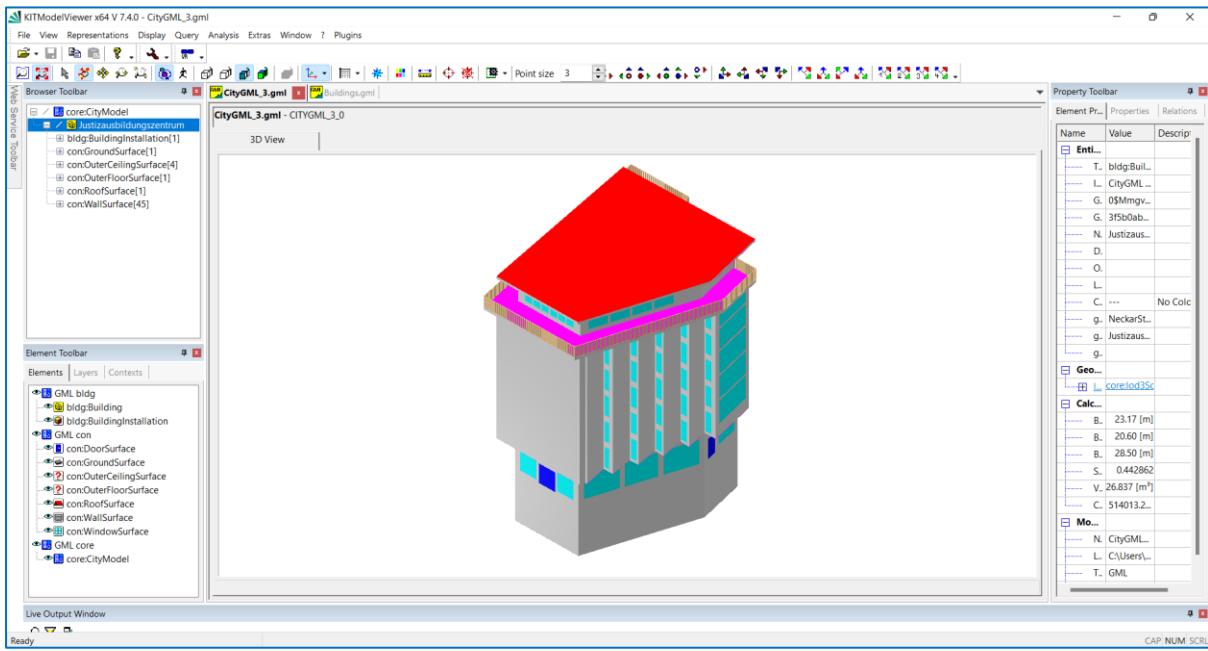


Figure 27: Autre Face du Bâtiment.

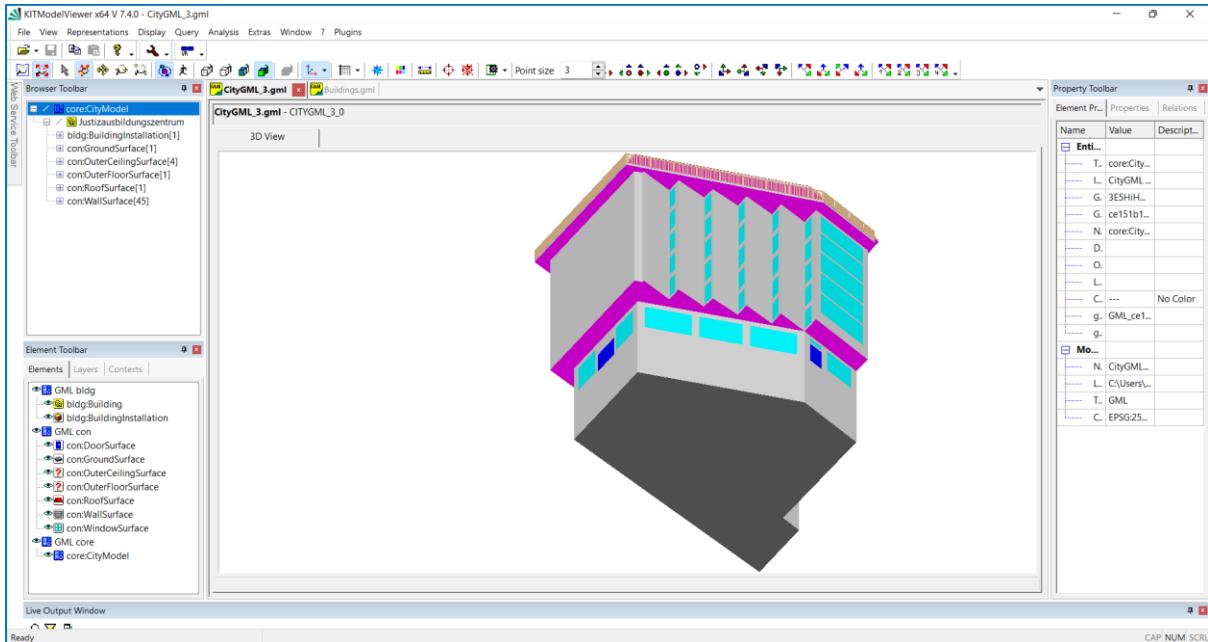


Figure 28: Vue du bas du Bâtiment.

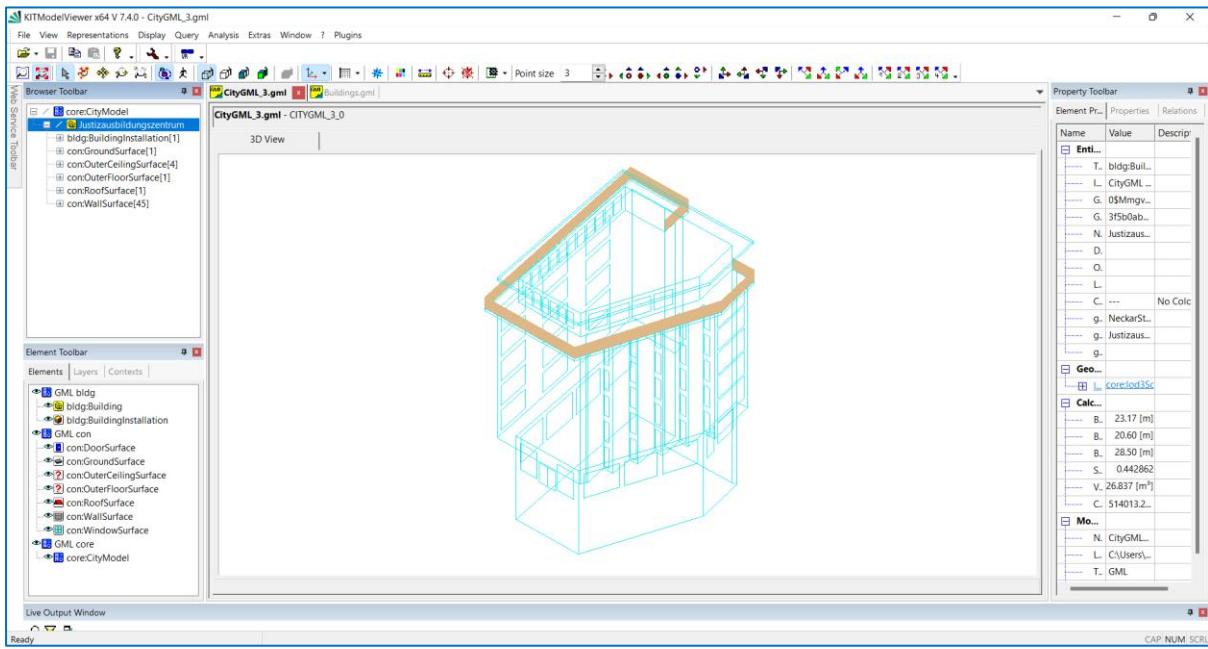


Figure 29: Vue de la charpente du bâtiment.

5.2 Illustration des niveaux de détails (LOD-Level Of Details)

L'un des apports majeurs de CityGML réside dans sa capacité à représenter les objets urbains selon plusieurs niveaux de détails. Les résultats obtenus permettent d'illustrer ces différents LOD :

Le **LOD0 (Level Of Details 0)** concerne l'empreinte au sol des infrastructures ;

Au **LOD1**, les bâtiments apparaissent comme de simples volumes extrudés à partir de leur emprise au sol et de leur hauteur ;

Au **LOD2**, des détails architecturaux tels que la forme des toitures ou certains éléments de façade sont intégrés, offrant une vision plus réaliste du bâti ;

Aux niveaux supérieurs (**LOD3 et LOD4**), le modèle permettrait de représenter des éléments encore plus précis, allant jusqu'aux structures intérieures.



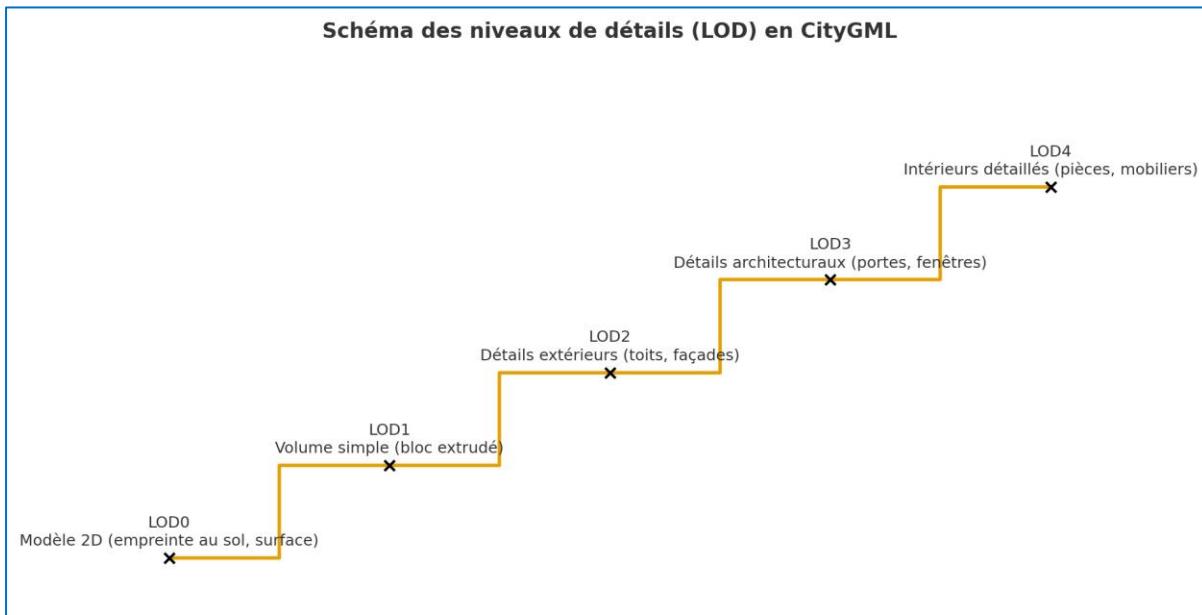


Figure 30: Illustration allégée des niveaux de détails

5.3 Exemple d'une ville entière

Enfin, pour compléter l'étude, un autre exemple est mobilisé à l'échelle d'une ville entière. Ce jeu de données montre comment le CityGML peut structurer et organiser un ensemble d'objets urbains à large échelle. Il met en évidence la capacité du format à gérer la complexité d'un territoire tout en conservant la cohérence entre les différents bâtiments et infrastructures. Cet exemple illustre la puissance du modèle pour des projets urbains de grande envergure, en comparaison avec l'approche manuelle limitée à un bâtiment isolé.

Modèle accessible à travers le lien (<https://kartta.hel.fi/3d/>)

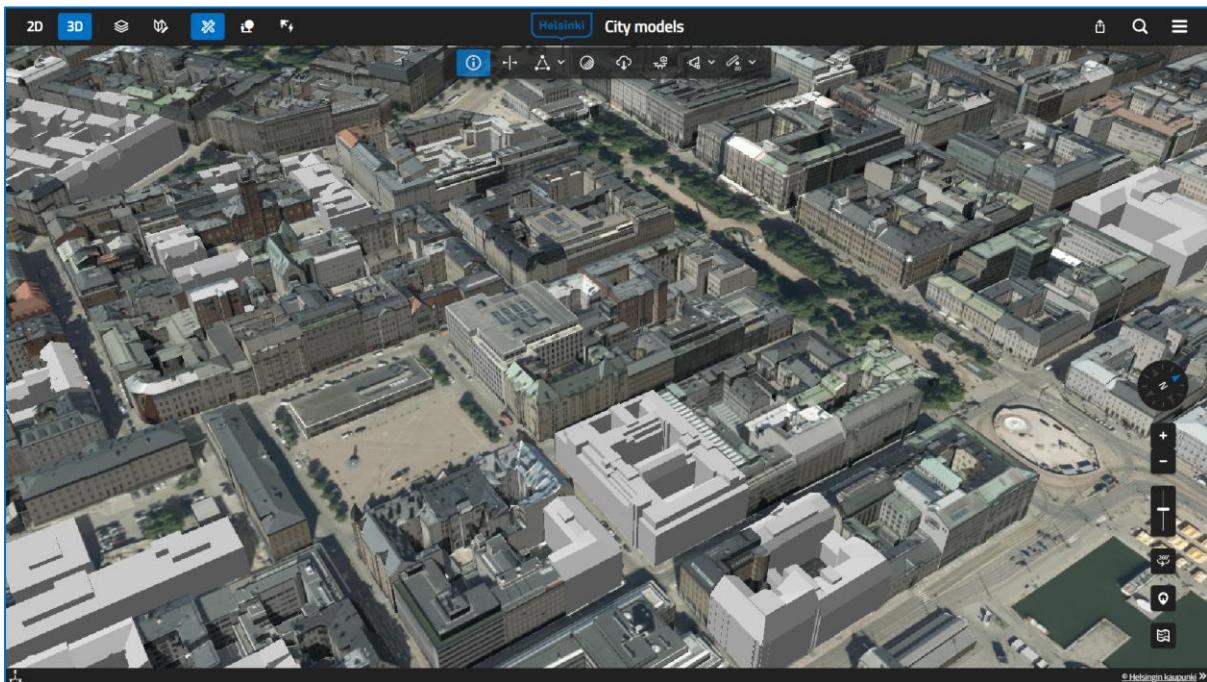


Figure 31: Exemple du cityGML appliquée à une ville.

6. Discussion et limites

L’expérimentation réalisée a permis de démontrer la faisabilité d’une chaîne de production CityGML reposant exclusivement sur des outils open source. Cette approche garantit une transparence méthodologique, une meilleure reproductibilité et une réduction significative des coûts liés aux licences logicielles. Cependant, son caractère manuel et chronophage constitue une limite majeure à une mise en œuvre à grande échelle.

Comparée aux solutions propriétaires comme Rhinocity, utilisées en interne chez ETAFAT, la méthode adoptée présente l’avantage d’une indépendance technologique et d’une flexibilité accrue. En revanche, elle reste moins adaptée aux projets nécessitant des délais serrés ou le traitement de volumes massifs de données.

Du point de vue stratégique, ce stage illustre la complémentarité entre approches manuelles et automatisées. La méthode manuelle offre un contrôle fin et une meilleure compréhension des standards (CityGML, UML, XML/GML), tandis que les approches automatisées (scripts Python, plugins QGIS) constituent une piste incontournable pour l’industrialisation. L’avenir réside probablement dans un **pipeline hybride**, associant automatisation des étapes répétitives et validation manuelle des données critiques.

Pour ETAFAT, ce travail exploratoire ouvre plusieurs perspectives :

- **Renforcement de la compétitivité** grâce à l’intégration d’outils open source dans ses workflows,

- **Positionnement différenciant** sur le marché africain, où les solutions open source sont particulièrement attractives pour des raisons économiques,
- **Développement d'une expertise interne** sur CityGML et 3DCityDB, encore peu maîtrisés par les acteurs locaux,
- **Capacité d'innovation** en s'inscrivant dans les standards internationaux promus par l'OGC.

6.1 Comparaison avec d'autres méthodes

Des alternatives automatiques ont été envisagées, notamment :

- Des **scripts Python** (via les bibliothèques gdal/ogr, pycitygml) pour générer des modèles à partir de shapefiles,
- Des **modules QGIS** (comme QGIS2CityGML) visant à exporter directement des entités en CityGML.

Cependant, ces solutions se sont révélées limitées ou non adaptées au cas d'étude (perte d'information, incompatibilités avec CityGML 3.0, difficulté de personnalisation, non open source).

L'approche manuelle, bien que chronophage, a donc permis de **garantir la conformité au standard** et un meilleur contrôle du modèle produit.

6.2 Limites techniques

Malgré les résultats obtenus, plusieurs limites techniques demeurent :

- **Taille des fichiers** : un modèle CityGML complet peut rapidement atteindre plusieurs mégaoctets, ce qui complique le stockage et l'échange.
- **Erreurs géométriques** : certaines incohérences apparaissent lors de l'import, liées à des polygones mal définis ou à des surfaces non fermées.
- **Non-scalabilité** : la méthode n'est pas applicable à grande échelle (quartier ou ville entière) en raison du temps et de l'effort requis.

6.3 Perspectives d'amélioration

Pour dépasser ces limites, plusieurs pistes d'amélioration sont envisageables :

- **Automatisation partielle** : développer des scripts Python permettant de générer automatiquement les balises CityGML à partir de shapefiles enrichis (dimensions, hauteurs).

- **Utilisation de pipelines hybrides** : combiner QGIS pour la préparation, des scripts de conversion semi-automatique, puis une vérification manuelle ciblée.
- **Optimisation du stockage** : explorer des formats de compression ou des schémas simplifiés pour réduire la taille des fichiers.
- **Extension de l'expérimentation** : appliquer la méthode sur plusieurs bâtiments représentatifs afin de valider sa robustesse.

7. Conclusion

Le stage réalisé au sein du département SIG d'ETAFAT a constitué une immersion dans le domaine en pleine évolution de la modélisation urbaine 3D et de l'usage de standards ouverts tels que CityGML. L'objectif principal consistait à mettre en place une chaîne de production open source pour la modélisation 3D, depuis la préparation des données géospatiales jusqu'à leur intégration dans une base 3D (3DCityDB) et leur visualisation dans des outils dédiés. Les travaux menés ont démontré la faisabilité d'une telle approche. Un pipeline complet a été conçu et documenté : nettoyage et préparation des données, conception UML, édition d'un fichier CityGML, importation dans une base PostgreSQL/PostGIS via Docker, puis exploitation et visualisation de la maquette 3D. L'expérience a mis en évidence la robustesse du standard CityGML, sa richesse sémantique et son potentiel pour l'aménagement urbain et la recherche, mais également la complexité de sa mise en œuvre lorsqu'elle repose uniquement sur une édition manuelle.

Sur le plan technique, ce stage m'a permis d'approfondir ma maîtrise des outils SIG (QGIS, CloudCompare...), des environnements de gestion de bases de données spatiales (PostgreSQL/PostGIS, 3DCityDB), ainsi que des environnements de développement (Visual Studio Code, Docker). Sur le plan professionnel, il a offert une première expérience d'intégration dans un bureau d'études reconnu, en collaboration avec des spécialistes de l'ingénierie géospatiale. Pour l'entreprise, ce stage a représenté une étape exploratoire démontrant le potentiel d'intégration des chaînes open source dans les workflows existants. Il a ouvert des pistes vers le développement de processus plus automatisés, adaptés à des projets de plus grande envergure et reproductibles à large échelle.

En perspective, les travaux pourraient être prolongés par la mise en place de scripts d'automatisation, l'expérimentation sur des zones urbaines plus vastes, ou encore l'exploration d'outils plus récents intégrant le standard CityGML 3.0. L'évolution vers des pipelines hybrides, combinant automatisation et contrôle manuel, apparaît comme une voie prometteuse pour dépasser les limites rencontrées. En définitive, ce stage s'est révélé formateur et enrichissant, favorisant le développement de compétences techniques pointues tout en contribuant à un projet innovant à forte valeur ajoutée pour l'ingénierie urbaine.

8. Bibliographie - webographie

- Gröger, G., Kolbe, T.H., Nagel, C., & Häfele, K.-H. (2012). **OGC City Geography Markup Language (CityGML) Encoding Standard Version 2.0.0.** Open Geospatial Consortium (OGC). Document officiel OGC 12-019. Régulièrement consulté pendant toute la période du stage. [Disponible en ligne](#)
- Kutzner, T., Chaturvedi, K., & Kolbe, T.H. (2020). **OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard Version 3.0.** Open Geospatial Consortium (OGC). Document officiel OGC 20-010. Régulièrement consulté pendant toute la période du stage. [Disponible en ligne](#)
- Open Geospatial Consortium (OGC). **CityGML Standards and Specifications.** Portail officiel OGC. Régulièrement consulté pendant toute la période du stage. <https://www.ogc.org/standards/citygml>
- 3D City Database (3DCityDB). **Documentation officielle et manuels d'utilisation.** Régulièrement consulté pendant toute la période du stage. <https://www.3dcitydb.org>
- Kolbe, T.H., Nagel, C., & Stadler, A. (2009). **CityGML – Interoperable Access to 3D City Models.** In: *Proceedings of the 3rd International Conference on GeoSpatial Semantics (GeoS 2009)*, Mexico City. Springer, Lecture Notes in Computer Science, vol 5. Régulièrement consulté pendant toute la période du stage.

9. Annexes

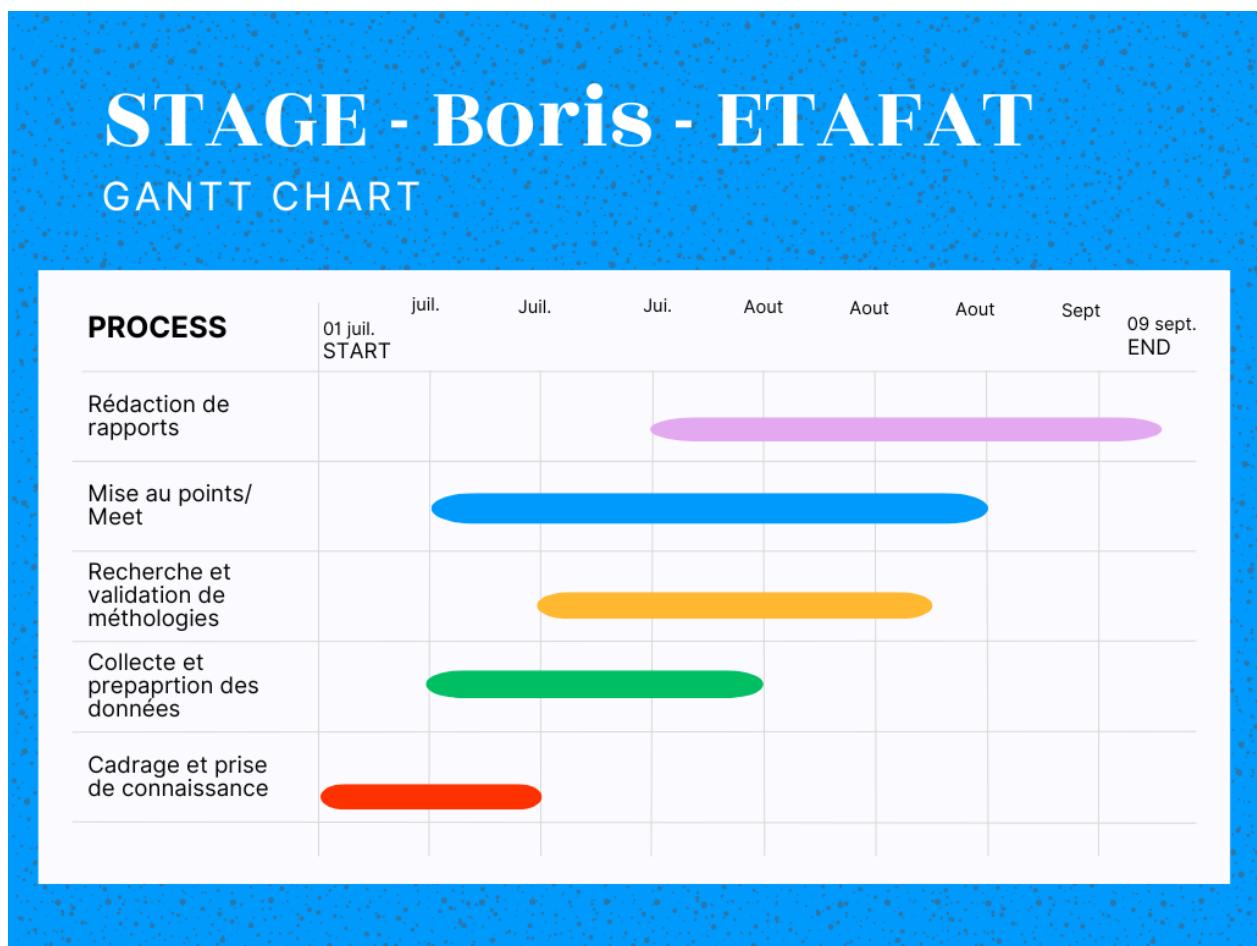


Figure 32: Organisation temporelle du stage/ diagramme de gantt

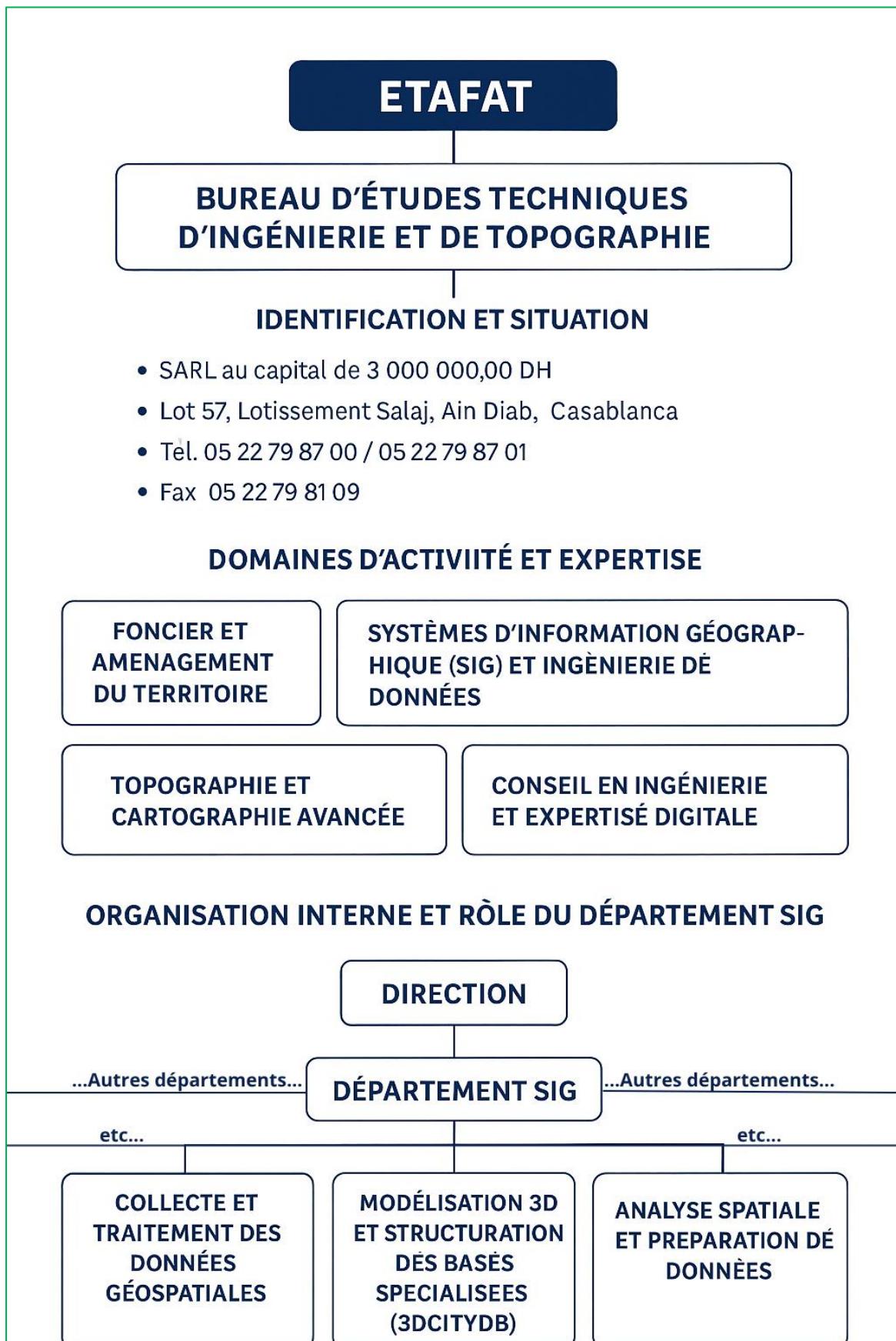


Figure 33: Résumé de la présentation de l'entreprise

Manuel d'installation des logiciels open source nécessaires à la chaîne de production CityGML

1. Installation de QGIS

1. Se rendre sur le site officiel : <https://qgis.org>.
2. Aller dans l'onglet **Téléchargement**.
3. Choisir la version **Long Term Release (LTR)** (plus stable) ou la dernière version.

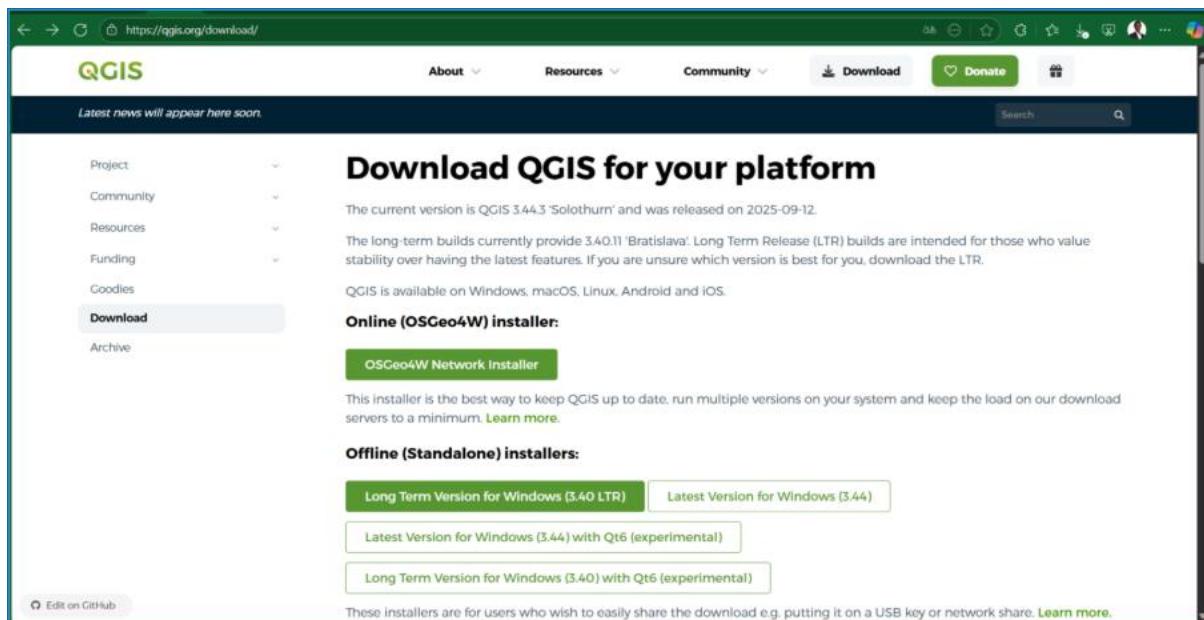


Figure 34: page de téléchargement de QGIS

4. Télécharger le fichier correspondant à votre système (Windows, MacOS ou Linux).
5. Lancer le module d'installation (.exe sous Windows).
6. Suivre l'assistant et installer avec les options par défaut.
7. Vérifier après installation que QGIS se lance correctement.

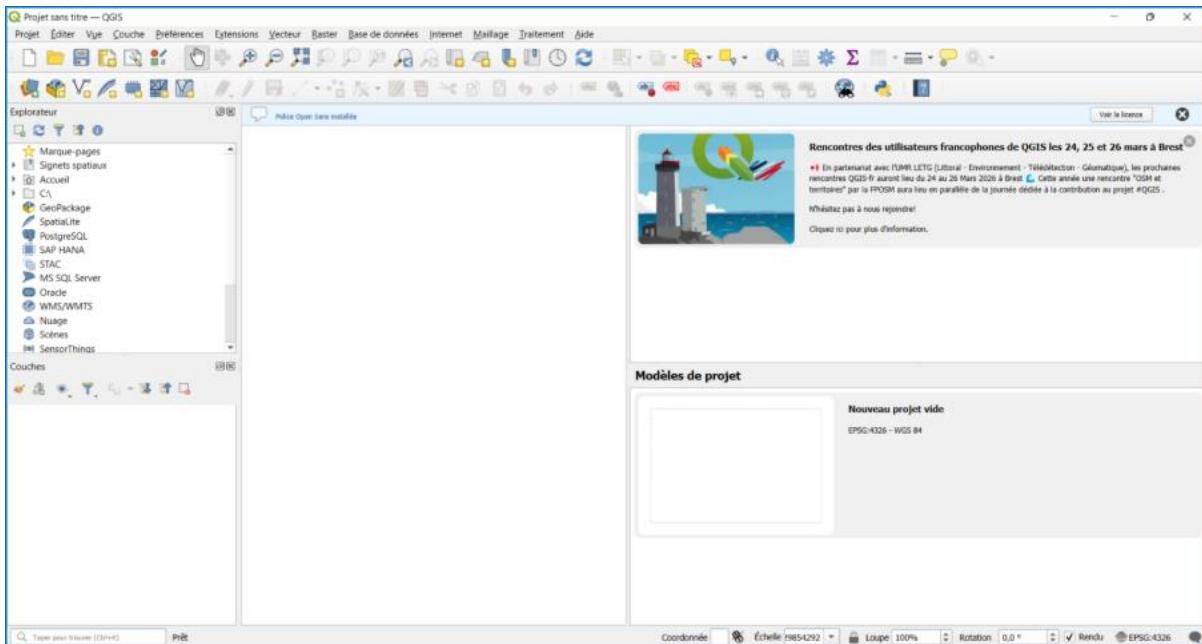


Figure 35: Interface de QGIS

2. Installation de CloudCompare

1. Aller sur le site officiel [CloudCompare 2.6 Download \(Free\) - CloudCompare-Chinese.exe](https://cloudcompare.software.informer.com/download/#downloading).
2. Cliquer sur **Download**.
3. Sélectionner la version correspondant à votre système (Windows/Mac/Linux).

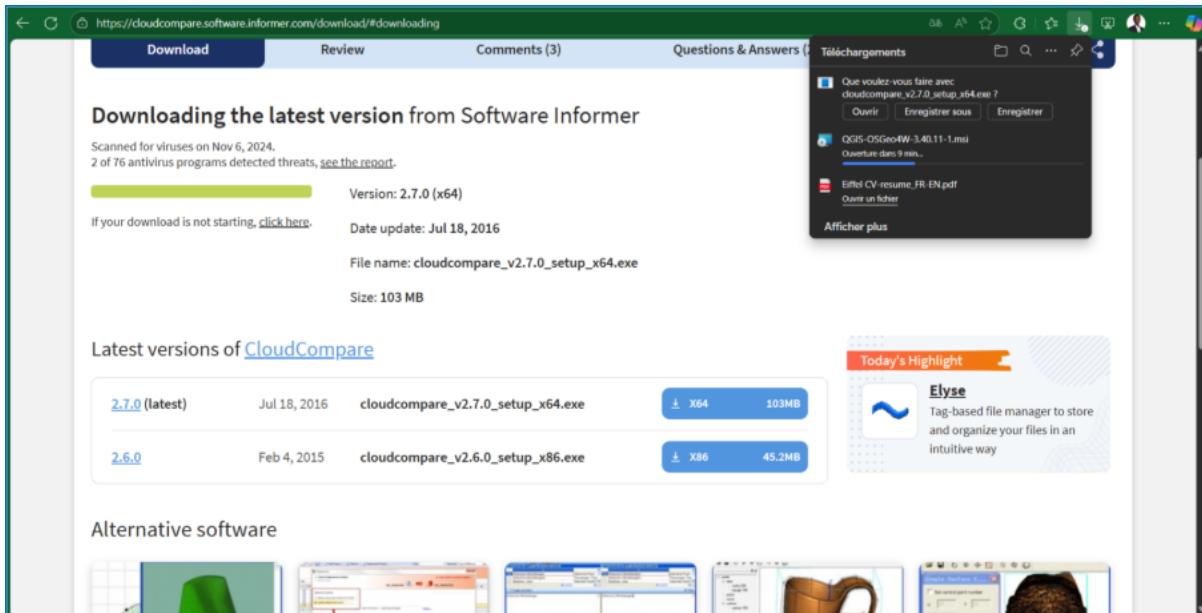


Figure 36: page de téléchargement de cloud compare

4. Télécharger et exécuter le fichier d'installation.
5. Suivre l'assistant (installation classique).

6. Lancer le logiciel pour confirmer son bon fonctionnement.
(Éventuellement ouvrir un nuage de point comme dans ce cas)

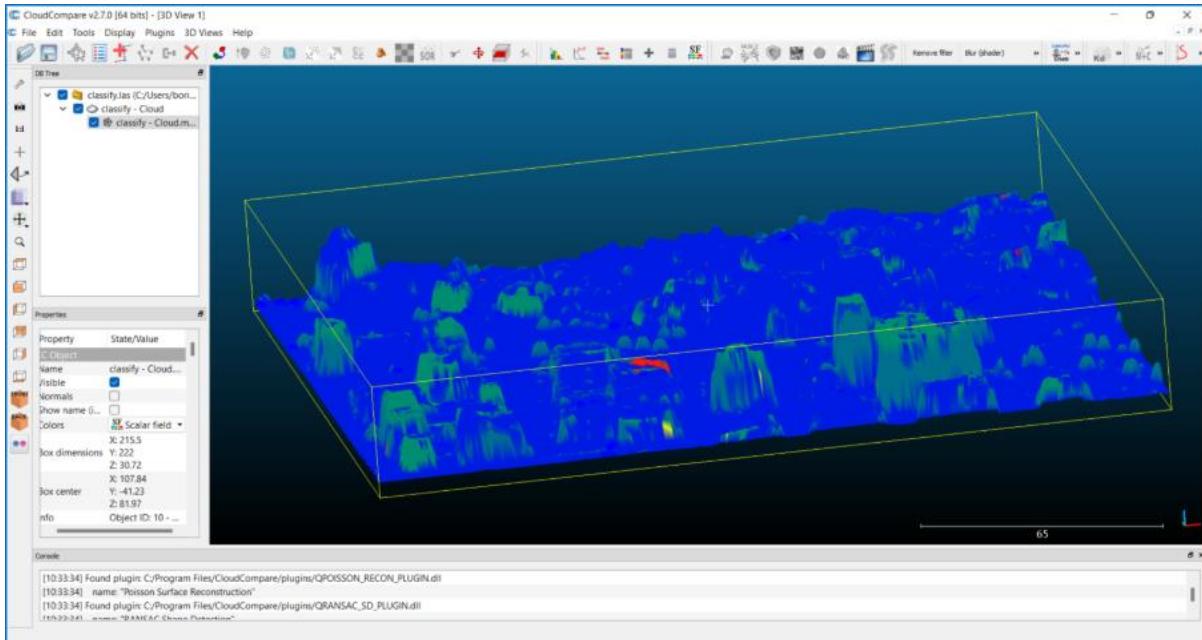


Figure 37: Interface de cloud compare

3. Installation de Visual Studio Code (VS Code)

1. Se rendre sur le site officiel : <https://code.visualstudio.com>.

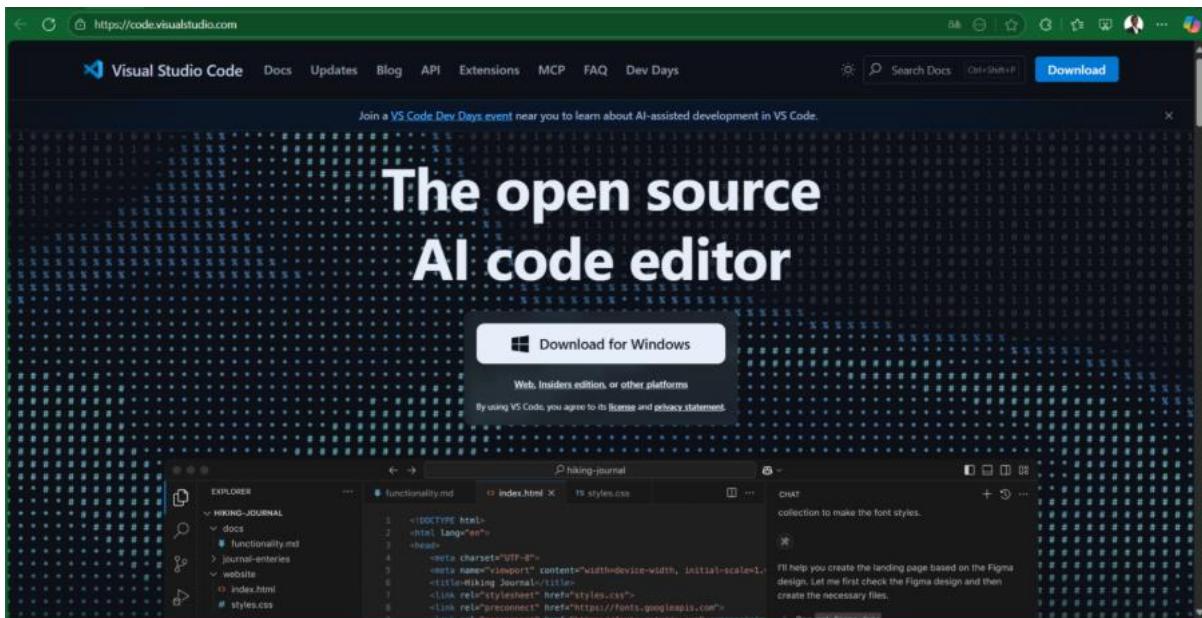


Figure 38: téléchargement de Visual Studio Code

2. Cliquer sur **Download**.
3. Choisir la version Windows (ou autre OS selon votre machine).
4. Télécharger le fichier **VSCodeSetup.exe**.

5. Lancer l'installation et cocher l'option **Ajouter à PATH** (utile pour la ligne de commande).
6. Une fois installé, ouvrir VS Code.

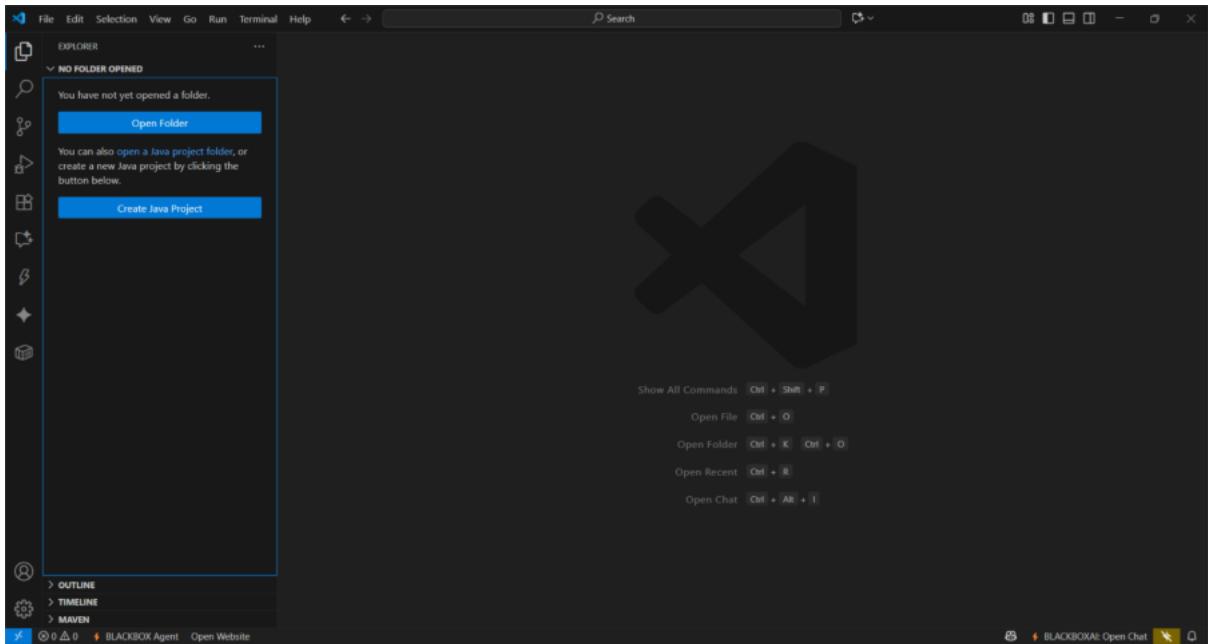


Figure 39: Interface de Vs code

7. (Optionnel) Installer les extensions utiles :
 - a. *Python*
 - b. *Docker*
 - c. *SQL Tools*

4. Installation de PostgreSQL + PostGIS + 3DCityDB (via Docker)

Pré-requis :

- Installer **Docker Desktop** : <https://www.docker.com/products/docker-desktop>.

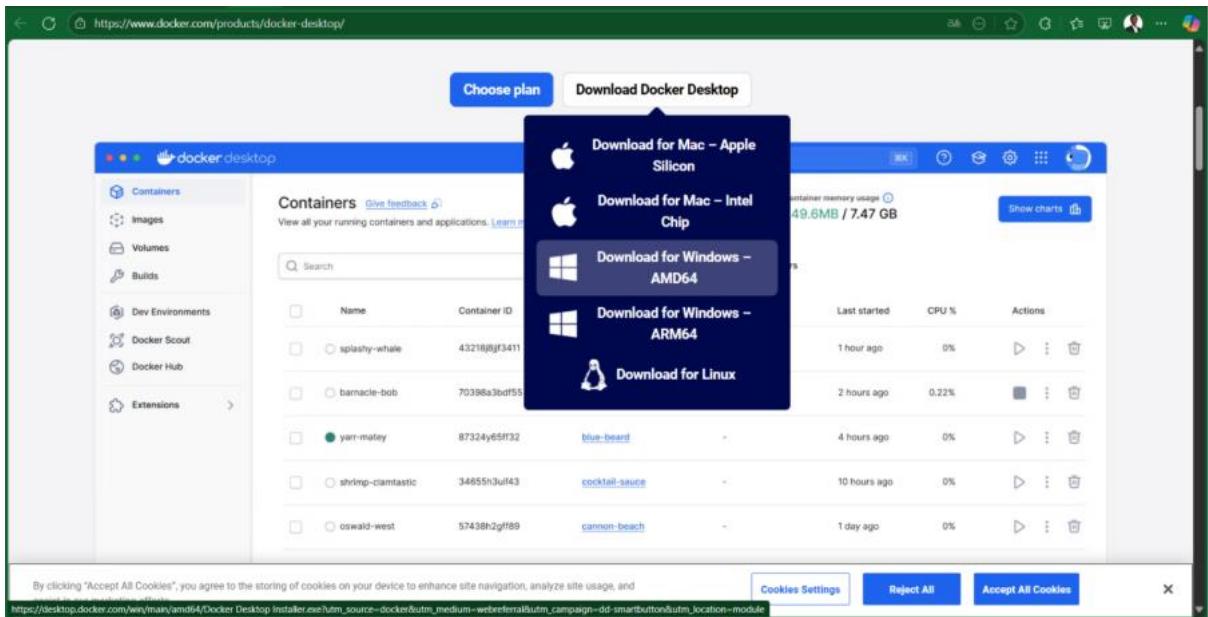


Figure 40: Téléchargement de docker

- Ouvrir docker et s'inscrire pour pouvoir l'utiliser

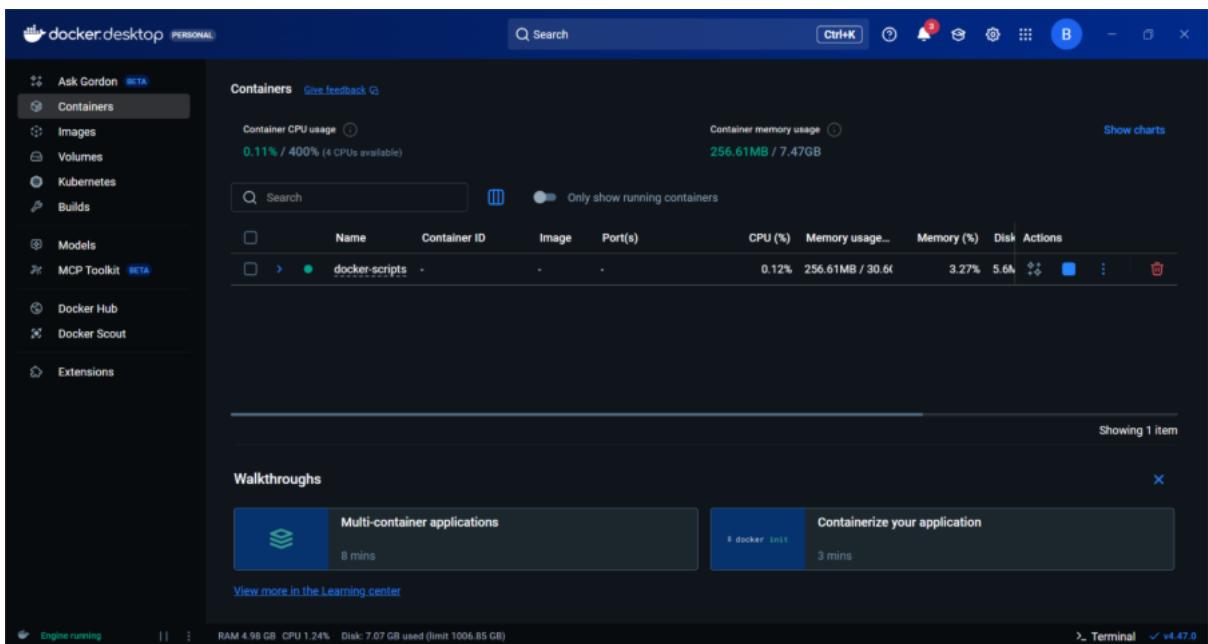


Figure 41: Interface de docker

- Vérifier que Docker est bien installé avec la commande (sur la ligne de commande PowerShell ou autres) : la version de docker sera affichée si tout est correcte. Dans le cas échéant, on observe une erreur ou la commande ne sera pas reconnue : **docker –version** (commande)

Étapes d'installation :

1. Récupérer les scripts Docker de 3DCityDB sur GitHub :

a. <https://github.com/3dcitydb/3dcitydb>.

2. Cloner le dépôt ou télécharger le ZIP et le dézipper dans un dossier(emplacement) bien connu.

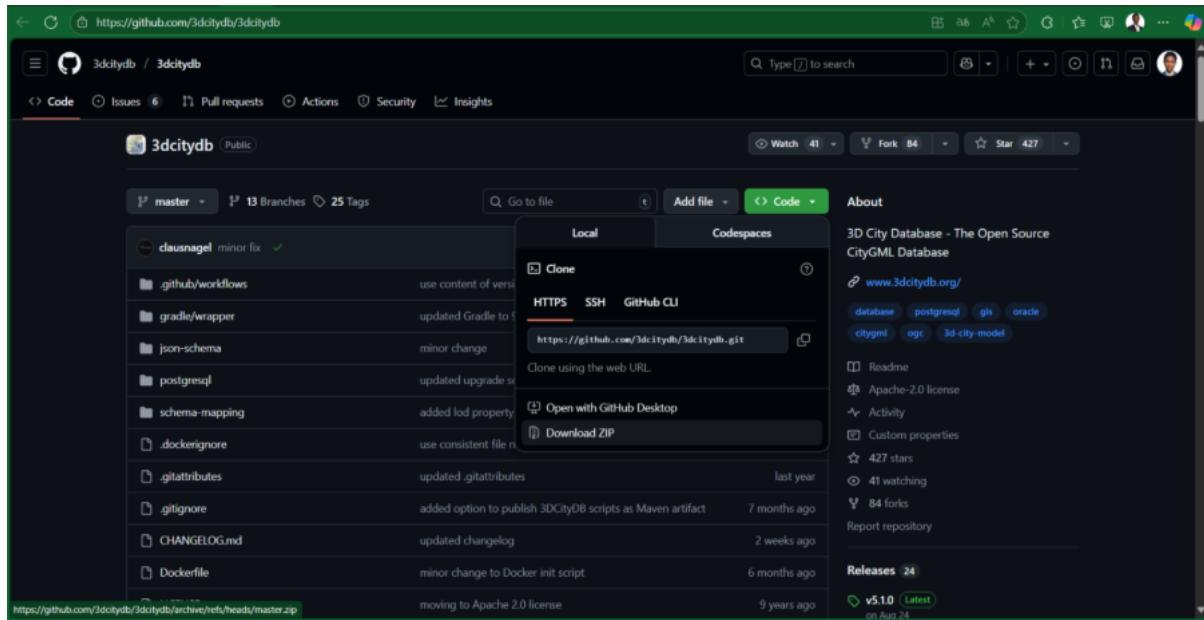


Figure 42: Téléchargement de 3DcityDb sur Github

3. Ouvrir un terminal dans le dossier postgresql/docker-scripts du dossier dézipper.

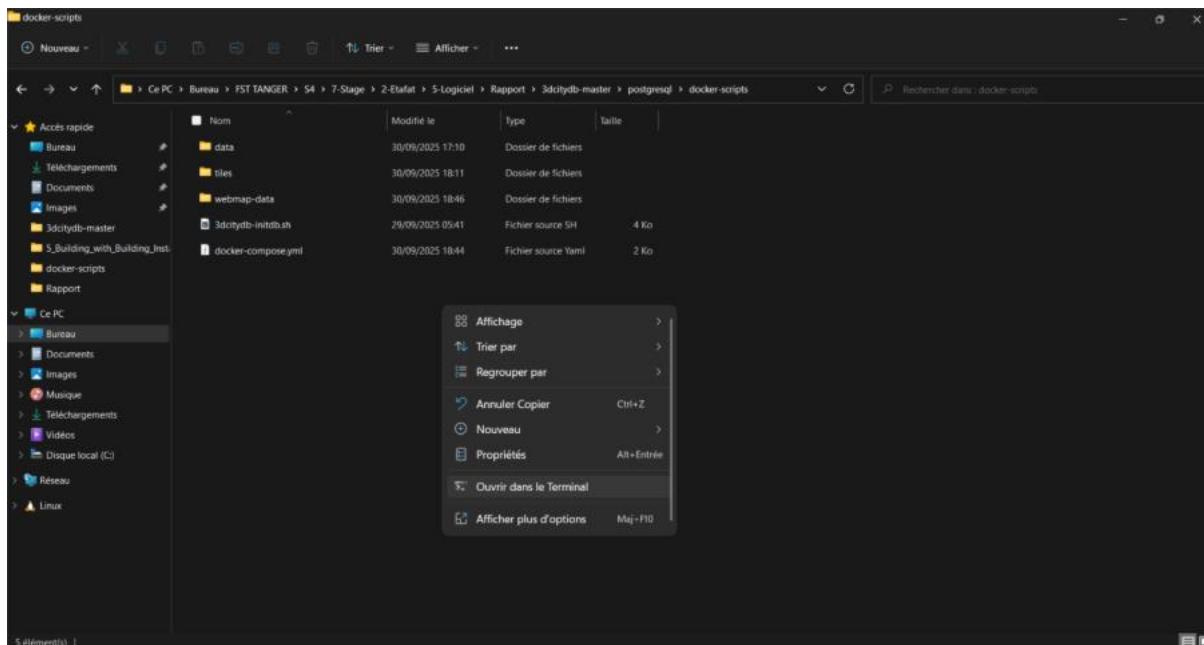


Figure 43: ouverture du dossier dans un terminal

4. Lancer la base PostgreSQL/PostGIS et 3DCityDB (un fichier docker-compose.yml doit être préalablement crééé) avec :
docker compose up -d

Contenu du fichier docker compose en annexe

```

version: "3.9"
services:
  db:
    image: 3dcitydb/pg:latest
    container_name: citydb
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: geoinfo
      POSTGRES_DB: citydb
    ports:
      - "5432:5432"
    volumes:
      - citydb_data:/var/lib/postgresql/data

  pgadmin:
    image: dpage/pgadmin4
    container_name: pgadmin
    environment:
      PGADMIN_DEFAULT_EMAIL: borissanne@gmail.com
      PGADMIN_DEFAULT_PASSWORD: geoinfo
    ports:
      - "5050:80"
    depends_on:
      - db

  citydb-tools:
    image: 3dcitydb/citydb-tool:latest
    container_name: citydb-tool
    depends_on:
      - db
    environment:

volumes:
  citydb_data:

```

Figure 44: extrait du contenu du fichier docker-compse.yml

5. Vérifier que les conteneurs sont bien lancés sur docker desktop ou via la ligne de commande (docker ps) :

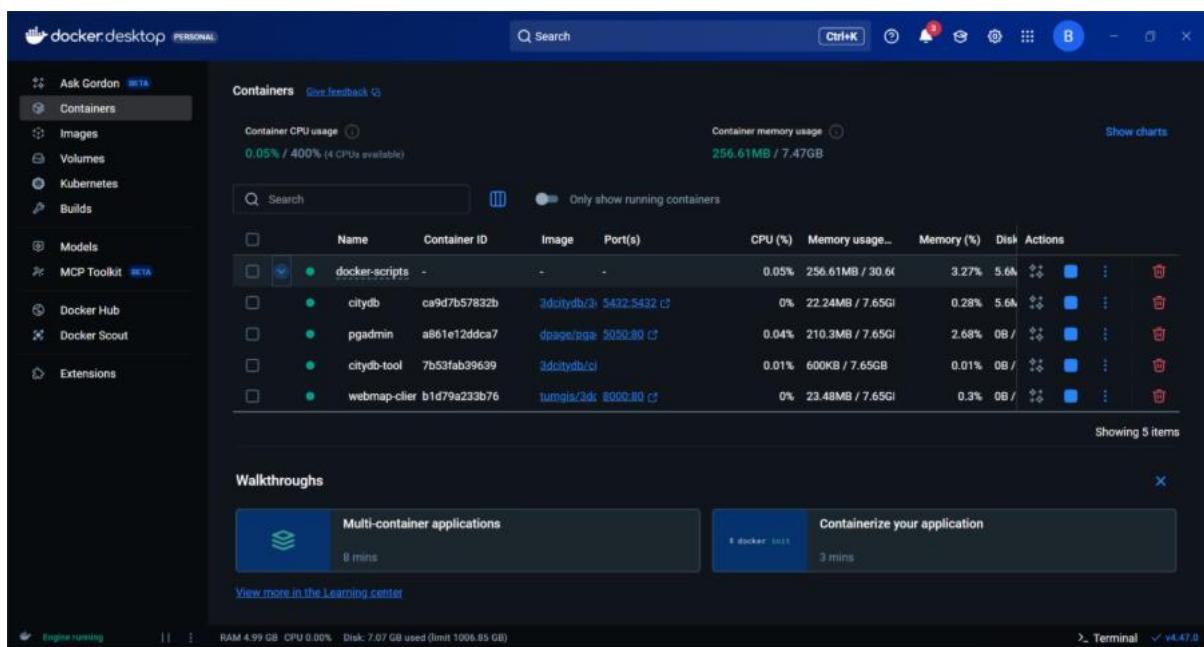


Figure 45: Interface de Docker avec les services lancés.

6. Connexion à la base via pgAdmin pour la visualisation des données (cliquer sur le champ port correspondant au service pgAdmin pour ouvrir la page de connexion à pgAdmin) :

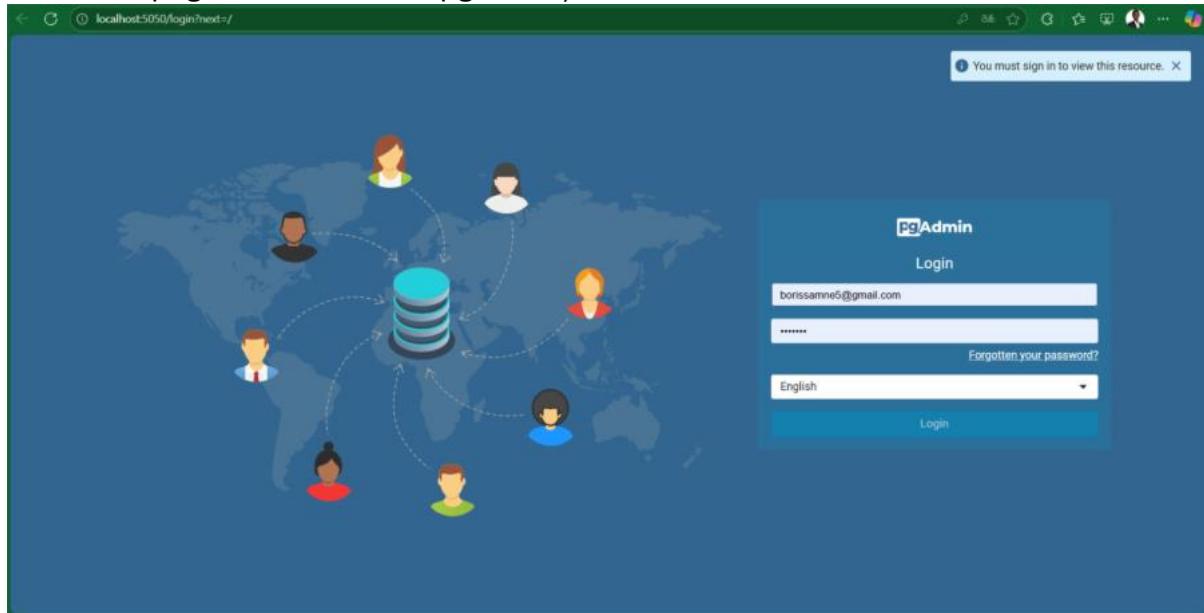


Figure 46: Interface de connexion à PG admin

7. Une fois connecté sur pgAdmin, se connecter à la base de données 3DcityDB : dans l'explorateur d'objets (en haut à gauche), faire un **clic droit** sur server :

Server-> register -> server et maintenant entrer les informations de connexions

- Host** : localhost (local) ou le nom du service (docker)
- Port** : 5432 (peut changer en fonction de la config. du fichier compose)
- User** : postgres
- Password** : celui défini dans docker-compose.yml.

Les autres infos de connexions peuvent également différer en fonction du fichier de configuration docker-compose.yml (ne pas oublier de donner un nom quelconque au server dans l'onglet général).

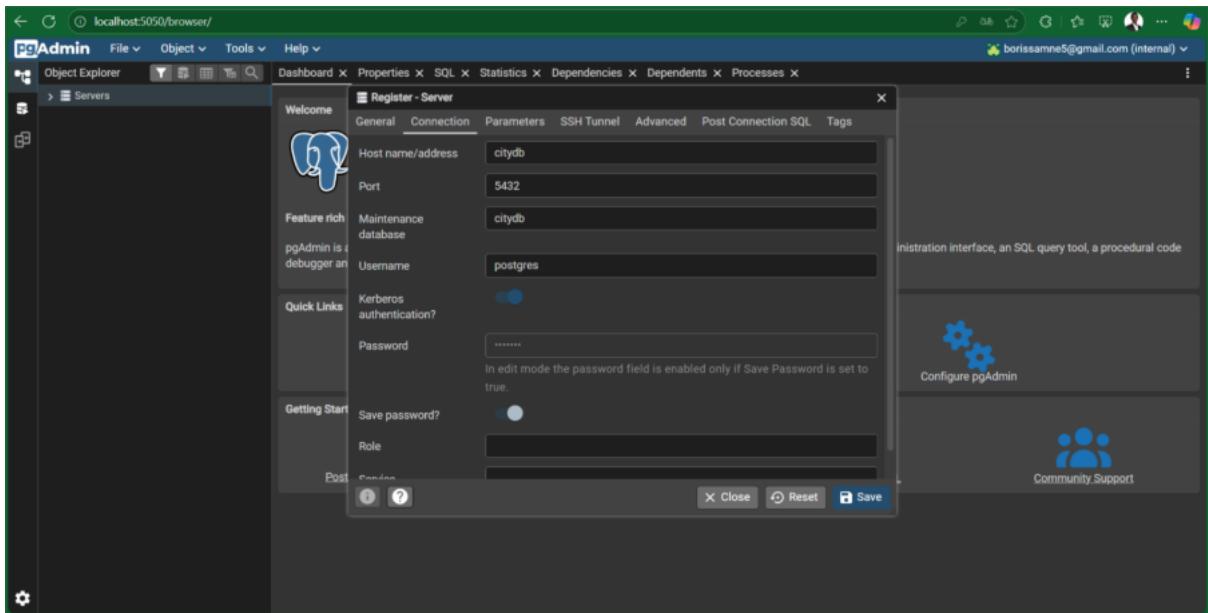


Figure 47: Connexion à la base de données 3D city DB

8. La base contient maintenant le schéma citydb prêt à l'emploi.

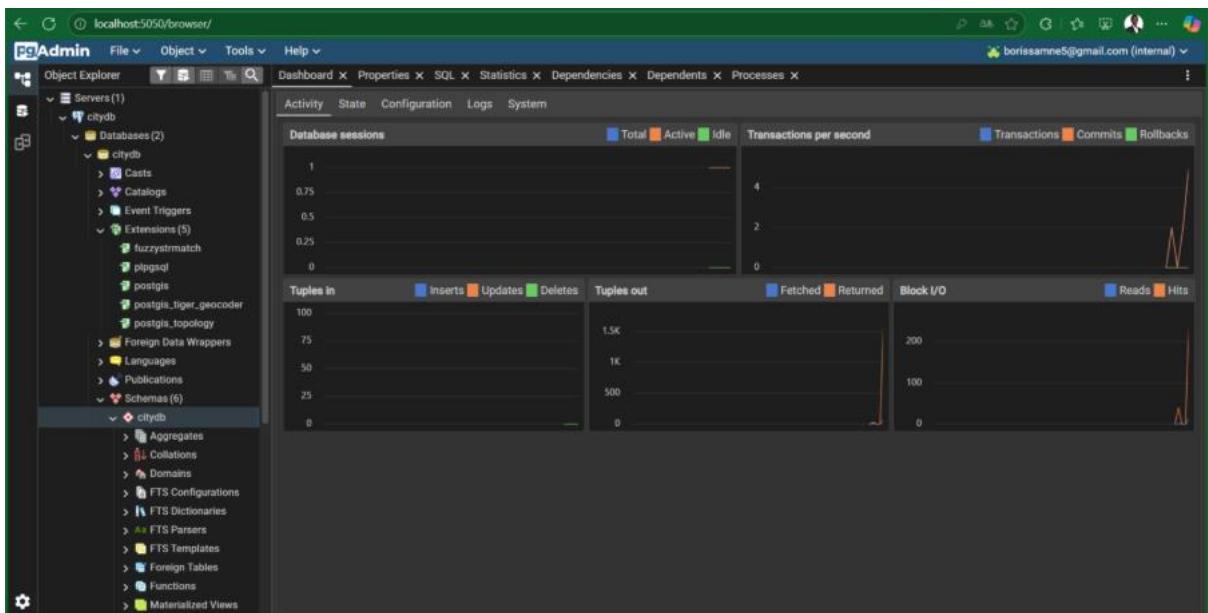


Figure 48: Consultation de la BD qui contient bien les schéma adéquats.

5. Installation de KIT Model Viewer (KITModelViewer.exe)

1. Se rendre sur le site officiel de KIT :
<https://www.iai.kit.edu/english/1302.php>

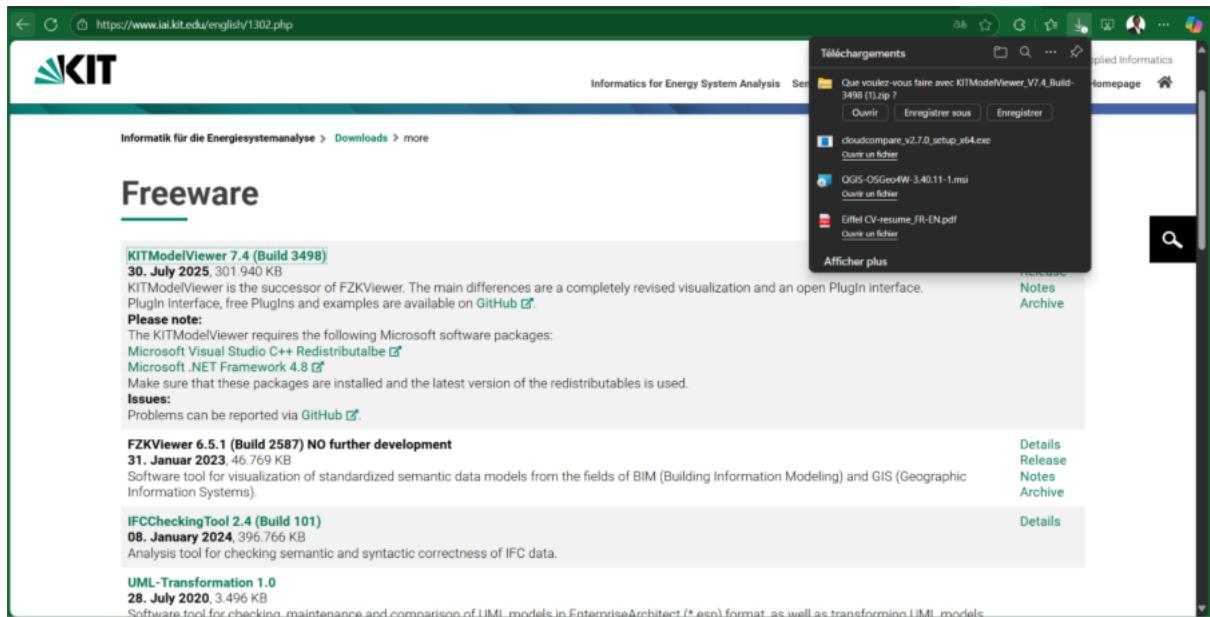


Figure 49: Téléchargement de Kit Model Viewer

2. Télécharger le fichier exécutable (juste télécharger le zip et double-cliquer sur **KITModelViewer.exe** pour ouvrir le logiciel, aucune installation complexe requise)
3. Vérifier l'ouverture d'un fichier CityGML ou CityJSON pour tester l'installation (glisser – déposer ou via file->open).

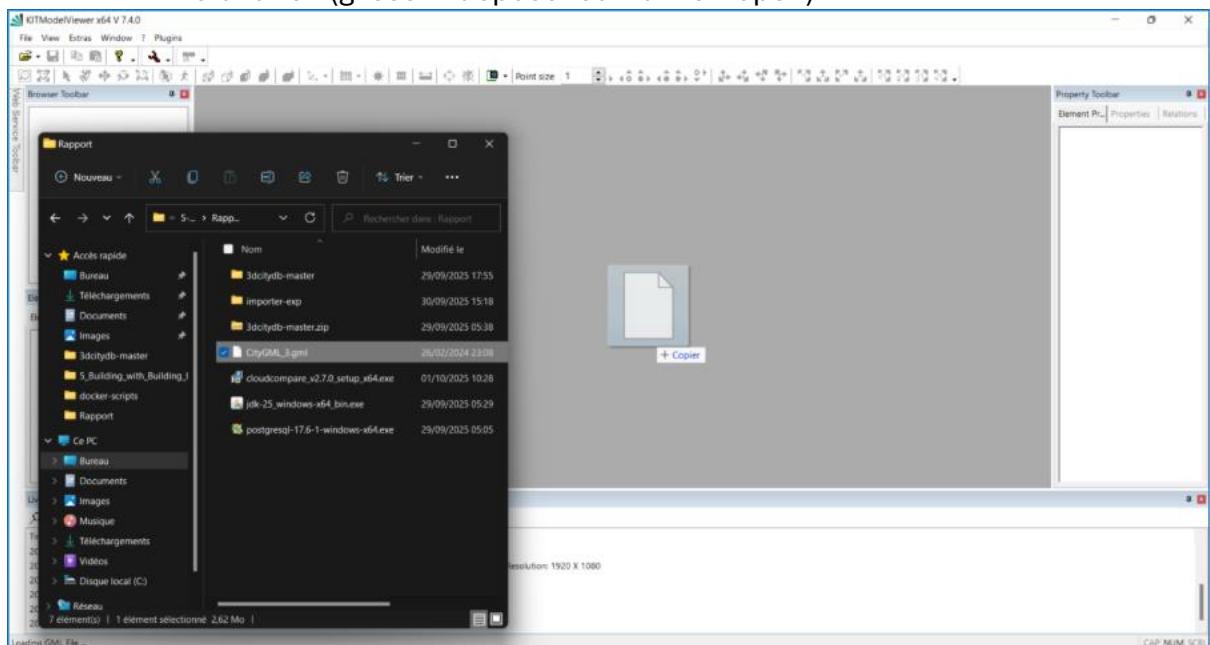


Figure 50: Ouverture d'un fichier CityGML sur KIT model viewer

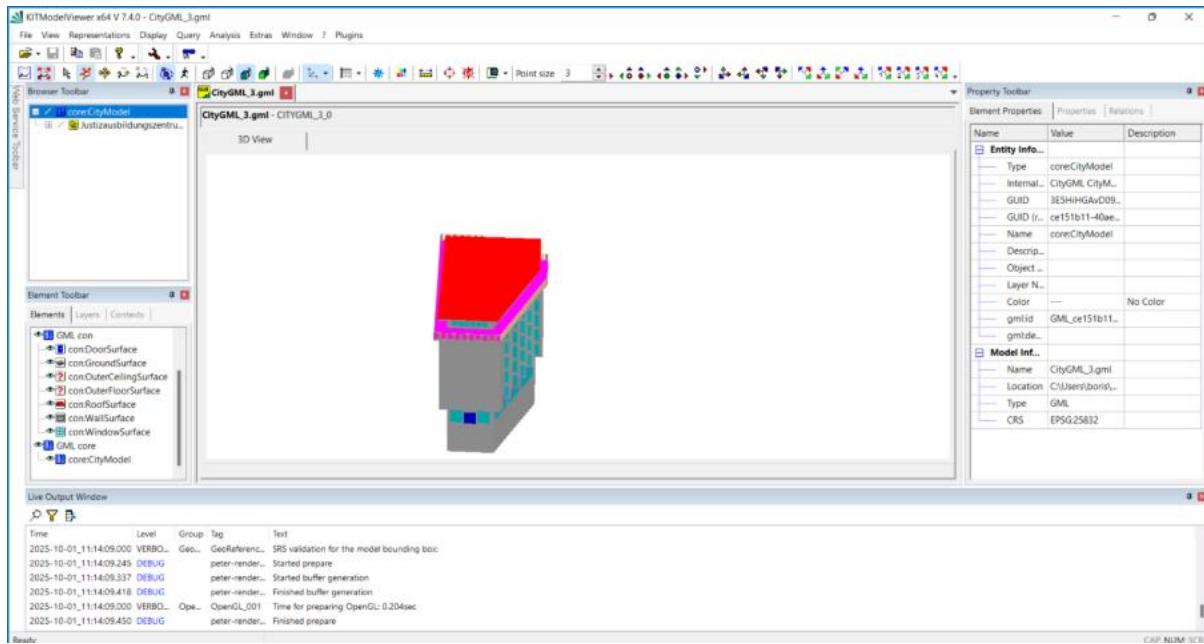


Figure 51: Fichier Ouvert sur Kit Model Viewer

5. Annexe du Manuel d'installation des logiciels

Le fichier docker-compose.yml est un fichier de configuration qui facilite le lancement d'un ensemble de service (liés ou pas) sur docker. Dans ce cas, les services lancés sont citydb, pgAdmin, citydb-tool. Voilà le script correspondant (les indentations (espaces) doivent être respectées comme en python pour que le script marche correctement) :

```
version: "3.9"

services:
  db:
    image: 3dcitydb/3dcitydb-pg:latest
    container_name: citydb
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: geoinfo
      POSTGRES_DB: citydb
      SRID: 4326
    ports:
      - "5432:5432"
    volumes:
      - citydb_data:/var/lib/postgresql/data

  pgadmin:
    image: dpage/pgadmin4
    container_name: pgadmin
    environment:
      PGADMIN_DEFAULT_EMAIL: borissamne5@gmail.com
      PGADMIN_DEFAULT_PASSWORD: geoinfo
    ports:
      - "5050:80"
```

```
depends_on:
- db

citydb-tool:
image: 3dcitydb/citydb-tool:latest
container_name: citydb-tool
depends_on:
- db
environment:
CITYDB_HOST: db
CITYDB_PORT: 5432
CITYDB_NAME: citydb
CITYDB_SCHEMA: citydb
CITYDB_USERNAME: postgres
CITYDB_PASSWORD: geoinfo
volumes:
- ./data:/data
entrypoint: ["tail", "-f", "/dev/null"]

volumes:
citydb_data:
```