كلية العلوم و التقنيات بطنجة
Faculté des Sciences et Techniques de Tanger

جامعة عبد المالك السعدي
Université Abdelmalek Essaadi

Cycle d'ingénieurs Géo-Information
2e année 2024-2025
Programmation SIG et Télédétection

QGIS

EXTENSION DEVELOPPEMENT

Download Satellites Images and Check "Dérogation"

Encadré par :
- Mr. Otmane YAZIDI ALAOUI

Réalisé par :
- BORIS Pengdwende SAMNE
- Lamiae EL HAMRAOUI
- Imad NIFDAL
- Hanae NEBKHOUT

Remerciements

La réalisation de ce projet a été rendue possible grâce au soutien, à l'encadrement et à la bienveillance de nombreuses personnes que nous souhaitons chaleureusement remercier.

Nous exprimons notre profonde gratitude à Monsieur Otmane YAZIDI ALAOUI, enseignant à la Faculté des Sciences et Techniques de Tanger, pour son encadrement rigoureux, ses conseils avisés et sa disponibilité constante tout au long de notre travail. Son implication et sa confiance ont été essentielles dans l'avancement et la réussite de ce projet.

Nous tenons également à remercier l'ensemble de nos enseignants pour leur engagement, la qualité de leur enseignement et les compétences qu'ils nous ont transmises tout au long de notre parcours universitaire.

Enfin, nous adressons nos remerciements les plus sincères à toutes les personnes qui, de près ou de loin, ont contribué à la réussite de ce projet. Même si leurs noms ne figurent pas ici, nous leur témoignons toute notre reconnaissance pour leur aide précieuse, leur soutien ou leurs encouragements.

Résumé

Ce projet présente le développement d'un plugin QGIS multifonctionnel visant à améliorer la gestion de données spatiales pour des applications environnementales et réglementaires.

Le plugin intègre deux fonctionnalités principales : le téléchargement automatisé d'images satellites (Sentinel, Landsat) filtrées selon des critères spatio-temporels, et la vérification des dérogations réglementaires par analyse spatiale des couches vectorielles. Conçu en Python avec les bibliothèques PyQt5, rasterio, geopandas et reportlab, l'outil permet également le calcul d'indices spectraux (NDVI, NDWI, etc.) et la génération de rapports PDF.

Ce projet démontre l'intérêt d'un outil SIG intégré pour faciliter la prise de décision, l'optimisation des ressources et le respect des réglementations environnementales.

Table des matières

Liste des figures

Introduction

L'évolution rapide des technologies géospatiales a profondément transformé les pratiques de gestion territoriale, de surveillance environnementale et de prise de décision dans de nombreux domaines. L'accès aux images satellites et l'utilisation de systèmes d'information géographique (SIG) permettent aujourd'hui d'analyser, de modéliser et de contrôler les dynamiques spatiales avec une précision et une efficacité accrue. Cependant, l'exploitation de ces outils reste souvent complexe pour les utilisateurs non experts, notamment en matière de traitement des images et d'analyse réglementaire.

C'est dans ce contexte que s'inscrit le présent projet, qui vise à développer un plugin intégré dans QGIS, une plateforme SIG libre et open-source largement adoptée dans le monde. Le plugin, pensé comme un outil pratique et accessible, permet à la fois le téléchargement ciblé d'images satellites selon des critères définis, et la vérification automatisée des dérogations d'exploitation à partir d'analyses spatiales avancées. En combinant ces deux fonctionnalités au sein d'une interface intuitive, ce projet propose une solution innovante pour améliorer la gestion des données spatiales et renforcer le contrôle réglementaire.

Cette introduction pose ainsi les bases d'un travail orienté vers la production d'un outil opérationnel, pensé pour répondre à des besoins concrets dans les domaines de l'aménagement du territoire, de l'environnement et du suivi foncier.

I. Contexte et environnement du projet

I.1 Contexte et enjeux du projet

Le projet s'inscrit dans le contexte croissant de l'utilisation des systèmes d'information géographique (SIG) pour la gestion environnementale et foncière. L'intégration de données satellitaires et la gestion des dérogations réglementaires sont des enjeux majeurs pour assurer une exploitation durable et conforme des territoires. Face à la multiplication des sources de données et à la complexité des réglementations, il devient essentiel de proposer des outils intégrés, performants et accessibles pour les gestionnaires et décideurs.

I.2 Objectifs généraux et spécifiques

L'objectif principal est de développer un plugin QGIS multifonctionnel permettant à la fois le téléchargement optimisé d'images satellites et la vérification des dérogations sur des surfaces d'exploitation. Ce plugin vise à faciliter les processus décisionnels en fournissant des outils intégrés et interactifs, adaptés aux besoins des utilisateurs SIG dans le domaine de l'environnement et de l'aménagement du territoire.

I.3 Environnement technique et choix de QGIS comme plateforme

QGIS a été choisi pour sa flexibilité, son ouverture, et sa large communauté d'utilisateurs et de développeurs. Le plugin est développé en Python, utilisant les API QGIS pour la gestion spatiale et PyQt5 pour l'interface utilisateur. L'intégration de bibliothèques tierces (sentinelsat, rasterio, geopandas, matplotlib, numpy, pandas, reportlab) permet d'assurer la robustesse et la richesse fonctionnelle de l'outil.

- Système SIG : QGIS (version 3.x)

QGIS est un logiciel SIG libre et gratuit, largement utilisé dans les domaines de la géomatique, de l'environnement, de l'urbanisme et de la planification territoriale. Il offre une grande flexibilité grâce à son architecture basée sur des plugins Python, ce qui permet d'ajouter facilement des fonctionnalités personnalisées. La version 3.x de QGIS est stable, riche en outils de visualisation, d'analyse spatiale, de géotraitement et d'export de données.

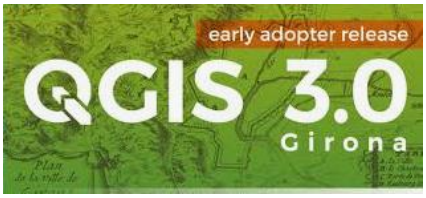


Figure 1 : Logo de QGIS

- Langage de développement : Python

L'utilisation de Python s'explique par :

- Sa compatibilité native avec l'API de QGIS.

- La richesse de son écosystème pour la manipulation de données spatiales.

- La possibilité de créer des interfaces graphiques et d'automatiser des traitements complexes.

Figure 2 : Logo de python

- Outils de développement utilisés

Plugin Builder : Extension officielle de QGIS facilitant la création de la structure de base d'un plugin (arborescence, fichiers Python, fichiers de configuration .ui, etc.). Il accélère la mise en place du projet.

Plugin Reloader : Extension officielle de QGIS indispensable pour un développement itératif efficace. Il permet de recharger le plugin dans QGIS sans redémarrer l'application, économisant un temps précieux lors des tests et débogages. Après chaque modification du code (par exemple dans main_plugin.py), un simple rafraîchissement via le Plugin Reloader applique les changements immédiatement. Cet outil est particulièrement utile pour ajuster les interactions utilisateur ou corriger des bugs en temps réel.

Qt Creator : Environnement de développement intégré (IDE) utilisé pour la conception visuelle de l'interface utilisateur (UI) à l'aide de fichiers .ui compatibles avec PyQt5. Il permet d'éditer graphiquement les fenêtres, boutons, menus, etc.

Figure 3 : Logo de QT Creator

Vs code :

VS Code, équipé d'extensions Python et PyQt, a été l'IDE principal pour coder le plugin. Ses fonctionnalités clés incluent :

- Auto-complétion intelligente pour PyQt5 et QGIS API.
- Débogage pas à pas pour tester les requêtes API (ex: SentinelHub) ou les calculs spatiaux.
- Gestion de Git intégrée pour le versionnement.

Figure 4 : Logo de VS code

- Bibliothèques et modules Python utilisés

Catégorie	Bibliothèques / Modules
Modules QGIS (Core & GUI)	qgis.core, qgis.gui, qgis.PyQt.QtCore, qgis.PyQt.QtGui, qgis.PyQt.QtWidgets
Interface graphique (PyQt5)	PyQt5.QtWidgets (QTableWidgetItem, QMessageBox, QFileDialog)
Traitement de données	numpy, pandas, seaborn
Visualisation de données	matplotlib.pyplot
Géodonnées / SIG	geopandas, rasterio, rasterstats.zonal_stats, shapely.geometry.box
Téléchargement / Web	requests, zipfile, io.BytesIO, xml.etree.ElementTree, os.path
Imagerie satellite	sentinelsat.SentinelAPI, sentinelhub (SentinelHubCatalog, SentinelHubRequest, BBox, CRS, DataCollection, etc.)
Modules internes au plugin	.resources, .Download_Products_dialog

Figure 5 : Tableau des modules externes à installer.

- Sources de données

Couches géographiques locales ou fournies :

Domaine communal, forestier, public, privé

Localisation des projets existants

Espaces verts, zones naturelles

Images satellites :

Sentinel-2 : Imagerie multispectrale à haute résolution

Landsat : Données historiques utiles pour le suivi temporel

I.4 Présentation synthétique du plugin et de ses deux fonctionnalités majeures

Le plugin intègre deux fonctionnalités principales :

- **Téléchargement d'images satellites :** filtrage spatial et temporel, gestion de la couverture nuageuse, visualisation des résultats et optimisation des données téléchargées.
- **Vérification des dérogations :** contrôle de la conformité des surfaces exploitées via des analyses spatiales avancées, visualisation interactive, et génération de rapports PDF détaillés.

II. Analyse des besoins et cahier des charges

II.1 Fonctionnalité 1 : Téléchargement d'images satellites

Cette fonctionnalité permet à l'utilisateur de rechercher, filtrer et télécharger des images satellites adaptées à ses besoins. Les critères incluent :

- Date d'acquisition (début/fin)
- Pourcentage maximal de couverture nuageuse
- Zone géographique (sélection manuelle sur la carte, rectangle ou polygone)
- Résolution spatiale

L'interface doit être intuitive et permettre une visualisation préalable des images (miniatures, métadonnées principales). Le téléchargement se fait au format GeoTIFF, avec une organisation locale structurée. Les métadonnées sont extraites pour faciliter la gestion et l'analyse ultérieure.

II.2 Fonctionnalité 2 : Vérification des dérogations

Cette fonctionnalité vise à vérifier la conformité des surfaces exploitées en analysant les couches vectorielles de dérogation et d'exclusion.

- L'utilisateur sélectionne la couche de dérogation et définit un point central (par clic sur la carte ou saisie de coordonnées).
- Il choisit dynamiquement les couches d'exclusion à prendre en compte.
- Le plugin crée un buffer autour du point central et recherche toutes les entités intersectant cette zone dans les couches sélectionnées.
- Les entités détectées sont listées dans une interface interactive : chaque entité peut être surlignée sur la carte.
- Un bouton dédié avec icône flèche vers le bas permet de générer manuellement un rapport PDF.
- Le rapport PDF contient :
    - Un tableau des entités détectées (avec retour à la ligne automatique pour les noms longs)
    - Statistiques : nombre total, distances moyenne/min/max, surface totale et moyenne
    - Coordonnées du point central et rayon du buffer

- **Aucune option d'enregistrement ou de modification directe dans la couche n'est proposée** : la fonctionnalité est strictement consultative et de reporting.

II.3 Contraintes techniques et fonctionnelles communes

Le plugin doit :

- Être performant et compatible avec QGIS 3.x
- Gérer correctement les projections spatiales
- Proposer une interface réactive et fluide, avec navigation aisée entre les fonctionnalités
- Assurer la validation des données et la gestion des erreurs (messages clairs, robustesse)
- S'appuyer sur des bibliothèques éprouvées pour le traitement spatial, l'analyse et la génération de rapports

II.4 Spécifications utilisateurs et scénarios d'utilisation

Les utilisateurs doivent pouvoir alterner facilement entre les deux fonctionnalités. Exemple de scénario type :

1. Préparation d'une zone d'étude par téléchargement d'images satellites adaptées
2. Vérification réglementaire via la fonctionnalité de dérogation sur cette même zone
3. Génération d'un rapport PDF pour archivage ou transmission

III. Architecture générale du plugin

III.1 Organisation modulaire intégrant les deux fonctionnalités

Le plugin est structuré en modules distincts mais intégrés :

- Un module pour le téléchargement et le traitement des images satellites
- Un module pour la vérification des dérogations et la génération de rapports

Cette organisation permet une maintenance facilitée et une évolution indépendante des fonctionnalités.

III.2 Description des composants principaux

- **Interface utilisateur** :
    - Onglets ou sections dédiées pour chaque fonctionnalité

- Widgets adaptés : combo box, listes, cases à cocher, boutons avec icônes, visualiseur de miniatures

- **Logique métier** :
  - Classes Python dédiées pour le traitement spatial, la gestion des couches, l'interaction avec les API satellites et la génération de rapports
- **Accès aux données** :
  - Gestion des couches vectorielles et raster via QGIS
  - Intégration des API externes pour le téléchargement d'images

III.3 Gestion des interactions utilisateur et navigation

L'utilisateur navigue facilement entre les fonctionnalités via l'interface. Les événements (clics, sélections) déclenchent des mises à jour dynamiques des couches et des données affichées. La synchronisation entre la carte, les listes et les formulaires est assurée pour une expérience fluide.

IV. Développement de la fonctionnalité Téléchargement d'images satellites

IV.1 Description fonctionnelle détaillée

L'utilisateur peut :

Partie 1 : Téléchargement des images satellites

Zone d'étude :

Saisie manuelle (rectangle, polygone) ou sélection directement sur la carte QGIS.

Choix des paramètres :

- ✓ Date de début et de fin
- ✓ Pourcentage maximal de couverture nuageuse

Lancement de la recherche :

- ✓ Interrogation de l'API SentinelHub ou autre (Sentinelsat, Landsatxplore)
- ✓ Affichage des images disponibles dans un tableau

Téléchargement :

- ✓ Téléchargement direct de l'image choisie par l'utilisateur
- ✓ Enregistrement local (format GeoTIFF)

Partie 2 : Visualisation des métadonnées

Affichage des informations principales de l'image sélectionnée :

- ✓ Nom du produit
- ✓ Date d'acquisition
- ✓ Taux de couverture nuageuse
- ✓ Résolution spatiale
- ✓ Système de coordonnées

Partie 3 : Traitement et analyse des indices

Choix du thème par l'utilisateur :

- ✓ Végétation : NDVI, NDCI
- ✓ Eau : NDWI, FDI
- ✓ Sol : BSI, SI

Calcul automatique des indices à partir des bandes téléchargées.

Visualisation des résultats sur QGIS sous forme de couches raster.

Fonctionnalités d'analyse avancée

- ✓ Matrice de corrélation : génération d'une heatmap pour explorer les relations entre indices calculés.
- ✓ Statistiques zonales : extraction des statistiques (min, max, moyenne...) d'un ou plusieurs indices dans une zone donnée (polygone).

IV.2 Architecture technique et intégration des API satellites

Le module interroge l'API SentinelHub pour filtrer les images selon les critères définis. Les appels sont optimisés pour limiter la taille des données et respecter les contraintes de couverture. Les images téléchargées sont stockées localement, avec extraction des métadonnées associées.

IV.3 Interface utilisateur dédiée

L'interface propose :

- Filtres dynamiques (date, nuages, résolution)
- Visualiseur de miniatures
- Boutons pour lancer les téléchargements
- Affichage des métadonnées principales

IV.4 Traitement et gestion des données téléchargées

Les images sont organisées localement. Les métadonnées sont extraites pour chaque image et associées dans un fichier dédié, facilitant la gestion et l'analyse ultérieure.

IV.5 Difficultés rencontrées et solutions apportées

- **Gestion des volumes de données** : filtrage spatial précis, limitation des résultats
- **Compatibilité des formats** : conversion systématique en GeoTIFF
- **Intégration avec QGIS** : gestion des projections et des systèmes de coordonnées
- **Réactivité de l'interface** : utilisation de threads pour ne pas bloquer l'UI

V. Développement de la fonctionnalité Vérification des dérogations

V.1 Description fonctionnelle détaillée

Zone d'étude : définition du point central et du rayon du buffer

- ✓ L'utilisateur sélectionne une couche de dérogation parmi les couches vectorielles disponibles dans le projet QGIS.
- ✓ Il définit un point central soit par un clic direct sur la carte QGIS, soit par saisie manuelle des coordonnées X et Y dans l'interface.
- ✓ Il choisit un rayon (en mètres) qui délimitera une zone tampon (buffer) autour du point central pour la recherche spatiale.

Choix des couches d'exclusion

- ✓ L'interface propose dynamiquement une liste des couches vectorielles disponibles, autres que la couche de dérogation, que l'utilisateur peut sélectionner via des cases à cocher.

- ✓ Ces couches représentent les zones d'exclusion réglementaires ou environnementales à prendre en compte dans la vérification.

Lancement de la vérification spatiale

- ✓ Le plugin crée un buffer géométrique autour du point central avec le rayon spécifié.
- ✓ Il interroge la couche de dérogation pour extraire toutes les entités dont la géométrie intersecte ce buffer.
- ✓ De même, il interroge toutes les couches d'exclusion sélectionnées pour détecter les entités intersectant la même zone tampon.

Affichage des résultats dans l'interface

- ✓ **Les entités détectées sont listées dans une zone d'affichage** sous forme textuelle.
- ✓ Chaque entité est présentée avec :
  - L'identifiant (ID) ou un attribut unique
  - Le nom de la couche
  - La distance entre le centre du buffer et le centroïde de l'entité
- ✓ La liste est **interactive** :
  - Cliquer sur une entité dans la liste la met en surbrillance sur la carte QGIS.
  - La vue cartographique se centre automatiquement sur l'entité sélectionnée pour faciliter l'analyse visuelle.

Génération de rapport PDF

- ✓ Un bouton distinct, identifiable par une icône flèche vers le bas, est disponible pour générer un rapport PDF.
- ✓ Ce rapport contient :
  - Les coordonnées précises du point central et le rayon du buffer.
  - Un tableau clair listant toutes les entités détectées dans la zone tampon, avec retour à la ligne automatique dans la colonne « Couche » pour les noms longs.
  - Des statistiques synthétiques :
    - Nombre total d'entités détectées
    - Distance moyenne, minimale et maximale entre le point central et les entités
    - Surface totale et moyenne des entités détectées
- ✓ Le rapport est généré uniquement sur demande explicite de l'utilisateur via ce bouton, et sauvegardé dans un dossier choisi par l'utilisateur.

V.2 Gestion des couches vectorielles et spatiales

Le plugin exploite les capacités spatiales de QGIS pour :

- Créer dynamiquement un buffer autour du point central
- Rechercher les intersections avec les entités des couches de dérogation et d'exclusion
- Calculer les distances entre le centre du buffer et les entités détectées

V.3 Interface utilisateur

- Combo box pour la sélection de la couche de dérogation
- Champs pour la saisie ou la sélection du point central
- Cases à cocher dynamiques pour la sélection des couches d'exclusion
- Liste interactive des entités détectées

V.4 Génération de rapport PDF

- Un bouton dédié avec icône flèche vers le bas permet de générer manuellement un rapport PDF
- Le rapport contient :
    - Tableau des entités détectées (avec retour à la ligne automatique pour les noms longs)
    - Statistiques : nombre total, distances moyenne/min/max, surface totale et moyenne
    - Coordonnées du point central et rayon du buffer

V.5 Interaction avancée

- Liste interactive : cliquer sur une entité la met en surbrillance sur la carte et centre la vue sur celle-ci

V.6 Difficultés rencontrées et solutions apportées

- **Gestion des widgets personnalisés** : adaptation de Qt Designer et gestion dynamique des widgets
- **Synchronisation des couches et des données** : mise à jour automatique des listes et des surbrillances
- **Gestion des projections** : utilisation systématique des outils de reprojection de QGIS pour garantir la cohérence spatiale
- La gestion des projections est assurée pour que les calculs de distances et surfaces soient cohérents quel que soit le système de coordonnées des couches.

- Le rapport PDF est généré avec la bibliothèque reportlab, utilisant des tableaux stylisés et une mise en forme adaptée pour la lisibilité.
- L'interface est développée avec PyQt5, intégrant des widgets dynamiques et interactifs pour la sélection des couches et la navigation dans les résultats.

VI. Résultats et démonstration

VI.1 Scénarios d'utilisation combinés

Le plugin a été testé dans des scénarios typiques où l'utilisateur télécharge d'abord des images satellites pour une zone d'étude, puis effectue une vérification des dérogations sur cette même zone. Cette approche intégrée facilite la prise de décision et la conformité réglementaire.

VI.2 Captures d'écran et exemples

Téléchargement des images satellites.

Cette interface permet à l'utilisateur de définir une zone d'étude (via des coordonnées BBOX ou un tracé interactif), de filtrer les images disponibles par date et couverture nuageuse, et de lancer une recherche via des API comme SentinelHub. Les résultats s'affichent sous forme de tableau listant les produits (ID, date, capteur), avec des options pour télécharger les images en GeoTIFF. L'utilisateur peut voir les miniatures des images disponibles avant de les télécharger. Un bouton "Show" permet de visualiser la zone sélectionnée sur la carte QGIS.

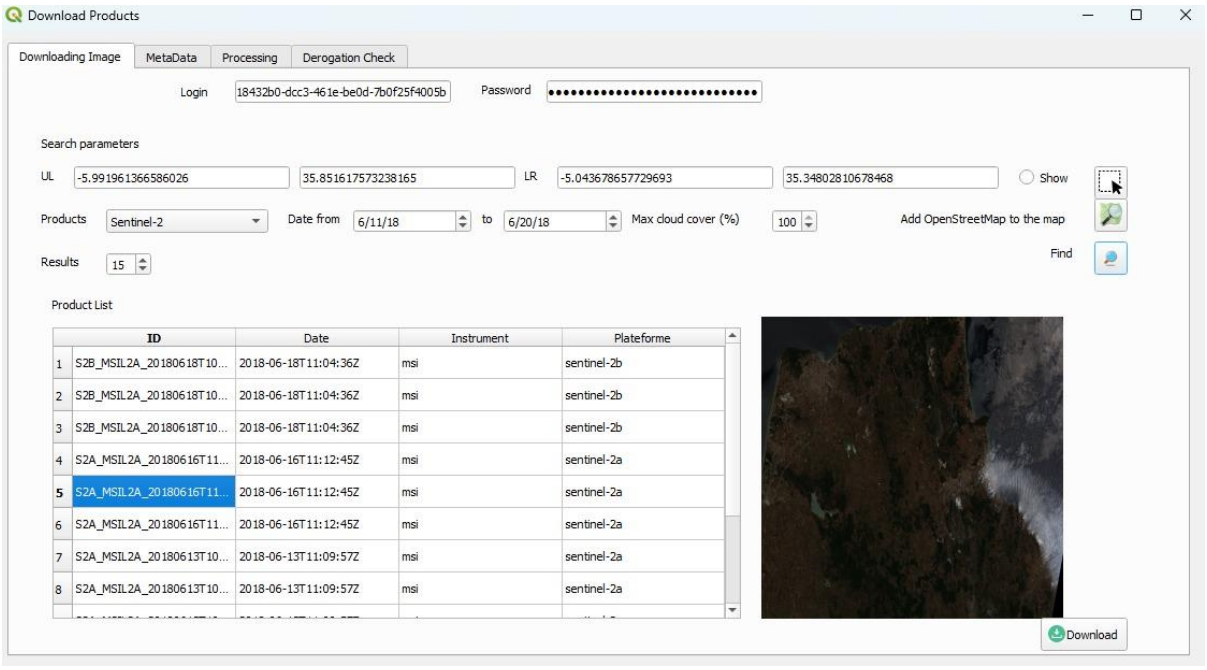


Figure 6 : Interface de téléchargement d'images.

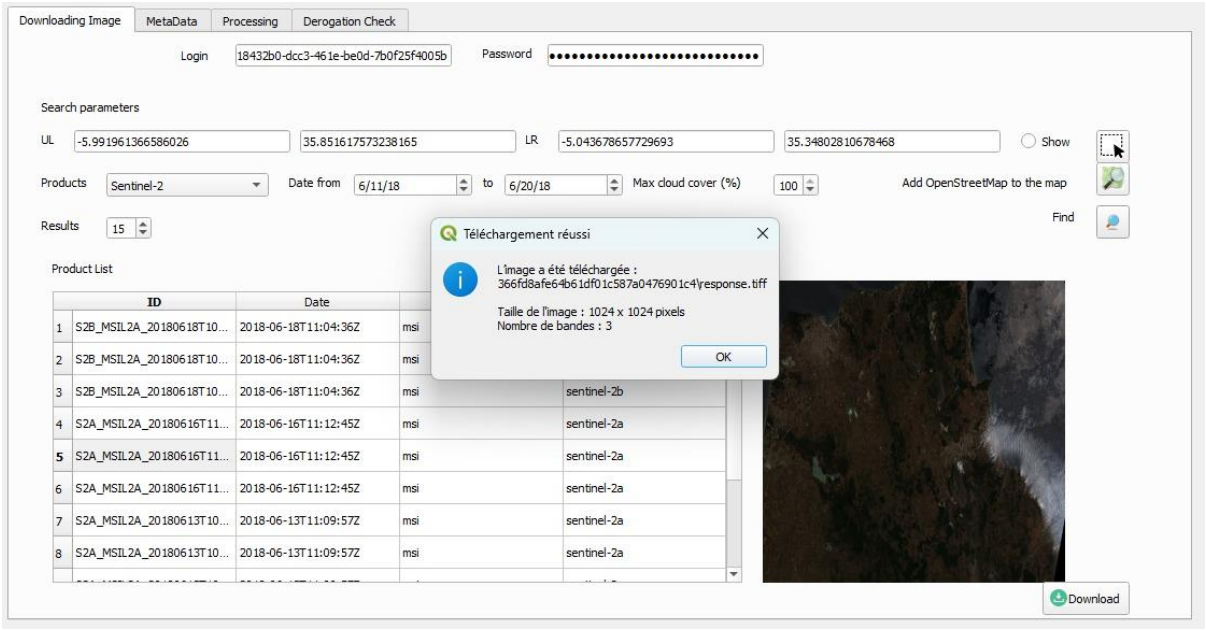


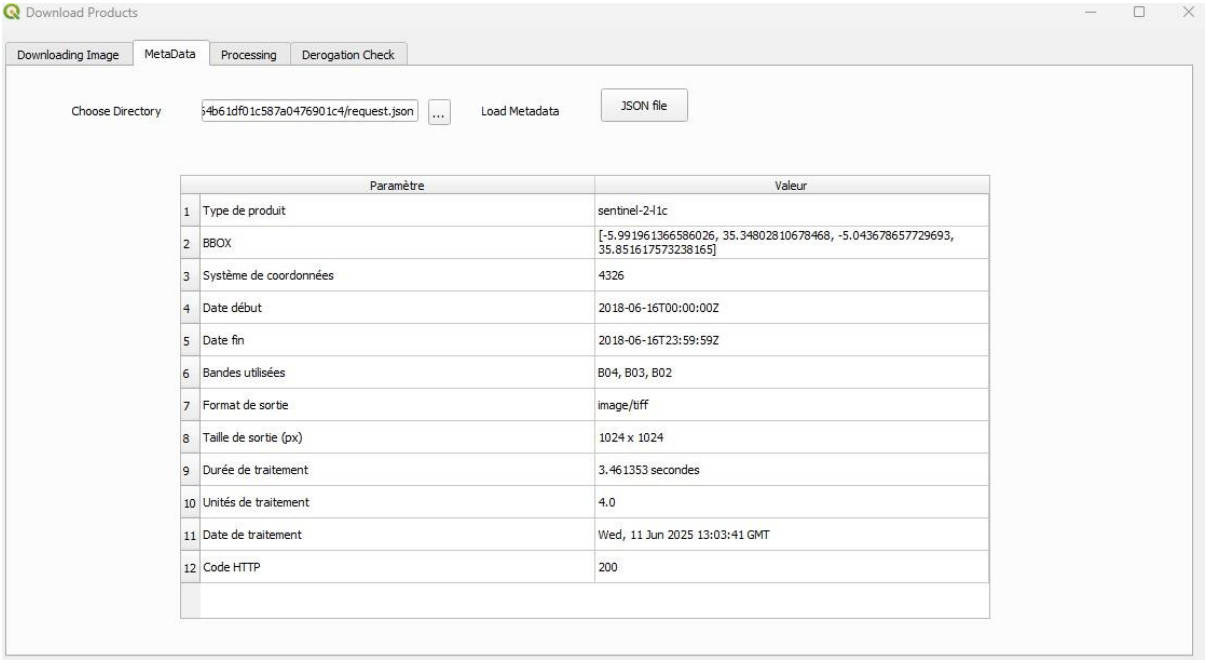
Figure 7 : Images correctement téléchargées.

-Après le téléchargement de l'image, l'utilisateur peut afficher l'image satellite (qui est déjà géoréférencée) sur QGIS.

Figure 8 : Affichage de l'image sur QGIS avec un fond open Street Map.

Interface des Métadonnées de l'image

Après téléchargement, cette section affiche les métadonnées techniques de l'image (système de coordonnées, bandes spectrales utilisées, résolution, etc.) dans un tableau clair. Les données sont extraites d'un fichier JSON généré automatiquement lors du téléchargement, offrant une traçabilité complète (dates de traitement, taille de l'image, code HTTP de la requête). L'interface inclut un bouton pour charger manuellement ce fichier si nécessaire.



	Paramètre	Valeur
1	Type de produit	sentinel-2-l1c
2	BBOX	[-5.991961366586026, 35.34802810678468, -5.043678657729693, 35.851617573238165]
3	Système de coordonnées	4326
4	Date début	2018-06-16T00:00:00Z
5	Date fin	2018-06-16T23:59:59Z
6	Bandes utilisées	B04, B03, B02
7	Format de sortie	image/tiff
8	Taille de sortie (px)	1024 x 1024
9	Durée de traitement	3.461353 secondes
10	Unités de traitement	4.0
11	Date de traitement	Wed, 11 Jun 2025 13:03:41 GMT
12	Code HTTP	200

Figure 9 : Interface de visualisation des Meta données de l'image.

Interface de Processing

Dédiée au traitement avancé, cette interface propose une liste d'indices (NDVI, NDWI, SAVI, etc.) avec leurs formules et usages explicites. L'utilisateur sélectionne un thème (végétation, eau, sol) et lance le calcul. Les résultats incluent une matrice de corrélation (heatmap) pour analyser les relations entre indices (ex: NDVI vs NDWI) et des options pour exporter les couches raster vers QGIS.

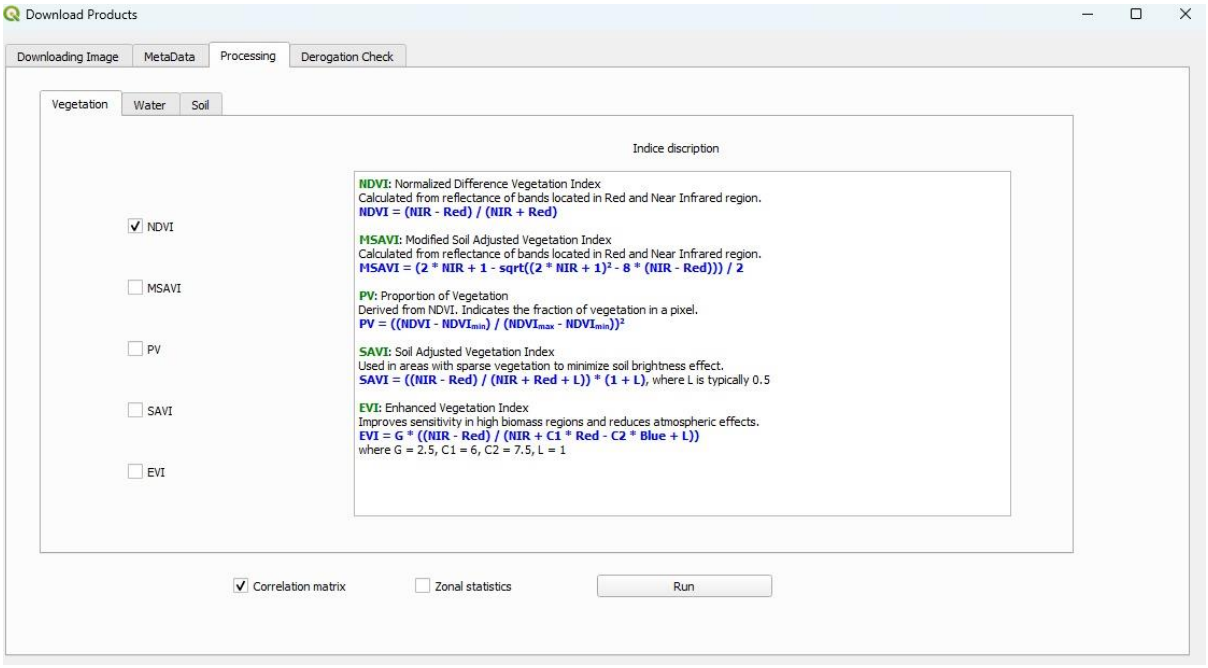


Figure 10 : Interface de traitement de l'image (indices spectraux)

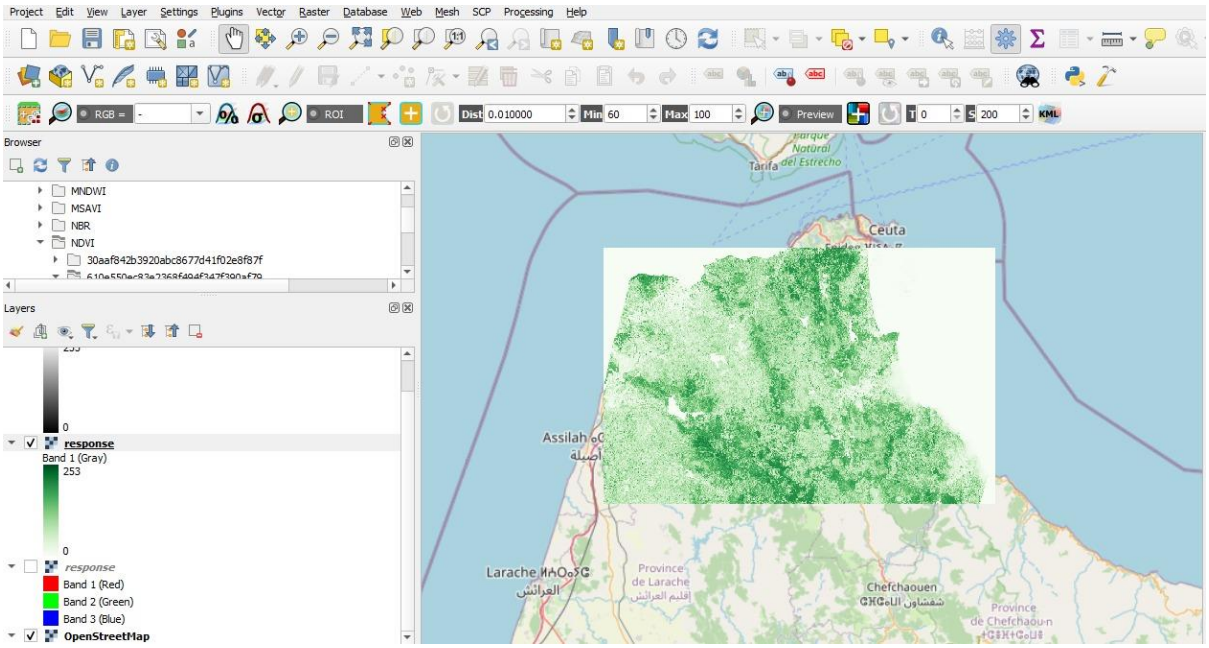


Figure 11 : Affichage du traitement NDVI de l'image.

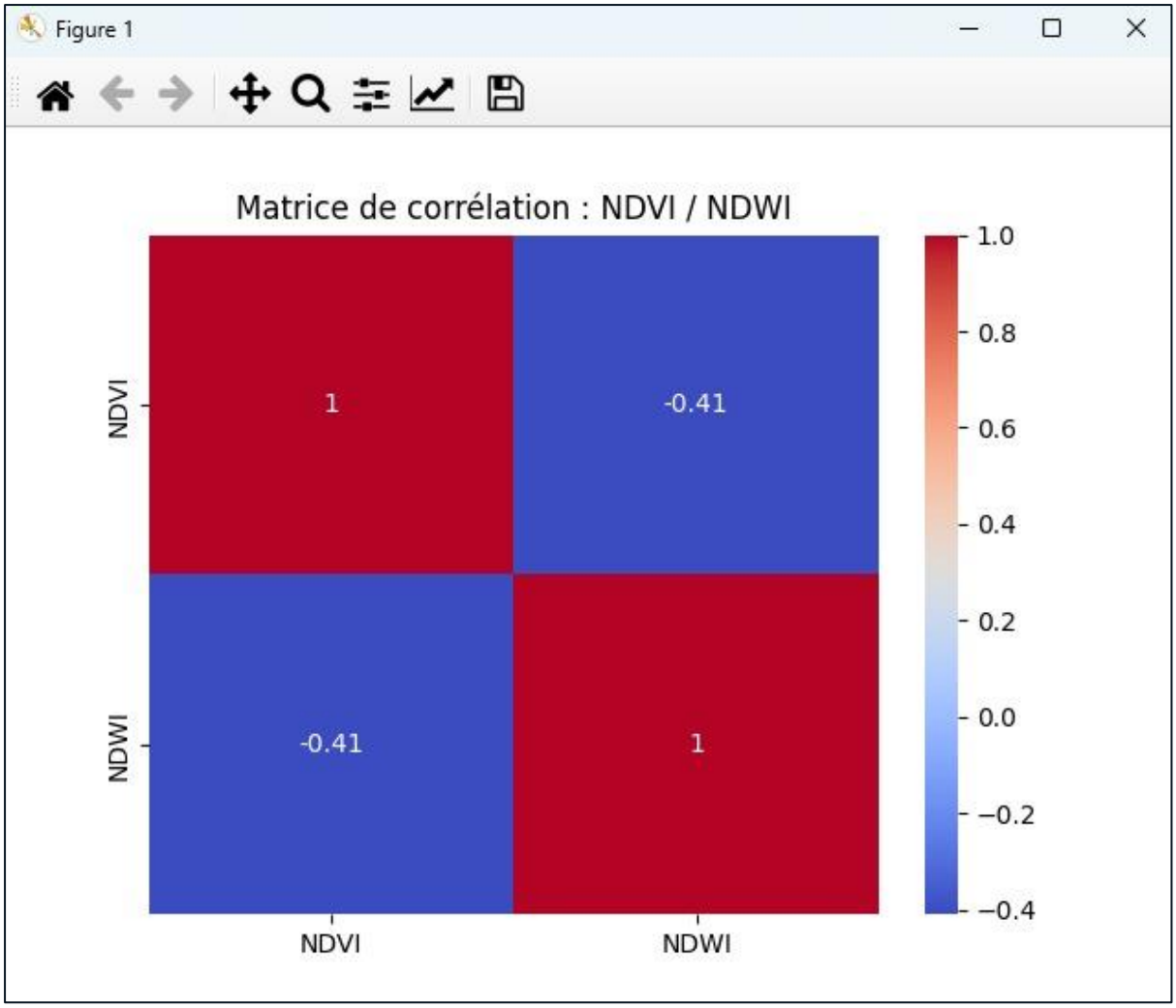


Figure 12 : Matrice de corrélation entre le NDVI et le NDWI sur l'image.

Dérogation

Sélection de la Zone de Vérification

Cette partie permet de configurer la zone d'analyse des dérogations en trois étapes simples : sélection de la couche de dérogation principale, définition du point central (manuellement ou par clic sur la carte), et spécification du rayon de recherche. Les couches d'exclusion (domaines forestiers, communaux, etc.) peuvent être activées via des cases à cocher. Un bouton "Vérifier dérogations" lance l'analyse spatiale, offrant une prise en main intuitive pour délimiter précisément la zone d'étude.

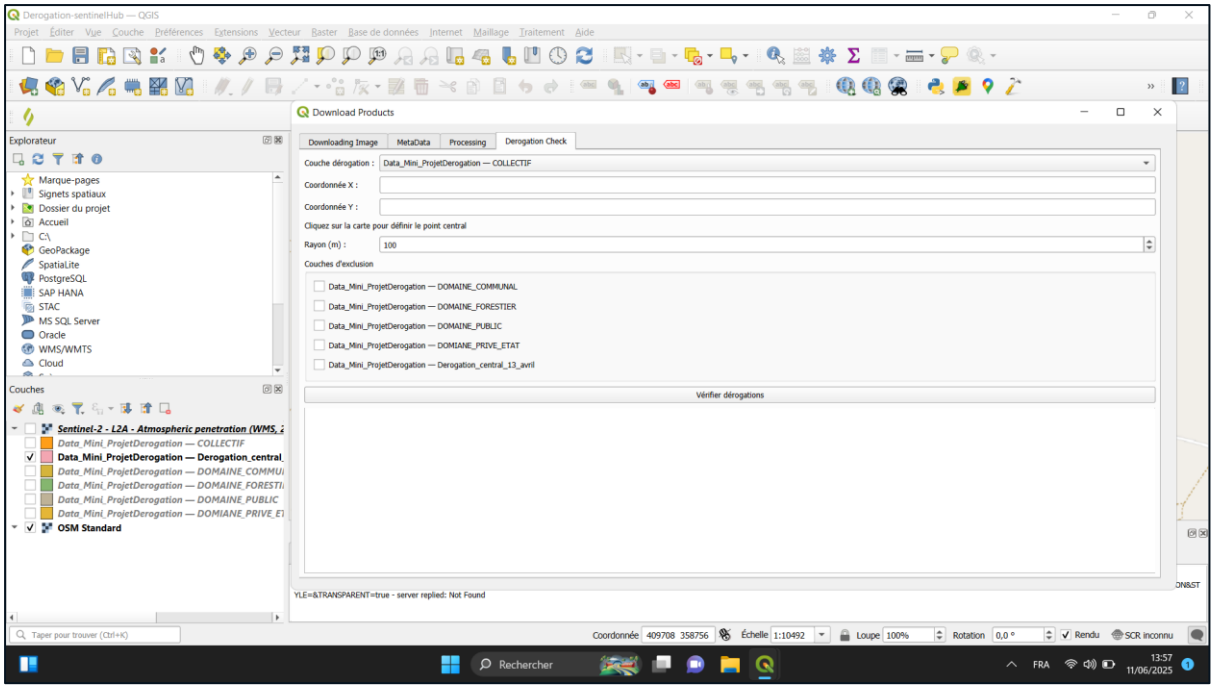


Figure 12 : Interface de vérification de la dérogation.

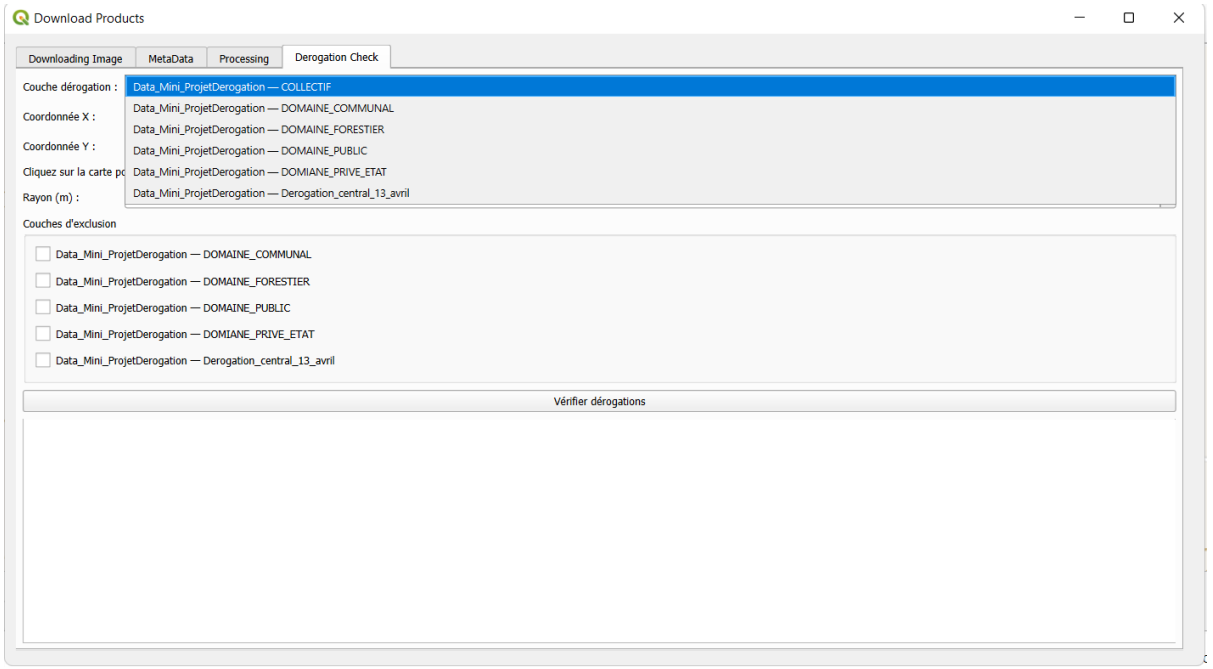


Figure 13 : Choix de la couche principale contenant les entités déjà dérogées.

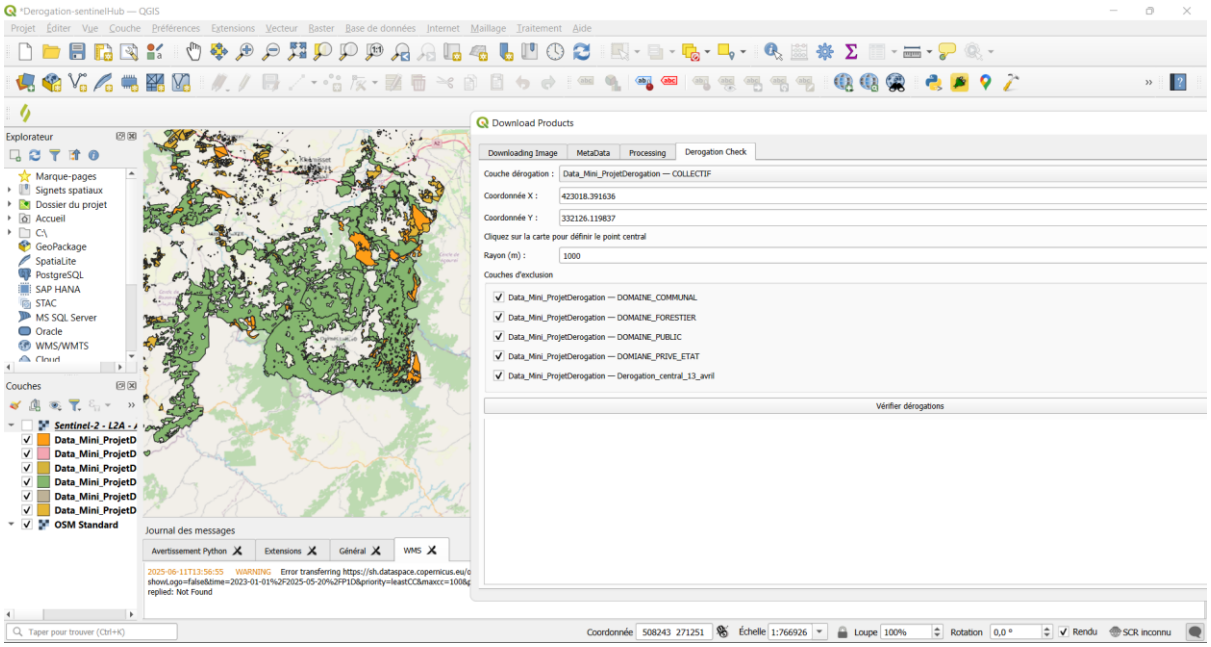


Figure 14 : Remplissage de tous les paramètres nécessaires pour la vérification.

Résultats de Vérification

L'analyse génère une liste détaillée des entités intersectant la zone tampon, présentant pour chacune son ID, sa couche d'origine et sa distance au point central. Interactive, elle permet de visualiser les éléments directement sur la carte QGIS par un simple clic. Les statistiques synthétiques (nombre d'entités, distances extrêmes) fournissent une vue d'ensemble rapide, tandis qu'un bouton dédié initie la création d'un rapport PDF pour archivage ou partage.

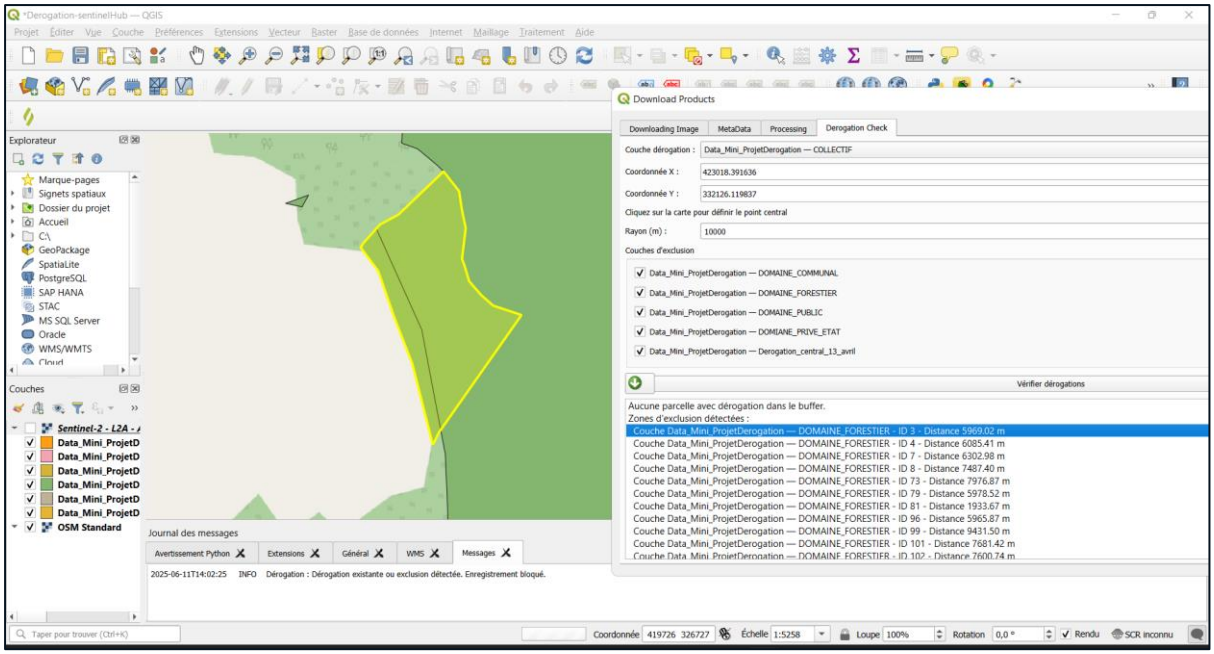


Figure 15 : Résultats de la vérification de la dérogation.

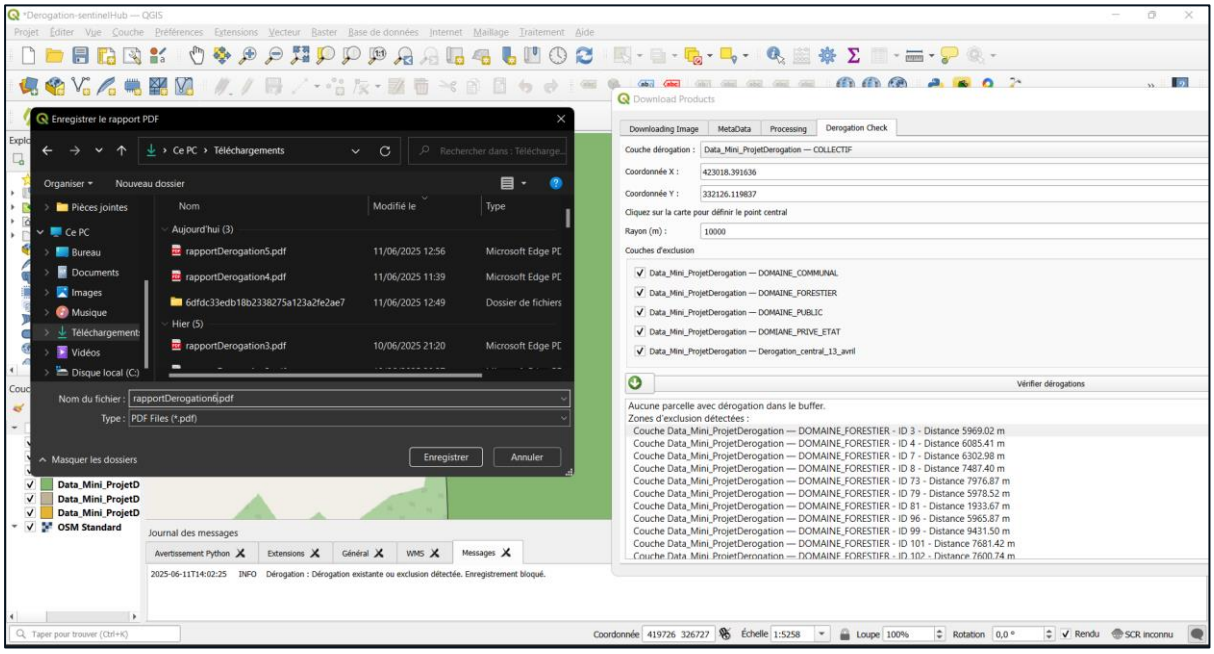


Figure 16 : Génération du rapport de vérification de la dérogation.

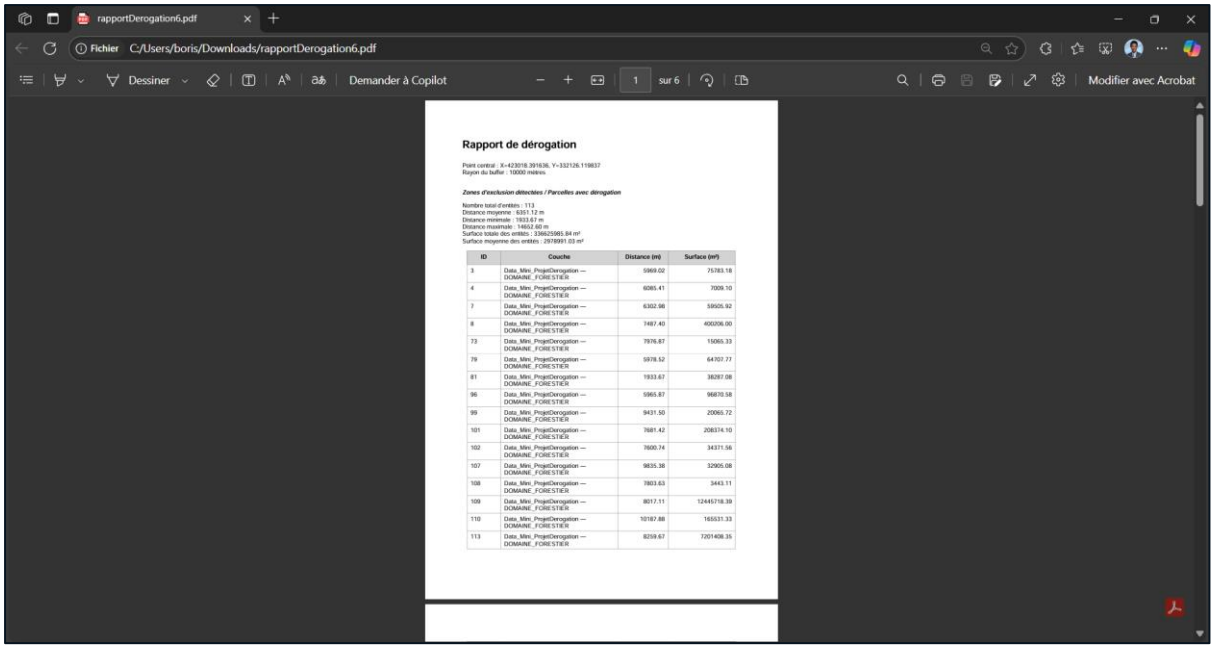


Figure 17 : Rapport de vérification de la dérogation.

Lorsqu'aucune parcelle de terrain n'a été trouvée portant déjà une dérogation dans le buffer : une simple notification apparait, il n'y a pas de rapport PDF dans ce cas.

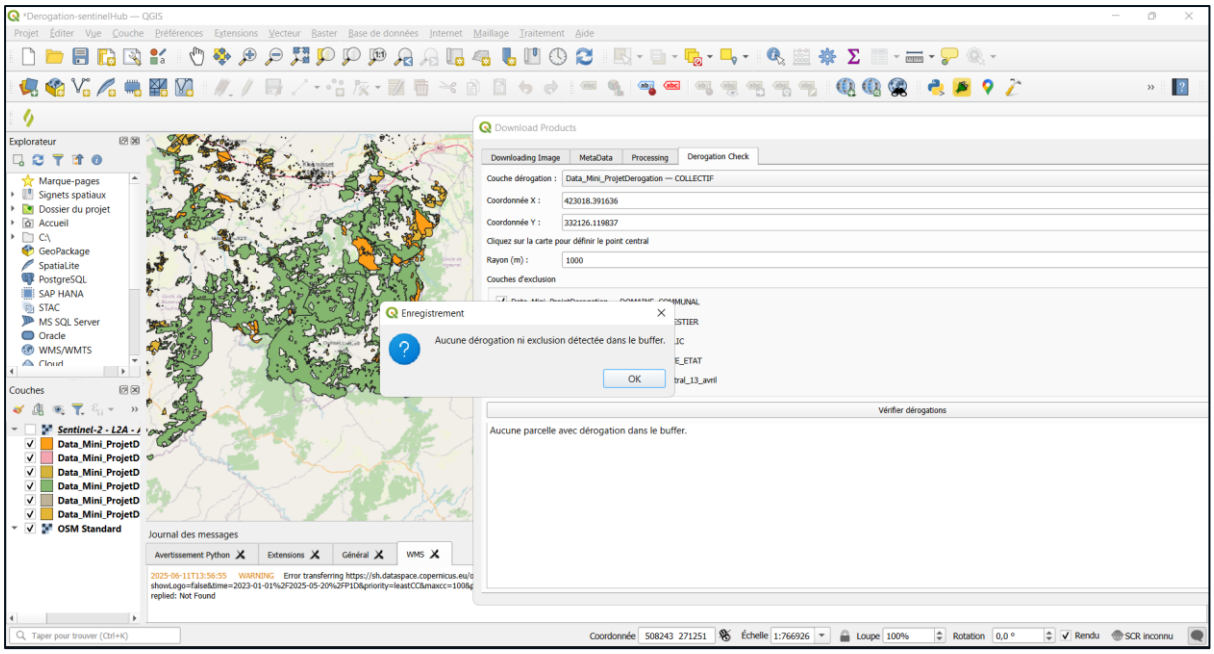


Figure 18 : Message affiché en cas de résultat nul.

VII. Bilan et perspectives

VII.1 Bilan global du projet et des fonctionnalités intégrées

Le développement du plugin QGIS intégrant à la fois le téléchargement d'images satellites et la vérification des dérogations a permis de proposer un outil complet et adapté aux besoins des gestionnaires fonciers et environnementaux.

VII.2 Apports pour les utilisateurs et impact métier

Le plugin facilite la prise de décision en combinant données satellitaires actualisées et contrôle réglementaire, améliorant ainsi la gestion durable des territoires.

VII.3 Perspectives d'évolution et améliorations futures

Des pistes d'amélioration incluent :

- L'intégration de nouvelles sources d'images satellites
- Le renforcement des fonctionnalités collaboratives et du partage de résultats

Conclusion

Le développement de ce plugin QGIS a permis de répondre à un double besoin essentiel dans le domaine de la géomatique appliquée : d'une part, l'accès simplifié à des images satellites pertinentes pour l'analyse environnementale et territoriale, et d'autre part, le contrôle automatisé des dérogations réglementaires à travers des traitements spatiaux rigoureux. Grâce à l'intégration d'outils tels que PyQt5, Sentinelsat, Rasterio ou encore ReportLab, ce projet offre une solution complète, interactive et adaptable aux exigences des utilisateurs professionnels.

Les deux fonctionnalités majeures du plugin – le téléchargement d'images et la vérification des dérogations – ont été conçues pour fonctionner de manière complémentaire, tout en assurant une interface conviviale et un haut degré d'autonomie pour l'utilisateur. Le système de génération de rapports PDF renforce la dimension professionnelle de l'outil, facilitant la restitution des résultats et leur archivage.

Ce projet constitue ainsi une contribution concrète à l'amélioration des pratiques d'analyse spatiale dans QGIS. Des perspectives d'évolution sont envisagées, notamment l'ajout de nouvelles sources d'images, l'intégration de traitements plus avancés (classification, détection de changements) ou encore le développement de fonctionnalités collaboratives pour le partage des résultats.

Annexes

A. Code source principale du plugin

- ✓ Lien GitHub pour accéder au code source **:**

https://github.com/Boris-Samne/PluginQGIS_SatelliteImagesDownload_DerogationCheck.git

- ✓ Extraits du code python du plugin :
  - **Téléchargement d'images satellites.**

Handle_rectangle() : Permet de sélectionner la zone d'intérêt et de récupérer les coordonnées.

```python
def handle_rectangle(self, rect):
    if rect.isEmpty():
        self.iface.messageBar().pushCritical("Erreur", "La zone sélectionnée est vide.")
        return

    crs_src = self.iface.mapCanvas().mapSettings().destinationCrs()
    crs_dest = QgsCoordinateReferenceSystem('EPSG:4326')
    xform = QgsCoordinateTransform(crs_src, crs_dest, QgsProject.instance())
    try:
        rect_wgs84 = xform.transformBoundingBox(rect)
    except Exception as e:
        self.iface.messageBar().pushCritical("Erreur", f"Transformation CRS échouée : {e}")
        return

    #crs_src = self.iface.mapCanvas().mapSettings().destinationCrs()
    #crs_dest = QgsCoordinateReferenceSystem('EPSG:4326')
    xform = QgsCoordinateTransform(crs_src, crs_dest, QgsProject.instance())
    rect_wgs84 = xform.transformBoundingBox(rect)
    # Extraire les coordonnées
    x_min = rect_wgs84.xMinimum()  # UL longitude
    y_max = rect_wgs84.yMaximum()  # UL latitude
    x_max = rect_wgs84.xMaximum()  # LR longitude
    y_min = rect_wgs84.yMinimum()  # LR latitude

    # Afficher dans la console (facultatif)
    print(f"UL: ({x_min}, {y_max})")
    print(f"LR: ({x_max}, {y_min})")

    # Afficher dans la barre d'info
    self.iface.messageBar().pushMessage("Zone sélectionnée", rect_wgs84.toString(), level=0)

    # Remplir les lineEdit du formulaire
    self.dlg.lineEdit.setText(str(x_min))     # UL Longitude
    self.dlg.lineEdit_2.setText(str(y_max))   # UL Latitude
    self.dlg.lineEdit_4.setText(str(x_max))   # LR Longitude
    self.dlg.lineEdit_3.setText(str(y_min))   # LR Latitude

    print("Rectangle sélectionné (WGS84) :", rect_wgs84.toString())

    self.iface.messageBar().pushMessage("Zone sélectionnée", rect_wgs84.toString(), level=0)

    # Si tu veux zoomer dessus :
    self.iface.mapCanvas().setExtent(rect)
    self.iface.mapCanvas().refresh()
```

Display_preview() : permet d'afficher l'image sélectionnée depuis la table.

```python
def display_preview(self, bbox, date_str, config):
    request = SentinelHubRequest(
        data_folder='.',
        evalscript="""
            function setup() {
                return {
                    input: ["B04", "B03", "B02"],
                    output: { bands: 3, sampleType: "UINT8" }
                };
            }

            function evaluatePixel(sample) {
                return [sample.B04*255, sample.B03*255, sample.B02*255];
            }
        """,
        input_data=[
            SentinelHubRequest.input_data(
                data_collection=DataCollection.SENTINEL2_L2A,
                time_interval=(date_str, date_str),
                mosaicking_order="leastCC"
            )
        ],
        responses=[SentinelHubRequest.output_response('default', MimeType.PNG)],
        bbox=bbox,
        size=(1024, 1024),
        config=config
    )

    try:
        image_data = request.get_data()[0]
        # Vérification de la forme
        if image_data.ndim != 3 or image_data.shape[2] != 3:
            QMessageBox.critical(None, "Erreur d'affichage", "L'image n'est pas au format RGB.")
            return
        print("Image shape:", image_data.shape)
        print("Max pixel value:", image_data.max())
        print("Min pixel value:", image_data.min())
        height, width, channel = image_data.shape
        bytes_per_line = 3 * width
        qimage = QImage(image_data.data, width, height, bytes_per_line, QImage.Format_RGB888)
        pixmap = QPixmap.fromImage(qimage)

        self.dlg.label_14.setPixmap(pixmap.scaled(
```

Load_metadata() : Permet d'afficher les données depuis un fichier json.

```python
def load_metadata(self):
    # Ouvre le fichier JSON
    directory = self.dlg.lineEdit_8.text()
    json_path, _ = QFileDialog.getOpenFileName(None, "Choisir un fichier JSON", directory, "JSON Files (*.json)")
    if not json_path:
        return

    with open(json_path, "r") as f:
        data = json.load(f)

    payload = data["request"]["payload"]
    response = data["response"]
    headers = response["headers"]

    # Liste des valeurs à afficher
    info_list = [
        ("Type de produit", payload["input"]["data"][0]["type"]),
        ("BBOX", str(payload["input"]["bounds"]["bbox"])),
        ("Système de coordonnées", payload["input"]["bounds"]["properties"]["crs"].split("/")[-1]),
        ("Date début", payload["input"]["data"][0]["dataFilter"]["timeRange"]["from"]),
        ("Date fin", payload["input"]["data"][0]["dataFilter"]["timeRange"]["to"]),
        ("Bandes utilisées", "B04, B03, B02"),
        ("Format de sortie", payload["output"]["responses"][0]["format"]["type"]),
        ("Taille de sortie (px)", f"{payload['output']['width']} x {payload['output']['height']}"),
        ("Durée de traitement", f"{response['elapsed']} secondes"),
        ("Unités de traitement", headers.get("x-processingunits-spent", "N/A")),
        ("Date de traitement", headers.get("Date", "N/A")),
        ("Code HTTP", str(response["status_code"]))
    ]

    # Préparer le tableau
    self.dlg.tableWidget_2.setRowCount(len(info_list))
    self.dlg.tableWidget_2.setColumnCount(2)
    self.dlg.tableWidget_2.setHorizontalHeaderLabels(["Paramètre", "Valeur"])

    for row, (key, value) in enumerate(info_list):
        self.dlg.tableWidget_2.setItem(row, 0, QTableWidgetItem(key))
        self.dlg.tableWidget_2.setItem(row, 1, QTableWidgetItem(value))
    self.dlg.tableWidget_2.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
```

find_product() : Permet de chercher les images depuis sentinelhub et de les afficher dans la table.

```python
def find_products(self):
    # Étape 1 : récupération des paramètres depuis l'interface
    ul_lon = float(self.dlg.lineEdit.text())
    ul_lat = float(self.dlg.lineEdit_2.text())
    lr_lon = float(self.dlg.lineEdit_4.text())
    lr_lat = float(self.dlg.lineEdit_3.text())

    start_date = self.dlg.dateEdit.date().toString("yyyy-MM-dd")
    end_date = self.dlg.dateEdit_2.date().toString("yyyy-MM-dd")
    label = self.dlg.comboBox.currentText()
    collection_type = self.product_map.get(label)
    cloud_cover = self.dlg.spinBox.value()
    username = self.dlg.lineEdit_7.text()
    password = self.dlg.lineEdit_5.text()
    max_results = self.dlg.spinBox_2.value()


    # Étape 2 : Configuration de Sentinel Hub
    config = SHConfig()
    #config.sh_client_id = '318432b0-dcc3-461e-be0d-7b0f25f4005b'
    #config.sh_client_secret = '8IKPep3PPwRNrP4J5DKj4OS9LyIA0sM1'
    config.sh_client_id = username
    config.sh_client_secret = password
    # Étape 3 : Définir la zone d'intérêt (bounding box)
    #bbox = BBox(bbox=[ul_lon, lr_lat, lr_lon, ul_lat], crs=CRS.WGS84)
    bbox = BBox(bbox=[ul_lon, ul_lat, lr_lon, lr_lat], crs=CRS.WGS84)


    # Étape 4 : Requête au catalogue Sentinel Hub
    catalog = SentinelHubCatalog(config=config)
    #cql_filter = Filter(f"eo:cloud_cover < {cloud_cover}")
    # Définir le filtre en fonction du type de collection
    if collection_type in ['sentinel-2-l2a', 'sentinel-2-l1c', 'sentinel-3']:
        filter_expression = f"eo:cloud_cover < {cloud_cover}"
    else:
        filter_expression = None

    results = catalog.search(
        collection=collection_type,
        bbox=bbox,
        time=(start_date, end_date),
        #filter="eo:cloud_cover < {cloud_cover}",
```

download_selected_image() : Permet de télécharger l'image.

compute_correlation_matrix() : pour calculer la corrélation entre deux indices...

- **Dérogation**

populate_derogation_layers() : charge dans la liste déroulante les couches vectorielles disponibles comme candidates à la dérogation.

```python
Tabnine | Edit | Test | Explain | Document
def populate_derogation_layers(self):
    self.comboBoxDerogLayer.clear()
    self.layer_map = {}
    for layer in self.project.mapLayers().values():
        if isinstance(layer, QgsVectorLayer):
            self.comboBoxDerogLayer.addItem(layer.name())
            self.layer_map[layer.name()] = layer
```

on_check_derogation() : exécute toute la logique de vérification (buffer, entités intersectées, messages et affichage).

```python
def on_check_derogation(self):
    derog_layer = self.get_selected_derogation_layer()
    if derog_layer is None:
        self.iface.messageBar().pushWarning("Dérogation", "Veuillez sélectionner une couche de dérogation.")
        return

    point = self.get_point()
    if point is None:
        self.iface.messageBar().pushWarning("Dérogation", "Veuillez définir un point central valide (clic ou coordonnées).")
        return

    radius = self.spinBoxRadius.value()
    buffer_geom = QgsGeometry.fromPointXY(point).buffer(radius, 20)

    derog_features = [
        (feat, dist, derog_layer)
        for feat, dist in self.get_features_in_buffer(derog_layer, buffer_geom, check_derog=True)
    ]

    exclusion_features = []
    for cb, layer in zip(self.exclusion_checkboxes, self.exclusion_layers):
        if cb.isChecked():
            exclusion_features.extend(
                [(feat, dist, layer) for feat, dist in self.get_features_in_buffer(layer, buffer_geom)]
            )

    self.listWidgetResults.clear()
    self.result_features = []

    if derog_features:
        self.listWidgetResults.addItem("Parcelles avec dérogation dans le buffer :")
        for feat, dist, layer in derog_features:
            self.listWidgetResults.addItem(f"  Couche {layer.name()} - ID {feat.id()} - Distance {dist:.2f} m")
            self.result_features.append((feat, layer))
    else:
        self.listWidgetResults.addItem("Aucune parcelle avec dérogation dans le buffer.")

    if exclusion_features:
        self.listWidgetResults.addItem("Zones d'exclusion détectées :")
        self.btnDownloadPdf.setVisible(True)
        for feat, dist, layer in exclusion_features:
            self.listWidgetResults.addItem(f"  Couche {layer.name()} - ID {feat.id()} - Distance {dist:.2f} m")
            self.result_features.append((feat, layer))
    else:
```

generate_pdf_report(point, radius, derog_features, exclusion_features) : génère un rapport PDF contenant les résultats détectés dans le buffer (entités, distances, surfaces…)...

```
Tabnine | Edit | Test | Explain | Document
def generate_pdf_report(self, point, radius, derog_features, exclusion_features):
    filename, _ = QFileDialog.getSaveFileName(self.parent, "Enregistrer le rapport PDF", "", "PDF Files (*.pdf)")
    if not filename:
        return

    doc = SimpleDocTemplate(filename, pagesize=A4,
                            rightMargin=2*cm, leftMargin=2*cm,
                            topMargin=2*cm, bottomMargin=2*cm)
    elements = []
    styles = getSampleStyleSheet()
    styleH = styles['Heading1']
    styleN = styles['Normal']
    styleBH = styles['Heading4']

    elements.append(Paragraph("Rapport de dérogation", styleH))
    elements.append(Spacer(1, 12))

    elements.append(Paragraph(f"Point central : X={point.x():.6f}, Y={point.y():.6f}", styleN))
    elements.append(Paragraph(f"Rayon du buffer : {radius} mètres", styleN))
    elements.append(Spacer(1, 12))

    def create_feature_table(title, features):
        elements.append(Paragraph(title, styleBH))
        elements.append(Spacer(1, 6))
        if len(features)==0:
            elements.append(Paragraph("Aucune entité détectée.", styleN))
            elements.append(Spacer(1, 12))
            return

        distances = [dist for _, dist, _ in features]
        surfaces = []
        for feat, _, _ in features:
            geom = feat.geometry()
            if geom:
                surfaces.append(geom.area())
        total_surface = sum(surfaces) if surfaces else 0
        avg_surface = total_surface / len(features) if features else 0

        elements.append(Paragraph(f"Nombre total d'entités : {len(features)}", styleN))
        elements.append(Paragraph(f"Distance moyenne : {sum(distances)/len(distances):.2f} m", styleN))
        elements.append(Paragraph(f"Distance minimale : {min(distances):.2f} m", styleN))
        elements.append(Paragraph(f"Distance maximale : {max(distances):.2f} m", styleN))
        elements.append(Paragraph(f"Surface totale des entités : {total_surface:.2f} m²", styleN))
```

B. Documentation technique complémentaire (Installation des dépendances nécessaires)

Commande à exécuter sur OSGeo Shell (ligne de commande de QGIS) pour pouvoir utiliser correctement le plugin.

- ✓ **python -m pip install --upgrade pip**
- ✓ **python -m pip install rasterio geopandas pandas seaborn matplotlib rasterstats sentinelsat sentinelhub**