GEO Nature

MISE EN PLACE DE GEONATURE POUR LA GESTION DES ESPACES PROTÉGÉS DE LA LPO FRANCE

Rapport de Stage

Maxime TOMA

Année universitaire 2019-2020

Licence Professionnelle S.I.G. - Université de La Rochelle

Lieu du stage: Ligue pour la Protection des Oiseaux (LPO), Rochefort

LUPSIG La Rochelle

La Rochelle Université

LPO

AGIR pour la BIODIVERSITÉ

FICHE D'INFORMATIONS

AGIR pour la BIODIVERSITÉ

Ligue pour la Protection des Oiseaux

Pôle Protection de la nature – Service Espaces protégés

Lieu du stage :
Fonderies Royales
8 rue du Docteur Pujos
CS 90263 Rochefort CEDEX 17305 France

Etudiant :
Maxime TOMA, Etudiant LUPSIG – maxime.toma@gmail.com

Maitre de stage :
Frédéric ROBIN, Chargé de mission scientifique – frederic.robin@lpo.fr

---

Tuteur de stage :
Alain LAYEC, Enseignant BDD/Python LUPSIG – alain.layec@univ-lr.fr

---

Période du stage :
25/05/2020 au 11/09/2020

SOMMAIRE

FICHE D'INFORMATIONS **2**

SOMMAIRE **3**

AVANT-PROPOS **4**

REMERCIEMENTS **4**

SIGLES ET ACRONYMES **5**

INTRODUCTION **7**

I/ CONTEXTE DU PROJET **8**

1. L'ORGANISME D'ACCUEIL 8
2. LE CONTEXTE DU STAGE 11

II/ ORGANISATION ET PLANIFICATION DU STAGE **14**

1. ETUDE DES BESOINS ET POSSIBILITES 14
2. DEFINITION DES OBJECTIFS DU STAGE 17
3. PLANIFICATION DES TACHES 18

III/ CONCEPTION DU STAGE **21**

1. ETAT DES CONNAISSANCES ET COMPETENCES PERSONNELLES 21
2. ETAT DE L'EXISTANT 22
3. ETAT DES UTILISATEURS ET DES BESOINS 25

IV/ REALISATION DU STAGE **28**

1. PRE-TRAITEMENT DES DONNEES 28
2. PRESENTATION DE L'APPLICATION GEONATURE 28
3. CONFIGURATION DES MODULES BASIQUES DE GEONATURE 31
4. INSTALLATION ET CONFIGURATION DE MODULES SUPPLEMENTAIRES 39
5. INSERTION DES DONNEES HISTORIQUES 42
6. REALISATION D'UN PLUGIN QGIS D'IMPORT MASSIF DE DONNEES 51
7. PERSONALISATION DE L'INTERFACE PRINCIPALE DE GEONATURE & REDACTION DES GUIDES 60

V/ BILAN DU DEPLOIMENT DE GEONATURE **62**

1. DIFFICULTES RENCONTREES 62
2. COUTS 63
3. CONTINUITE DU PROJET 64

CONCLUSION **65**

TABLE DES FIGURES **66**

BIBLIOGRAPHIE ET WEBOGRAPHIE **68**

COMPREHENSION DU PROJET / DE L'APPLICATION 68
BASE DE DONNEES / SQL 69
REALISATION DU PLUGIN QGIS / ELABORATION DES SCRIPTS PYTHON 70

TABLE DES ANNEXES **71**

AVANT-PROPOS

Le présent rapport développe la mise en place de GeoNature pour le Service Espaces Protégés de la LPO France, dans le cadre d'un stage pour la Licence professionnelle SIG 2019-2020 de l'Université de La Rochelle.

Les outils suivants ont été utilisés pour la durée totale de ce projet sous ces versions : GeoNature 2.3.4 et 2.4.1 (et les versions des modules compatibles), QGIS 3.10, PyCharm 2020.1.2, Python 3.7, PuTTy 0.73, Office 2013 et 2019, Windows Server 2012 R2, Brackets 1.14.2.0, FileZilla 3.48.1.0, PostgreSQL 11.7, pgAdmin4 4.21.0.0.

REMERCIEMENTS

Je souhaite avant tout remercier l'ensemble de la Ligue pour la Protection des Oiseaux de France représenté par son président Allain BOUGRAIN-DUBOURG et son secrétaire général exécutif Olivier DENOUE de m'avoir accueilli comme stagiaire et de m'avoir donné l'opportunité de réaliser mon stage de fin de licence professionnelle au sein du Pôle Protection de la Nature.

Je tiens sincèrement à remercier mon maitre de stage Frédéric ROBIN (responsable de la coordination des réserves naturelles gérées par la LPO) qui m'a encadré tout au long de ce projet, qui a su répondre à l'ensemble de mes questions et qui m'a accordé sa confiance pour la réalisation de cette tâche. De plus, je souhaite remercier chaleureusement Maxime ZUCCA (Directeur du Pôle Protection de la Nature), Ségolène TRAVICHON (Responsable du Service Espaces protégés) et Laurent COUZI (Responsable du Service Connaissance) de m'avoir apporté leur aide et leur soutien.

Je remercie également le Service Informatique, représenté par Serge THOMAS et François BOUCHET pour ce projet, qui m'ont accordé leur confiance tout au long du projet pour les manipulations effectuées au sein des serveurs de la LPO.

Je remercie tout particulièrement Camille MONCHICOURT (Responsable SI au Parc national des Ecrins et développeur de GeoNature) pour avoir répondu à toutes mes questions concernant l'application GeoNature et de m'avoir tenu au courant des nouveautés apportées à celui-ci.

De plus, j'adresse mes remerciements à Frédéric POUGET, Alain LAYEC ainsi que l'ensemble des professeurs et intervenants de la Licence professionnelle universitaire Systèmes d'Information Géographique de La Rochelle pour leur disponibilité, leur attention et l'ensemble des savoirs qu'ils ont sus me transmettre tout au long de cette année universitaire.

Enfin, je remercie très chaleureusement ma famille et mes amis, en particulier Justine VALENTIN (amie et ancienne étudiante de la licence), pour leurs soutiens et leurs conseils et sans qui je n'aurais jamais pu entreprendre mes projets d'études dans ce domaine.

SIGLES ET ACRONYMES

CEN
Conservatoire des Espaces Naturels 25

CRUVED
Create, Read, Update, Validate, Export, Delete 33, 34

CSS
Cascading Style Sheets (Feuille de style en cascade) 17, 21, 59

FTP
File Transfert Protocol 15, 59

HTML
Hypertext Markup Language (Langage de balise pour l'Hypertexte) 17, 21

IGN 16
Institut national de l'Information Géographique et Forestière 12

IHM
Interface Homme-Machine 50, 68

LPO
Ligue pour la Protection des Oiseaux 4, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 23, 24, 26, 28, 32, 60, 62, 68

MCD
Modèle Conceptuel de Données 42

MNT
Modèle Numérique de Terrain 29, 43

PNR
Parc Naturel National 25

RNMO
Réserves Naturelles Nationales Moëze-Oléron 39

SEP
Service Espaces Protégés 33, 34, 38

SERENA
Système d'Echange de données pour les Réseaux d'Espaces NAturels 11, 12, 13, 14, 15, 17, 18, 33

SFTP
Secure File Transfert Protocol 15

SGBD
Système de Gestion de Base de Données 12, 14, 15, 21

SI
Service Informatique 4, 22, 29, 39, 41, 65, 66

SINP
Système d'Information sur la Nature et les Paysages 12, 16, 22, 32, 35, 36, 37, 38, 62

SQL
Structured Query Language (Langage de requête structurée) 16, 17, 21, 22, 32, 33

SRID
Spatial Reference Identifier (Identifiant du Système de coordonnées) 31, 37, 39, 41

SSH
Secure Shell 15, 59

STEPRO
La Station de lagunage et Marais périurbain de Rochfort 48

INTRODUCTION

Dans le cadre de la licence professionnelle Systèmes d'Information Géographique de l'Université de La Rochelle, j'ai eu l'occasion de réaliser mon stage de fin d'études à la Ligue pour la Protection des Oiseaux. Les conditions particulières de cette année ont fait que j'ai réalisé entièrement ce stage en télétravail du 25 mai au 11 septembre 2020 sur une période de 16 semaines.

Le service Espaces Protégés (SEP) du Pôle Protection de la Nature de la LPO France travaille depuis plusieurs années avec l'outil SERENA pour la gestion des données de leurs réserves et espaces naturels. Cependant, cet outil n'ayant plus une très bonne maintenance, et du fait de son architecture et de son mode de fonctionnement, celui-ci devient inadapté aux grandes quantités de données géoréférencées.

Outil en plein essor au sein de la communauté naturaliste depuis presque 2 ans, GeoNature jouit de la reprise en 2017 de son développement par les Parcs Nationaux. Il dispose de nombreux modules permettant à ses utilisateurs de stocker efficacement leurs données naturalistes. L'application est entièrement basée sur les standards nationaux du SINP (Système d'Information de la Nature et des Paysages) et est également soutenue par le Ministère de la Transition Ecologique. Elle permet ainsi un partage standardisé d'informations entre tout organisme manipulant de la donnée naturaliste, et donnera à la LPO l'occasion de faciliter ses échanges avec ses partenaires.

C'est pourquoi il m'a été demandé par Frédéric ROBIN, mon maitre de stage et coordinateurs des réserves et espaces naturels, de :

« **Mettre en place l'outil libre GeoNature pour la gestion et le suivi de protocoles des Réserves Naturelles administrées par la LPO** »

Dans ce rapport, nous verrons dans un premier temps une présentation plus détaillée de la structure et du contexte. Dans un deuxième temps, nous étudierons les besoins du commanditaire ainsi que la planification de mon stage. Nous ferons, ensuite, un état des lieux de l'existant et de mes connaissances actuelles et à acquérir. Nous entamerons, par la suite, la partie sur la réalisation des livrables et de l'application des méthodes planifiées. Enfin, nous terminerons ce rapport par un bilan global et critique de mon stage et par mes impressions.

I/ CONTEXTE DU PROJET

1. L'organisme d'accueil

a) LA LIGUE POUR LA PROTECTION DES OISEAUX DE FRANCE

La Ligue de la Protection des Oiseaux (LPO) est aujourd'hui la première association de protection de la nature en France. Constituée de plus de 50 000 adhérents, 5 000 bénévoles actifs, plus de 400 salariés sur le territoire national, et d'un réseau d'associations locales actives dans 83 départements, elle œuvre au quotidien pour la protection des espèces, la préservation des espaces et pour l'éducation et la sensibilisation à l'environnement.

Partenaire officiel en France du réseau BirdLife International, la LPO met en œuvre des plans nationaux de restauration d'oiseaux parmi les plus menacés de France, et coordonne des programmes européens de sauvegarde et de réintroduction d'espèces menacées.

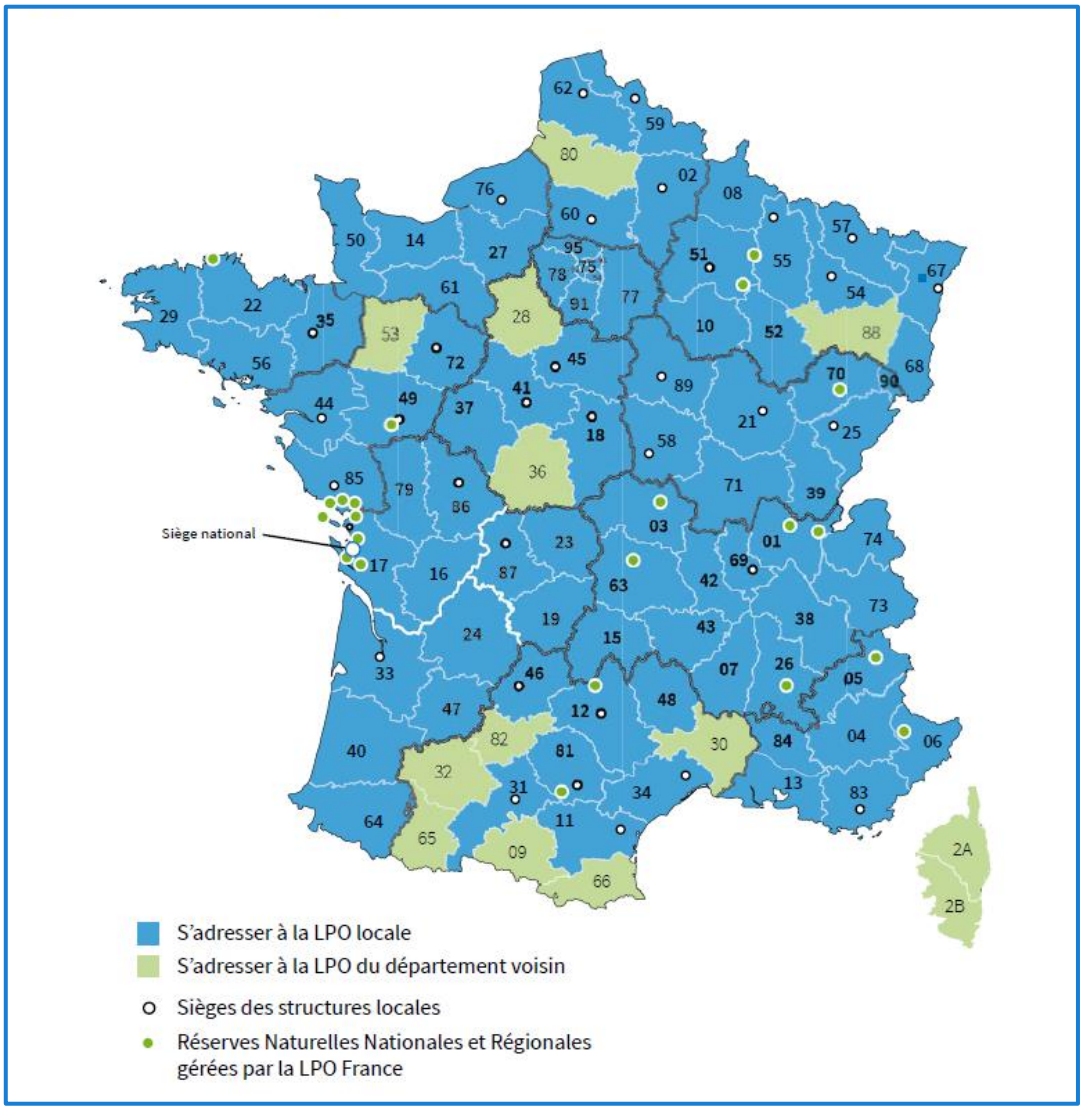


Figure 1: Carte de répartition des antennes LPO (Source : lpo.fr)

I. Historique

La création de l'association remonte à 1912. A l'époque, le lieutenant Hémery, membre de la Société nationale d'Acclimatation de France, dénonce le massacre des macareux moines par les chasseurs sur les côtes nord de la Bretagne qui y organisaient des safaris. Cette espèce était alors très menacée et au bord de l'extinction. Cette action de protection de la nature fut ainsi la première d'une longue liste et aujourd'hui, la LPO porte sur son logo deux macareux moines en son hommage.

Figure 2: Lieutenant Hémery portant deux macareux moines (Source : lpo.fr)

De ce même élan en septembre 1912 est née la Réserve naturelle des Sept-Îles, première réserve de France (privée jusqu'en 1976).

Depuis 2012, la LPO a décidé d'étendre ses actions à la protection de la biodiversité qui est menacée dans son ensemble.

II. Missions principales

Aujourd'hui, la LPO concentre une multitude d'actions, sur la biodiversité tout entière, qui s'articulent autour de trois missions principales :

- **La Protection des espèces**

Cette protection passe par la mise en place d'enquêtes et suivis scientifiques afin d'acquérir toujours plus de connaissances sur les oiseaux et leurs habitats naturels.

- **La Conservation des espaces**

La LPO est un gestionnaire majeur pour la préservation et gestion des espaces naturels terrestres et côtiers (comme la gestion des réserves naturelles nationales et régionales ou l'acquisition foncière d'espaces naturels). Elle gère également un programme national appelé « Refuges LPO » composé d'un réseau de particuliers, collectivités, écoles ou entreprises, engagés dans une démarche écocitoyenne de protection de la nature.

- **L'éducation et la sensibilisation**

La LPO propose également des conférences, expositions, sorties et animations nature au grand public ainsi qu'une édition de plusieurs médias (livres, magazines, revues ornithologiques, etc…).

III. Organisation de la « maison mère » et ma place en son sein

La « maison-mère » de la LPO France est basée à Rochefort en Charente-Maritime. Elle est composée de 450 salariés (regroupant les délégations territoriales). Voici un organigramme simplifié montrant ma position au sein de la structure. L'organigramme complet est disponible via le site de la LPO à cette adresse : https://www.lpo.fr/la-lpo/organigramme ou en Annexe I *(partie Service Espaces Protégés uniquement)*.

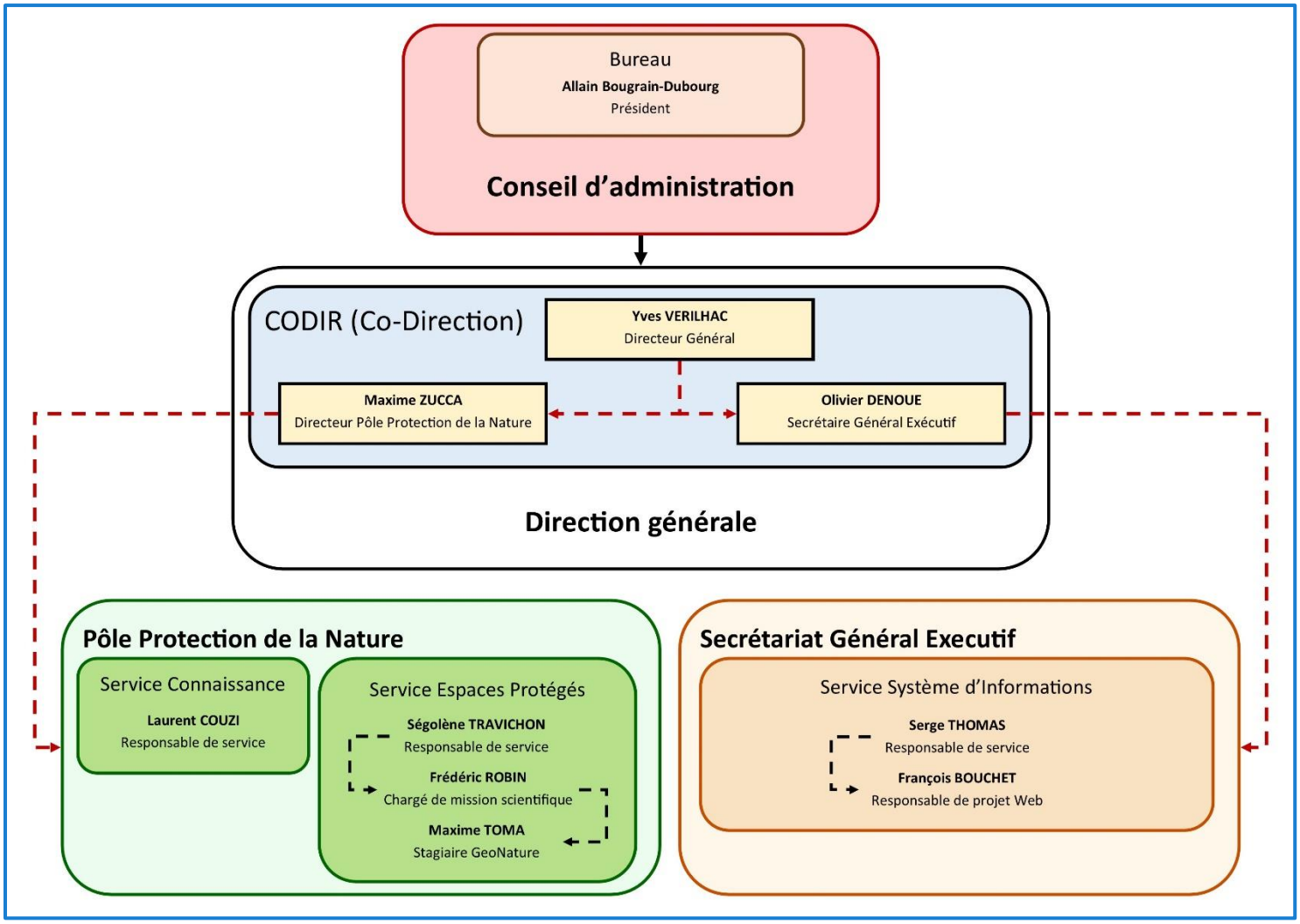


Figure 3: Organigramme simplifié de la LPO (Source : Maxime TOMA)

b) LE SERVICE ESPACES PROTEGES

La responsable de service est Ségolène TRAVICHON. Elle dirige les actions menées à travers les réserves naturelles et espaces protégés de la LPO.

Mon maitre de stage est Frédéric ROBIN. Il est chargé de mission scientifique et coordonne les conservateurs et gardes de ces réserves naturelles et espaces protégés.

Le service est composé de 11 conservateurs, 17 gardes, 7 chargés de missions, 4 animateurs et d'1 responsable de projet.

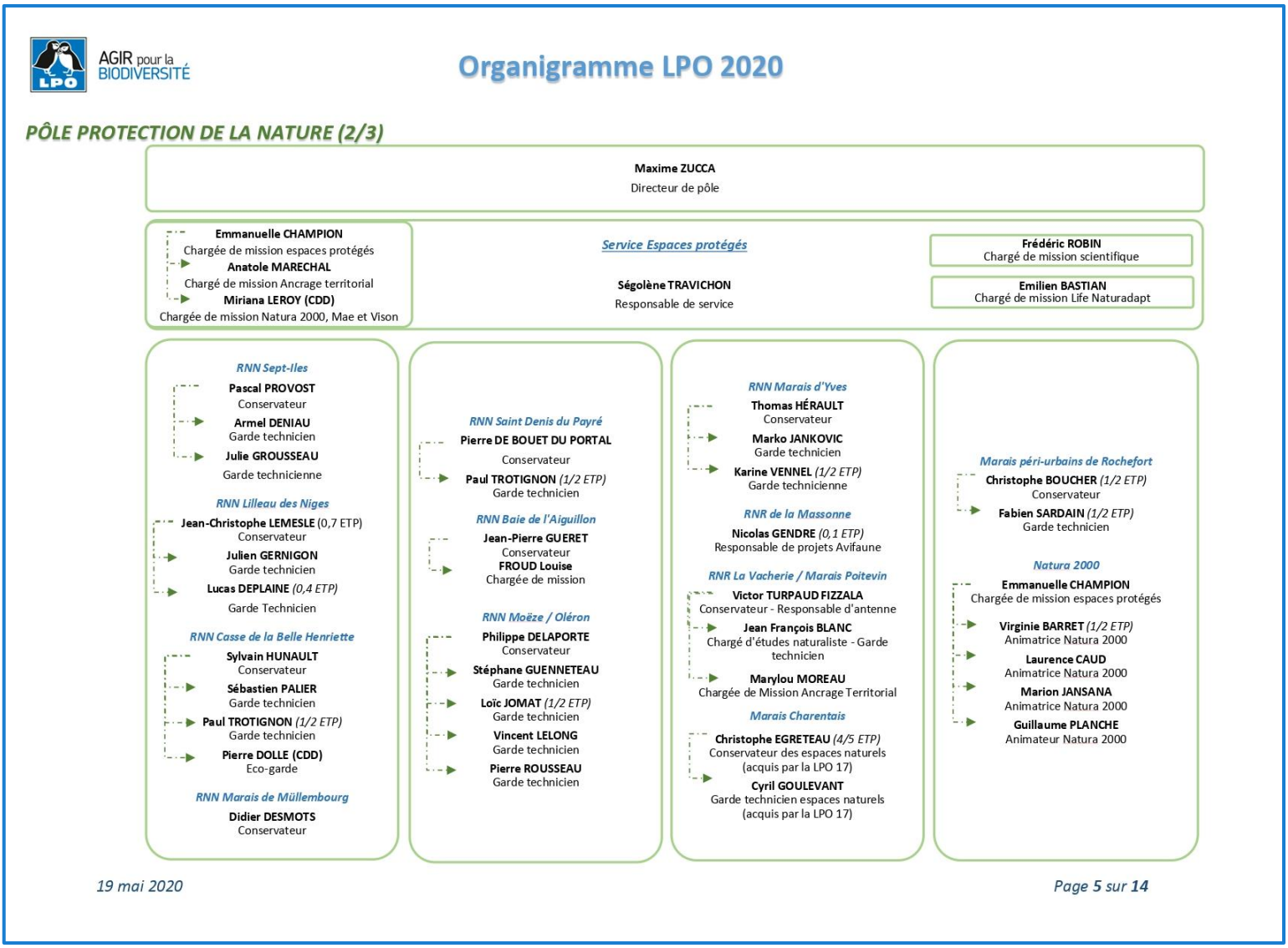


Figure 4: Organigramme détaillé du Service Espaces Protégés (Source : lpo.fr)

2. Le contexte du stage

a) SERENA : BASE HISTORIQUE DES RESERVES NATURELLES DE FRANCE

Dans les années 2000, les Réserves Naturelles de France (RNF) décidèrent de faire développer un logiciel de gestion de bases de données naturalistes. Ainsi, ce **S**ystème d'**E**change de données pour les **R**éseaux d'**E**spaces **NA**turels (**SERENA**), développé par Pierre Girard, permet aux « non-informaticiens » de créer et gérer facilement des bases de données faune-flore au format MS Access, ou dernièrement au format PostgreSQL.

Figure 5: Image de présentation de SERENA (Source : http://www.serena-rnf.net/v2/)

Cet outil a été adopté par la LPO, à l'époque, pour administrer les données des réserves naturelles qu'elle gère. Le grand avantage de cette base est qu'elle respecte les standards du SINP et permet d'introduire le référentiel taxonomique TaxRef.

Le **SINP**, pour Système d'Information sur la Nature et les Paysages, est un dispositif partenarial, collaboratif, entre le Ministère en charge de l'Écologie et les acteurs de la biodiversité et du paysage. Il permet, entre autres, de partager une même nomenclature des données naturalistes.

Aujourd'hui, l'outil SERENA est malheureusement dépassé pour cette gestion de base naturaliste, avec géométrie pour certaines données, car le développement s'en retrouve fortement ralenti par manque d'attractivité.

Le Système de Gestion de Base de Données (SGBD) qu'est MS Access ne gère pas la géométrie d'objets spatialisés et c'est sous cette forme que la LPO archivait les données de ses réserves et espaces naturels jusqu'à présent.

b) GEONATURE : UN OUTIL OPEN-SOURCE DEVELOPPE PAR LES PARCS NATIONNAUX

Figure 6: Logo de GeoNature (Source : geonature.fr)

Selon le document de présentation, **GeoNature** est « *un outil open source (sous licence libre) développé par les parcs nationaux des Ecrins et des Cévennes et qui a ainsi été déployé dans d'autres parcs nationaux et régionaux, des conservatoires d'espaces naturels, des conservatoires botaniques nationaux et des associations. Il est animé et maintenu par une communauté de développeurs, composée de différentes structures* ». Le projet a été lancé en 2013 par Gil DELUERMOZ (développeur du Parc National des Ecrins), mais a pleinement repris que depuis 2017.

Ainsi, GeoNature s'inscrit dans un réseau de collecte, de partage et de diffusion de données naturalistes à différentes échelles. Il s'appuie également, comme pour SERENA, sur les référentiels et standards nationaux (SINP, TaxRef, IGN, etc…).

Là où GeoNature se démarque de son prédécesseur sous MS Access, c'est que l'outil est sous licence libre, c'est-à-dire que n'importe qui peut apporter sa contribution dans son développement et l'utiliser librement. En revanche, le noyau principal de l'application est géré et développé par le service informatique des Parcs Nationaux.

De ce fait, il existe aujourd'hui plusieurs applications et modules englobant plusieurs besoins comme la synthèse de données, la saisie mobile, un atlas, la saisie de protocole simple ou complexe, la validation des données saisies, etc…

La base de données est gérée sous PostgreSQL avec, entre autres, la cartouche PostGIS permettant la gestion des objets géographiques. Nous reviendrons dans la partie *III/2.Etat de l'existant* et *IV/2.Présentaiton de l'application* pour plus de détail au sujet de l'application.

c) DEMANDE DU SERVICE ESPACES PROTEGES ET CONNAISSANCE

Un nouveau projet, initié depuis plus d'un an et piloté par Frédéric ROBIN en partenariat avec Laurent COUZI du Service Connaissance et François BOUCHET du Service Informatique de la LPO, a pour but de mettre en place l'outil GeoNature pour la gestion des données naturalistes autour des réserves naturelles dont la LPO France est gestionnaire.

Il s'agit également, pour le pôle Protection de la Nature, d'un premier pas sur cet outil qui souhaiterait l'utiliser, à terme, pour d'autres protocoles et programmes.

Ainsi, l'objectif serait de migrer les données historiques de SERENA vers GeoNature tout en mettant en place les outils proposés par la communauté de ce dernier.

d) ETENDU DU PROJET

Le projet s'inscrit à l'échelle de 11 réserves naturelles et espaces protégés, coordonnés par Frédéric ROBIN, représentant environ 13000 ha répartis sur 3 départements (Charente-Maritime, Vendée et Côtes-d'Armor) :

- Les Réserves Naturelles Nationales des Sept-Iles, de la Baie de l'Aiguillon, de la Casse de la Belle-Henriette, de Lilleau des Niges, de Saint-Denis du Payré, du Marais de Müllembourg, du Marais d'Yves et de Moëze-Oléron
- La Réserve Naturelle Régionale des Marais de la Vacherie
- Les Sites & Acquisitions du Marais Rochefortais
- La Station de lagunage et Marais périurbain de Rochefort

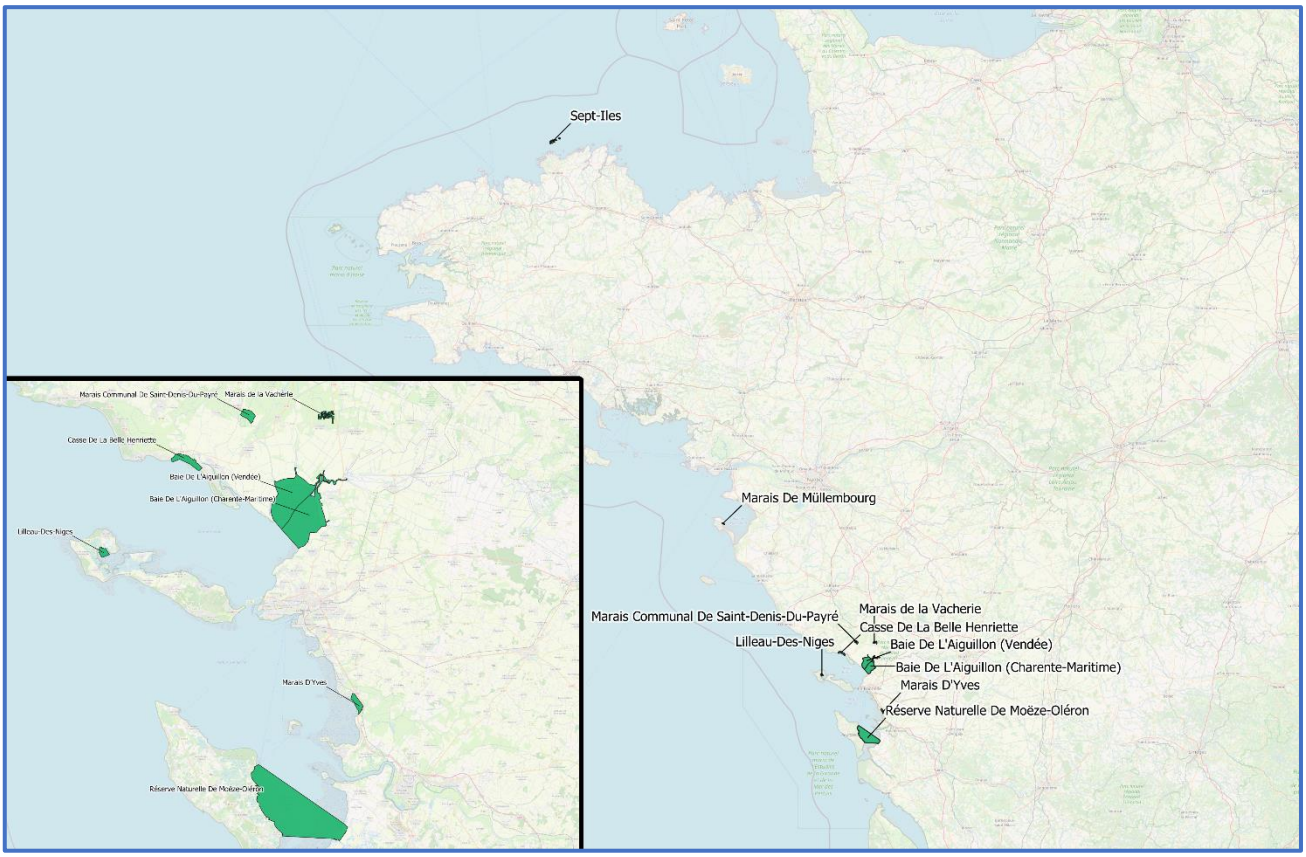


Figure 7: Répartition des différentes réserves naturelles gérées de la LPO (Source : Maxime TOMA)

II/ ORGANISATION ET PLANIFICATION DU STAGE

1. Etude des besoins et possibilités

a) VISUALISATION DES DONNEES

Après avoir obtenu la totalité des données nécessaires à la réalisation de ce projet, j'ai pu entreprendre une première visualisation globale des différents fichiers fournis, ce qui m'a permis de recenser un certain nombre de points flous, sur lesquelles j'avais besoin de plus d'informations et de clarté.

Ainsi, la donnée est présente sous différents formats (fichiers shapes, Excels, base Access SERENA) n'ayant pas les mêmes noms et nombre de champs entre les différents fichiers, protocoles et réserves.

b) ANALYSE DES BESOINS DU COMMANDITAIRE

Comme vu précédemment dans le point *I/2.Le contexte du stage*, Frédéric ROBIN souhaite mettre en place l'outil GeoNature pour la gestion des données des réserves naturelles de la LPO.

Ses besoins sont donc de centraliser les données existantes, actuellement dans la base Access, mais également les différents fichiers Excels, Shape, MapInfo, etc… dans une seule et même application.

Frédéric cherche également à saisir de façon massive les données sur l'outil. N'ayant qu'occasionnellement le temps d'informatiser les données quotidiennes de relevés naturalistes, les équipes restent très attachées à la méthode de remplissage d'un tableur Excel.

Autre besoin, Frédéric ne souhaite plus utiliser SERENA pour trois raisons :

- **L'impossibilité de traiter les données spatialisées** : Les données des réserves sont stockées sous format MS Access. Ce SGBD ne permet pas de gérer les données spatialisées. De plus, MS Access se retrouve limité en performance dès lors que la quantité de données est trop importante (comme c'est le cas avec les 610000 données actuelles des réserves).
  SERENA a récemment permis à ses utilisateurs de passer sur PostgreSQL. Une option assez facile d'utilisation permet de faire la transition. Mais cela ne résout pas les deux autres problèmes…
- **Le coût financier** : SERENA est un outil sous licence payante. Une version gratuite existe, mais un grand nombre de fonctionnalités utilisées par la LPO se retrouveraient indisponibles avec cette version.
- **La structure et fonctionnement de l'outil** : L'application adopte un mode de fonctionnement assez archaïque pour aujourd'hui. La base de données ne présente malheureusement aucune relation physique entre les tables parce qu'elles sont gérées par le logiciel directement.
  Certains champs spécifiques de certaines réserves/structures sont manquants dans le logiciel de base. Pour pallier ce problème, les développeurs ont rajouté la possibilité de créer des

« pseudo-champs » par les utilisateurs. Cependant, deux problèmes surviennent avec ce procédé sur SERENA : **il existe qu'un seul formulaire pour tous les protocoles** (donc présence de champs non utilisés par tel ou tel protocole et qui peuvent être néanmoins remplis) et donc **l'export de données ne dissocie pas les colonnes selon le protocole**. Ainsi, on se retrouve donc avec plus de 200 champs pour 12 utilisés, par exemple, pour l'export de données d'un seul protocole.
Enfin, un autre aspect de fonctionnement dérangeant de SERENA est le concept de base mère/base fille. Les réserves naturelles de la LPO possèdent chacune une base fille. Les données, attributs, utilisateurs, lieux, etc. des bases filles sont ensuite transmis à la base mère. Il existe donc un gros danger sur la propreté des données. Certains id, utilisateurs, lieux peuvent être doublés et/ou écrits de façon différente.

Ainsi, Frédéric a dû trouver une solution de remplacement gratuite, de préférence, et adaptée à la spatialisation des données (GeoNature).

c) ANALYSE DES BESOINS MATERIELS

Pour la réalisation de ce stage, en termes de choix du SGDB, il est imposé d'utiliser une base de données sous PostgreSQL, avec sa cartouche spatiale PostGIS. En plus d'être open source et très complet, il s'agit du SGBD utilisé par GeoNature.

GeoNature est un outil fonctionnant avec différents langages/librairies comme Python, Flask, Leaflet, Angular et Bootstrap. Il est donc nécessaire d'avoir un outil de rédaction de scripts sous différents langages.

Pour accéder au cœur de la configuration de GeoNature, il est nécessaire d'avoir un accès par FTP/SFTP ((Secure) File Transfer Protocol) pour la gestion des fichiers de l'application et un accès SSH (Secure Shell - invite de commande) pour exécuter des commandes spécifiques (ex : mise à jour de composants).

Enfin, un outil sera nécessaire pour l'import massif de données (méthode souhaitée par Frédéric ROBIN et le personnel des réserves).

d) MATERIEL MIS A DISPOSITION

Le contexte de ce stage se retrouve inédit par les conditions de la crise sanitaire due à la pandémie de la COVID-19. Ainsi, je ne dispose donc pas du matériel qu'aurait pu me fournir la LPO en temps normal.

Cependant, la LPO me fournit un accès VPN à distance à leurs serveurs afin d'avoir accès à un poste local donnant lui-même accès à tous les documents dont j'aurais besoin. La LPO met également à ma disposition un compte Microsoft donnant accès à la suite Office (Word, Excel, PowerPoint, Access, Outlook et Teams). Elle me donne aussi accès aux logiciels déjà installés sur leurs serveurs comme pgAdmin ou encore QGIS.

De mon côté, je dispose d'un bureau, d'un ordinateur portable sous Windows 10, d'un écran supplémentaire et d'une connexion internet assez bonne afin de disposer des meilleures conditions de télétravail possible.

e) METHODES ET LOGICIELS RETENUS POUR LA REALISATION DU PROJET

Revenons tout d'abord sur le choix de prendre GeoNature comme outil. Pourquoi lui en particulier ? Plusieurs raisons ont mené Frédéric et le Pôle Protection de la Nature à faire ce choix selon les points forts et points faibles de l'application :

Points forts	Points faibles
• Licence ouverte et gratuite (tout le monde peut développer et/ou faire force de propositions pour améliorer l'outil) • L'équipe de développement est très active et à l'écoute des utilisateurs • Utilise les référentiels du SINP • A le soutien du Ministère de la Transition Ecologique et Solidaire, du MNHN, de l'AFB et de l'IGN dans le but de proposer un outil de saisie des données brutes de la biodiversité aux maitres d'ouvrage, dans le cadre de la loi du 8 août 2016 sur la reconquête de la biodiversité et des paysages. • De plus en plus d'organismes/associations se mettent à l'utiliser	• Outil incomplet, manque des modules pour certains protocoles • Outil pas encore assez démocratisé

J'ai donc étudié les différentes options qui s'offraient à moi. Au regard de mes compétences sur les logiciels et les objectifs fixés, j'ai choisi de mettre en place les méthodes suivantes :

- Pour la configuration de GeoNature : Après m'être imprégné de la documentation de l'application, j'utiliserai un ou plusieurs outils pour accéder aux fichiers de configuration afin de personnaliser GeoNature vis-à-vis de l'utilisation que souhaitent en faire Frédéric et son équipe.
- Pour l'import massif de données historique : GeoNature propose d'importer massivement des données dans le module de Synthèse. Or, ces données ne peuvent pas être modifiées une fois dedans. Ainsi, je vais devoir m'imprégner du modèle de données de la base PostgreSQL utilisé par GeoNature afin de rédiger des scripts SQL (et éventuellement Python) pour l'intégration de ces données historiques.
- Pour l'import massif de données : Reprenant le même modèle d'insertion que pour les données historiques, je devrais développer un plugin QGIS pour effectuer des imports massifs de données issues de fichiers shapes/Excel. Cependant, les fichiers csv/shape devront être très restrictifs et tout le monde devra avoir le même fichier de base.
  J'ai choisi l'alternative de développer un plugin QGIS parce que je ne maitrise pas le développement sous Angular et Flask (principaux outils utilisés pour le développement d'un module GeoNature).

Pour mettre en place ces méthodes, j'aurais besoin des outils suivants :

OUTILS	JUSTIFICATIF
pgAdmin 4	Permet l'administration des bases PostgreSQL compatible aux versions 10 et ultérieure. Outil manipulé en cours.
FileZilla	Outil permettant le transfert de fichiers d'un serveur à une machine locale. Il est intégré à la liste des logiciels libres préconisés par l'État français dans le cadre de la modernisation globale de ses systèmes d'information (S.I.). Outil manipulé en cours.
PuTTy	PuTTy permet l'envoi de commande à travers une invite de commande via des chemins sécurisés. Outil manipulé en cours.
QGIS	SIG sous licence libre, il est également l'outil principal de la LPO en matière de géomatique. Outil manipulé en cours.
MS Excel	La LPO me met à disposition la suite Office. Excel permettra la manipulation des données .csv et .xlsx. Outil manipulé en cours.
MS Access	La LPO me met à disposition la suite Office. Access permettra la manipulation des bases mère et filles (SERENA) de la LPO. Outil manipulé en cours.
PyCharm	PyCharm s'inscrira dans la création de scripts python pour la création du plugin et l'intégration massive de données. Outil manipulé en cours.
QtDesigner / QtCreator	Complémentaires à la création du plugin, ces outils, manipulés en cours, permettront la création esthétique des formulaires du plugin sous QGIS.
Brackets	Utilitaire permettant le codage sous différents langages. Je l'utiliserai principalement pour mes codes en langage HTML, CSS et SQL. Outil manipulé en cours.

Tous ces outils ont été vus et manipulés en cours ou pendant de mon projet tutoré. Ils ont également l'avantage d'être gratuits (sauf pour la suite Office).

2. Définition des objectifs du stage

Afin de clarifier la demande et d'exposer comment nous allons procéder pour ce projet, Frédéric ROBIN m'a présenté tous les outils et toutes les données dont il dispose actuellement sur les réserves naturelles. Il m'a également exposé ses propres besoins et attentes (*voir II/1.b.Analyse des besoins du commanditaire*).

J'ai pu aborder les différentes solutions et éclaircir certains points encore flous de mon côté. J'ai alors mieux compris les attentes du commanditaire, ainsi que le travail à réaliser par rapport aux objectifs énoncés dans la commande.

L'objectif principal du projet est donc de « **Mettre en place l'outil GéoNature pour la gestion et le suivi de protocoles des Réserves Naturelles administrées par la LPO** ».

De cet objectif en découlent les sous objectifs suivants :

- Configurer les modules de base de GeoNature.
- Installer et configurer d'autres modules supplémentaires et utiles pour la LPO.
- Personnaliser l'outil (graphiquement, mais aussi au niveau des modules).
- Intégrer les données déjà existantes issues des fichiers Excels, Shapes, etc. de la LPO et de la base Access du logiciel SERENA dans GeoNature.
- Créer un plugin QGIS afin de faciliter l'import massif de données naturalistes dans GeoNature selon les modules.
- Créer des « fichiers types » au format .csv et .shp adaptés pour le plugin.

Pour des raisons de temps, je n'ai pas pu intégrer toutes les données historiques, mais cela a été vu avec mon maitre de stage et nous nous sommes mis d'accord sur sa non-faisabilité.

J'ai ainsi rédigé un cahier des charges que j'ai fait valider par mon maitre de stage. Ce cahier des charges reprenant le projet dans sa globalité, les problématiques actuelles, les objectifs et les attentes et besoins du commanditaire est disponible en Annexe II.

3. Planification des tâches

a) ORGANIGRAMME DES TACHES (Methode SADT)

Pour visualiser au mieux les étapes du projet, et ainsi passer des objectifs bien définis à l'analyse du projet, j'ai réalisé dans un premier temps un **Organigramme des Tâches** (OT) avec la **méthode SADT** (Structured Analysis Design Technic).

Il s'agit d'une méthode d'analyse visant à séparer les tâches en 6 boites maximum pouvant contenir elles-mêmes 6 autres boites maximum, etc. Pour chaque tâche, on définit les

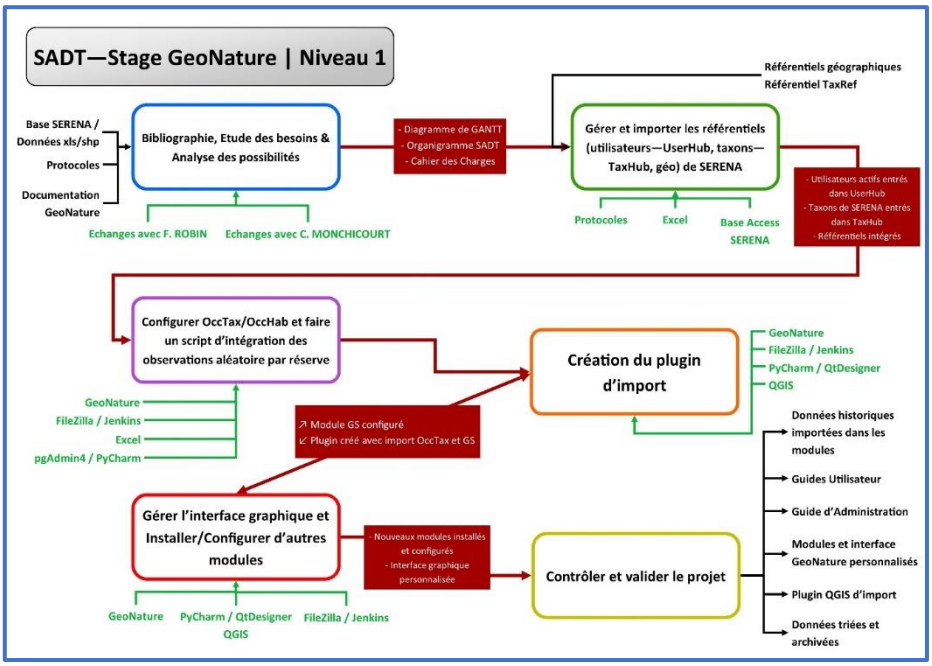


Figure 8: Niveau 1 de l'Organigramme des Tâches

moyens, les outils et les méthodes. On définit également l'enchainement de ces boites entre-elles.

L'Organigramme des Tâches complet est disponible en Annexe III. Cette hiérarchisation des tâches s'est aussi accompagnée de sa planification. Ainsi, j'ai décidé de réaliser un diagramme de GANTT.

b) DIAGRAMME DE GANTT PREVISIONNEL ET DEFINITIF

J'ai commencé par faire un **digramme de GANTT prévisionnel** afin de me guider temporairement dans le projet et ainsi aboutir à une version définitive correspondant au travail demandé (versions disponibles en Annexe IV et Annexe V).

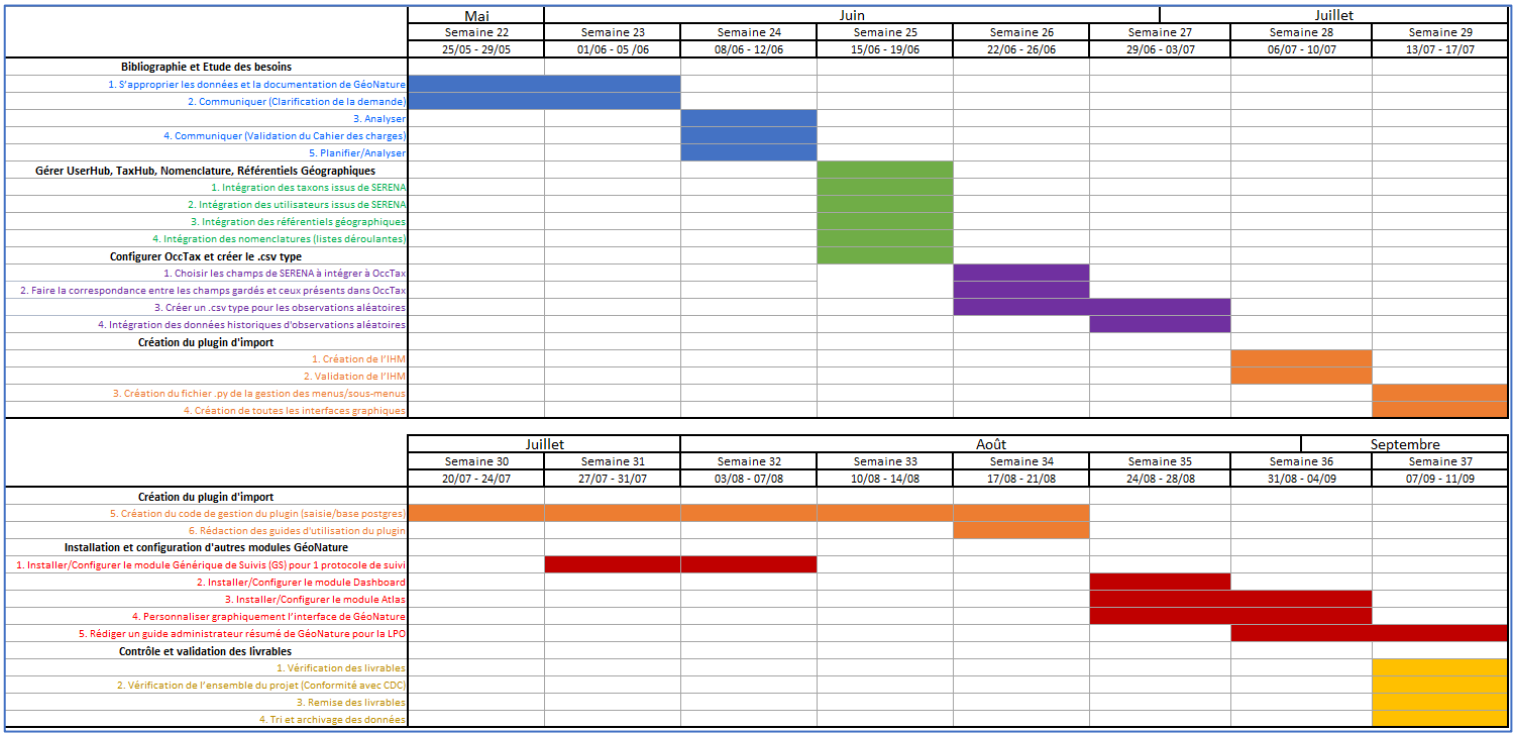


Figure 9: Planning GANTT prévisionnel

A noter que certains aléas comme des imprévus, congés, la date de sortie de certains modules et mise à jour de GeoNature ont perturbé le planning plusieurs fois. Ainsi on arrive à une forme finale du diagramme de GANTT :

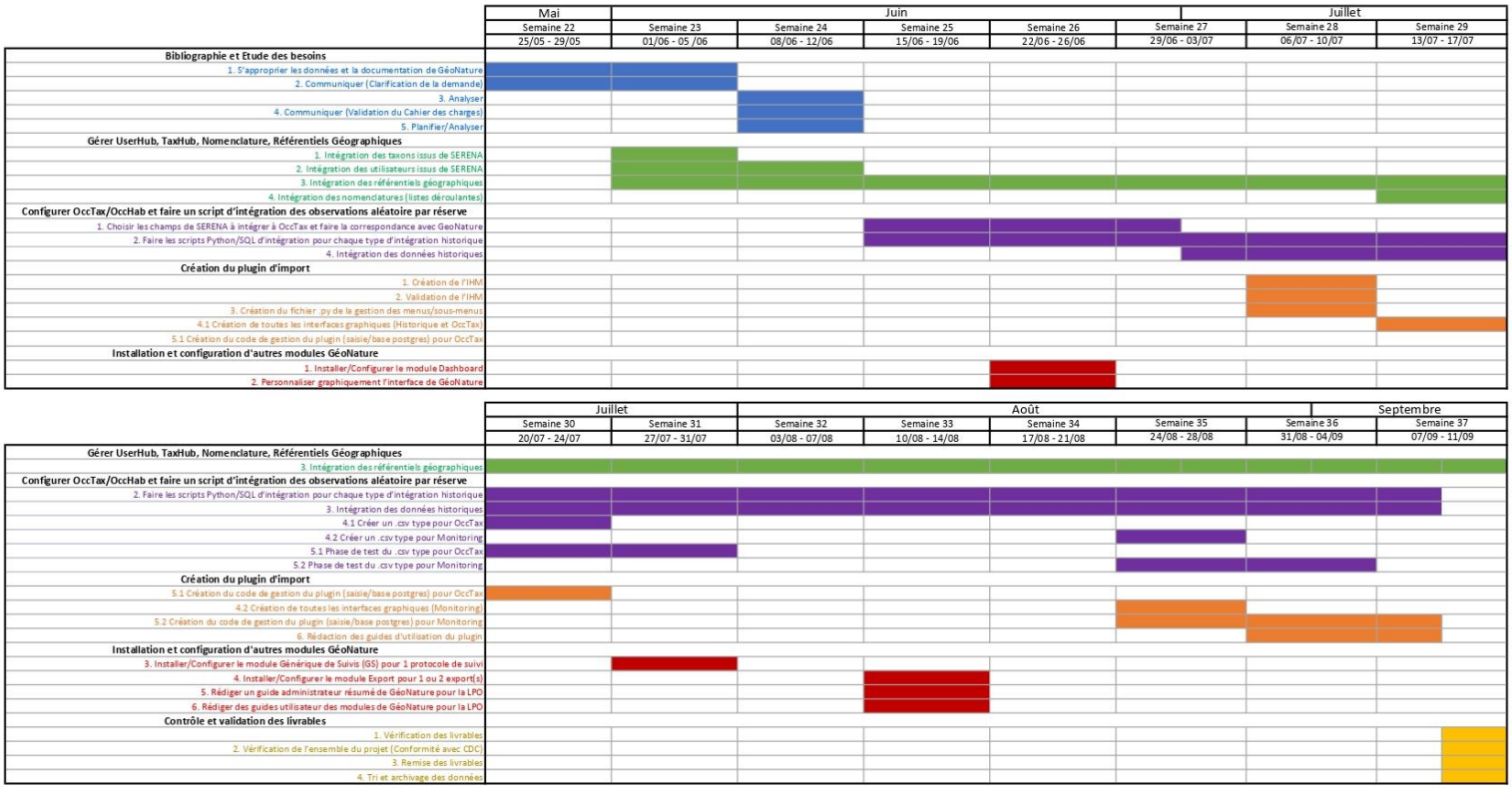


Figure 10: Planning GANTT définitif

c) FICHES DE TRAVAIL

En complément, j'ai organisé mon travail en faisant des fiches récapitulatives de mon travail journalier sur les différents modules et sur le plugin.

Cela permet ainsi d'avoir une traçabilité (temporelle) de mes actions pour moi, mon maitre de stage, ou toute autre personne qui sera sur le projet après mon départ. Ces fiches sont disponibles en Annexe XXV.

III/ CONCEPTION DU STAGE

1. État des connaissances et compétences personnelles

Avant de commencer, j'ai décidé de faire un bilan de mes compétences actuelles et de celles que je devrais approfondir pour mener à bien ce stage :

a) ACTUELLES

Concernant les compétences que j'ai pu acquérir tout au long de l'année au cours de la Licence professionnelle Systèmes d'Information géographique, il y a ma connaissance de l'outil grâce à mon projet tutoré.

En effet, nous avions évoqué avec Monsieur POUGET et Madame MARCHAND, à l'époque, de l'existence de GeoNature. Nous avions décidé avec mon équipe de ne pas poursuivre dans cette voie, malheureusement, car à l'époque l'outil nous semblait inadapté aux actions du LIFE VISON (notre sujet). Cela m'a néanmoins permis d'avoir une première approche de l'outil et de son architecture. Concernant toujours mon projet tutoré, j'ai eu l'occasion de développer mon premier plugin QGIS, de taille conséquente, sous PyQGIS. Ainsi, j'ai désormais de très bonnes bases pour le développement d'un autre plugin que je n'aurais pas pu acquérir avec le nombre assez restreint d'heures de cours sur PyQGIS.

De plus, j'ai pu apprendre à développer en HTML, CSS, JavaScript et SQL, langages de programmation qui me seront utiles pour configurer l'application.

Enfin, j'ai pu manipuler plusieurs types de SGBD dont PostgreSQL et MS Access, tous les deux au cœur de mon sujet de stage.

b) À APPROFONDIR

Plusieurs compétences seront néanmoins à approfondir durant ce stage :

- **GeoNature**, son principe de fonctionnement, ses modules, etc.
- **Champs JSONB** : Inerrant à la base de données PostgreSQL, le champ JSONB permettent de stocker plusieurs champs en JSON dans un seul champ « brute » d'une **table**.

Il reste également la problématique d'Angular et Flask. Si Angular se base sur du langage JavaScript et Flask sur du langage Python, il reste néanmoins impossible d'apprendre le fonctionnement des deux nouveaux langages (inconnus de mon maitre de stage) en autodidacte sur les 4 mois. Il me faudrait au moins 2 mois pour espérer manipuler ces outils.

Figure 11: Logos de Flask et Angular

J'ai donc choisi l'alternative de développer un plugin QGIS où j'avais l'assurance de connaitre déjà la manière de construire le plugin.

2. État de l'existant

Il était également important de faire un état de l'existant autour de GeoNature afin d'avoir une première approche du fonctionnement et des résultats de l'outil.

a) EQUIPE DE DEVELOPPEMENT

Le développement du corps principal de l'application est animé par un collectif de développeurs du réseau des Parcs Nationaux. Les « mainteneurs » de ce collectif sont :

- Gil DELUERMOZ (PnEcrins) : Base de données / SQL / Installation / Mise à jour
- Amandine SAHL (PnCevennes): Backend / Python Flask / API
- Theo LECHEMIA (PnEcrins) : Frontend / Angular 4
- Camille MONCHICOURT (PnEcrins) : Documentation / Gestion du projet

Ces quatre personnes ne manquent pas d'être actives et de répondre à toutes les questions et problématiques des utilisateurs de l'outil via les tickets du dépôt GitHub (ou par mail) : https://github.com/PnX-SI/GeoNature/issues

b) MODULES DE GEONATURE

GeoNature est composé de plusieurs modules. Un module est une application servant à décrire une fonction spécifique au sein de l'application principale. Ils sont le plus souvent reliés les uns aux autres. Voici les modules implémentés de base dans GeoNature :

- **Admin** : Outil d'administration de GeoNature. Permet de gérer les permissions des rôles et les nomenclatures (listes déroulantes).
- **Métadonnées** : Basé sur les standards SINP, permet la création de cadres d'acquisition et de jeux de données réutilisables par les modules de saisie. Permet d'attribuer un cadre ou une « éttiquette » à la donnée que l'on saisie.
- **OccTax :** Basé sur les standards SINP, permet la saisie d'observations aléatoires de taxons.
- **OccHab :** Basé sur les standards SINP, permet la saisie de données d'habitats.
- **Synthèse :** Outil permettant de montrer à l'utilisateur une synthèse des occurrences enregistrées dans différents modules.
- **Validation :** Ce module permet aux utilisateurs qui le peuvent de valider une donnée saisie dans différents modules.
- **TaxHub :** Basé sur le standard SINP, cette application permet de créer des listes de taxons réutilisables dans différents modules.
- **UsersHub :** Cette application permet la création et la gestion des utilisateurs de l'application. Attention cependant, la gestion des permissions se fait dans le module « Admin ».

Voici d'autres modules, non implémentés de base, développés à l'aide des utilisateurs :

- **Dashboard**: A l'initiative d'un sujet de stage, cet outil permet un affichage sous forme numérique et graphique de la synthèse des données.
- **OccTax-mobile** : Outil nomade du module OccTax. Permet la saisie, sur le terrain, de données d'observations aléatoires sur smartphone et tablette (Android seulement).
- **Atlas** : A l'initiative d'un sujet de stage, il s'agit d'une application complémentaire à GeoNature permettant de mettre en valeur les données historiques de la structure sous forme d'atlas consultable du grand public.

Figure 12: GeoNature-atlas du Parc National du Mercantour (Source : http://biodiversite.mercantour-parcnational.fr/)

- **Citizen** : Outil développé à l'initiative de la LPO Auvergne-Rhône-Alpes permettant la saisie participative et simplifiée d'observation naturaliste à destination du grand public.

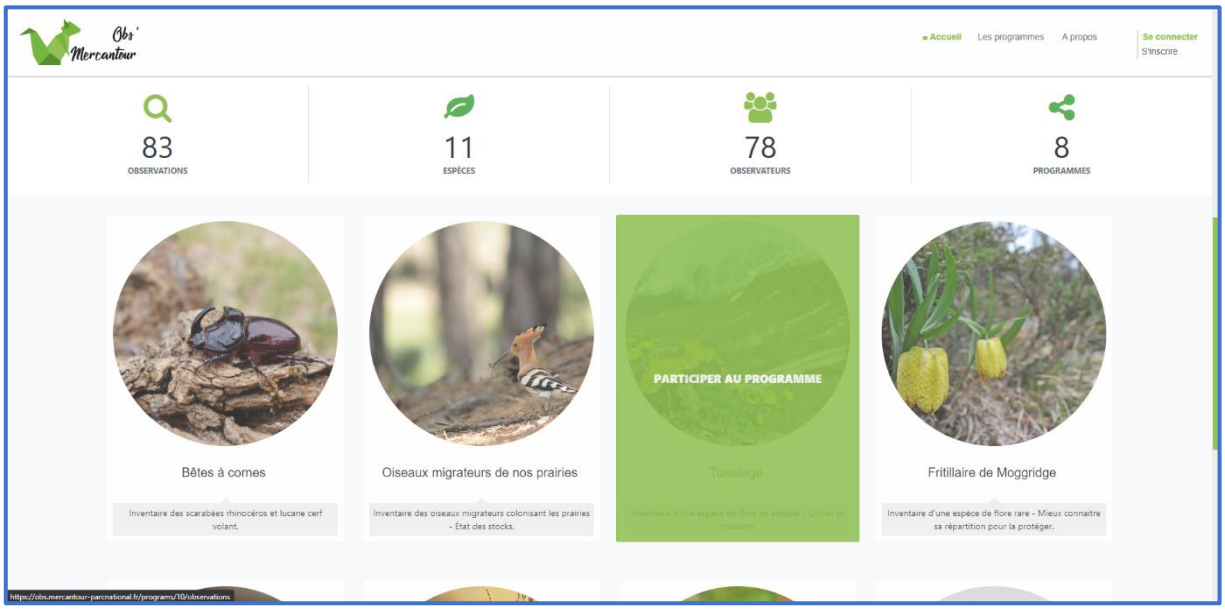


Figure 13: GeoNature-citizen du Parc National du Mercantour (Source : https://obs.mercantour-parcnational.fr/home)

c) MODULES EN COURS DE DEVELOPPEMENT

D'autres modules sont encore en cours de développement cette année, mais permettront d'élargir le champ de protocole de la LPO et qui pourront être intégrés à GeoNature :

- **Monitoring (Module générique de suivis)** : Outil permettant la saisie de protocoles de suivis selon le modèle « 1 site, n visites, n observations ». La première release du module est sortie 1 mois après le début de mon stage.
- **Modules de la SHF** : Dans le cadre de ses missions, la SHF (Société Herpétologique de France) a mis en place plusieurs protocoles scientifiques à l'échelle nationale pour l'étude de différentes espèces de Reptiles et Amphibiens. 5 protocoles font appel à 3 catégories de besoins différents : suivi de sites, occurrence de taxons, suivi d'individus. Les solutions techniques envisagées par la SHF sont :
  - Enrichir le module existant Occtax en permettant de paramétrer l'ajout de champs spécifiques
  - Développer un module CMR (Capture, Marquage, Recapture) qui intègre le suivi de sites et un formulaire de suivi d'individus/opérations
  - Développer un module qui intègre le suivi de sites et l'occurrence de taxons
  - Développer un sous-module de gestion de sites de suivi .

  Plus d'informations : http://lashf.org/wp-content/uploads/2020/01/SHF_CC_Developpement_modules_GeoNature.pdf

d) OUTILS IMPLEMENTE A TRAVERS LA FRANCE

Comme évoqué dans le point *II/1.e.Méthodes et logiciels retenus* concernant les points forts de GeoNature, plusieurs institutions, organismes, associations ont déployé leurs propres GeoNature.

Voici une liste non exhaustive des organismes utilisant GeoNature-atlas et/ou GeoNature-citizen au moment où j'ai fait mon stage :

LPO Auvergne-Rhône-Alpes	- Atlas : https://carto.fauneauvergnerhonealpes.org/presentation - Citizen Valence : http://biodiv-valenceromansagglo.lpo-aura.org/ - Citizen Grenoble : https://www.a-vos-mares.org/participez/ - Citizen Meylan : https://abc-meylan.lpo-aura.org/about
Société Herpétologique de France	- Atlas : https://atlas.lashf.org/
Parc National des Ecrins	- Atlas : https://biodiversite.ecrins-parcnational.fr/presentation
Parc National du Mercantour	- Atlas : http://biodiversite.mercantour-parcnational.fr/presentation - Citizen: https://obs.mercantour-parcnational.fr/home
Parc National de la Vanoise	- Atlas : http://biodiversite.vanoise-parcnational.fr/presentation

Parc National des Cévennes	-	Atlas : https://biodiversite.cevennes-parcnational.fr/
PNR Normandie Maine	-	Atlas : https://observatoire.parc-naturel-normandie-maine.fr/atlas/
CEN des Pays de la Loire	-	Atlas : http://www.biodiv-paysdelaloire.fr/
Atlas du Languedoc-Roussillon	-	Atlas : https://atlas.libellules-et-papillons-lr.org/atlas/
OcNat	-	Atlas : https://biodiv-occitanie.fr/
Picardie Nature	-	Atlas : https://clicnat.fr/
Cistude Nature	-	Atlas : En cours de développement

3. État des utilisateurs et des besoins

Afin de toujours mieux concevoir mon stage avant sa réalisation, je dois me poser les questions suivantes avec la méthode de questionnement « QQOQCCP » :

a) QUAND ?

La question « Quand ? » cherche à définir une temporalité au projet. Pour la durée de celui-ci, elle s'étale sur 16 semaines débutant le 25 mai et se terminant le 11 septembre 2020.

b) QUI ?

La question « Qui ? » permet de définir les personnes prenant parti dans le projet. On peut les distinguer en trois catégories :

- Les administrateurs :

Ce sont les personnes qui géreront et assureront la maintenance de l'outil ainsi que sa configuration auprès des utilisateurs. Ils sont au nombre de deux : Frédéric ROBIN (Chargé de mission) et François BOUCHET (Responsable WEB). J'intégrerai cette équipe tout au long de mon stage.

- Les utilisateurs :

Ce sont les personnes qui manieront l'outil. Elles correspondent aux différents techniciens et conservateurs des 11 réserves et espaces naturels dont Frédéric ROBIN a la charge.

- Les consultants :

Les consultants sont ceux qui portent un intérêt fort au projet sans forcément avoir une utilisation récurrente de l'outil. Ils sont au nombre de quatre :

- Sébastien DALLOYAU (Responsable de projets « Biodiversité », consultant SIG)
- Maxime ZUCCA (Directeur du Pôle Protection de la Nature)
- Ségolène TRAVICHON (Responsable du service Espaces protégés)
- Laurent COUZI (Responsable du service Connaissance).

c) OÙ ?

La question « Où ? » cherche à définir le périmètre d'utilisation de l'outil. Il correspond au périmètre d'actions des techniciens et conservateurs des 11 réserves et espaces naturels de la LPO (voir section *I/2.d.Etendu du projet*), soit 13000 ha répartis sur 21 communes, 3 départements (Charente-Maritime, Vendée et Côtes-d'Armor) et 3 régions (Bretagne, Pays-de-la-Loire et Nouvelle-Aquitaine).

d) QUOI ?

La question « Quoi ? » cherche à donner la finalité du projet. Ainsi, pour ce stage, la finalité est de mettre en place l'outil GeoNature pour le Service Espaces Protégés de la LPO France, et de le configurer pour les différentes données recueillies par les techniciens et conservateurs des réserves et espaces naturels coordonnés par Frédéric ROBIN.

e) COMMENT ?

La question « Comment ? » définit la méthodologie, mais récurent à la partie *II/2.Définition des objectifs du stage*, dont les objectifs et méthodes de travail ainsi que le fonctionnement de l'application.

Ainsi, pour cette deuxième réponse, les utilisateurs devront se connecter via internet au site du GeoNature de la LPO, donner leurs identifiants de connexion et pourront saisir, modifier, supprimer, valider, lire et/ou exporter des données.

f) COMBIEN ?

La question « Combien ? » se rapporte sur les différents moyens mis à disposition du projet : financiers, techniques, humains et matériels. Nous sommes 3 personnes à dédier du temps sur le projet (Frédéric, François et moi-même). Je suis également en relation avec les 11 conservateurs ou gestionnaires des réserves.

Cependant, il est nécessaire de développer davantage les étapes du projet et les choix retenus pour pouvoir répondre à la question du coût (cf. *V/2.Coûts*)

g) POURQUOI ?

Enfin, la question « Pourquoi ? » amène à se demander les raisons de mon stage. Faisant écho à la partie *II/1.b.Analyse des besoins du commanditaire*, les raisons qui ont poussé Frédéric ROBIN à entreprendre cette mission sont multiples :

- Abandonner une base de données naturaliste devenue obsolète et ayant du mal trouver de nouveaux développeurs.
- Rassembler les données éparpillées dans bon nombre de dossiers sur un serveur interne dans une seule et même application.
- Structurer les méthodes et contraindre les utilisateurs à utiliser un même format de création de données.
- Suivre la nouveauté, la gratuité de l'outil et la disponibilité des développeurs.

Une autre facette de la question peut être « Pourquoi GeoNature ? », mais cela fait déjà écho à la partie *II/1.e.Méthodes et logiciels retenus*.

IV/ REALISATION DU STAGE

1. Prétraitement des données

Avant de commencer, je tenais à préciser que les données fournies par la LPO ont préalablement été traitées et nettoyées. Pour des soucis d'efficacité, de rapidité et n'ayant pas les compétences suffisantes pour le faire, Frédéric ROBIN se proposa de traiter les données sous le logiciel de traitement statistique R pour me permettre de me concentrer et d'avancer sur la configuration des modules. Ainsi, je regardais la donnée initiale et indiquais à Frédéric quelles étaient les données à traiter sous R.

Figure 14: Logo du logiciel R

2. Présentation de l'application GeoNature

Pour débuter cette partie détaillée (*IV/Réalisation du Stage*) présentant mon travail durant ces 16 semaines, je vais présenter de façon plus précise l'application GeoNature. Il s'agit d'un outil que je n'avais pas manipulé jusqu'à présent. Il me fallait donc me documenter au maximum sur l'outil afin de comprendre son fonctionnement et ses attentes.

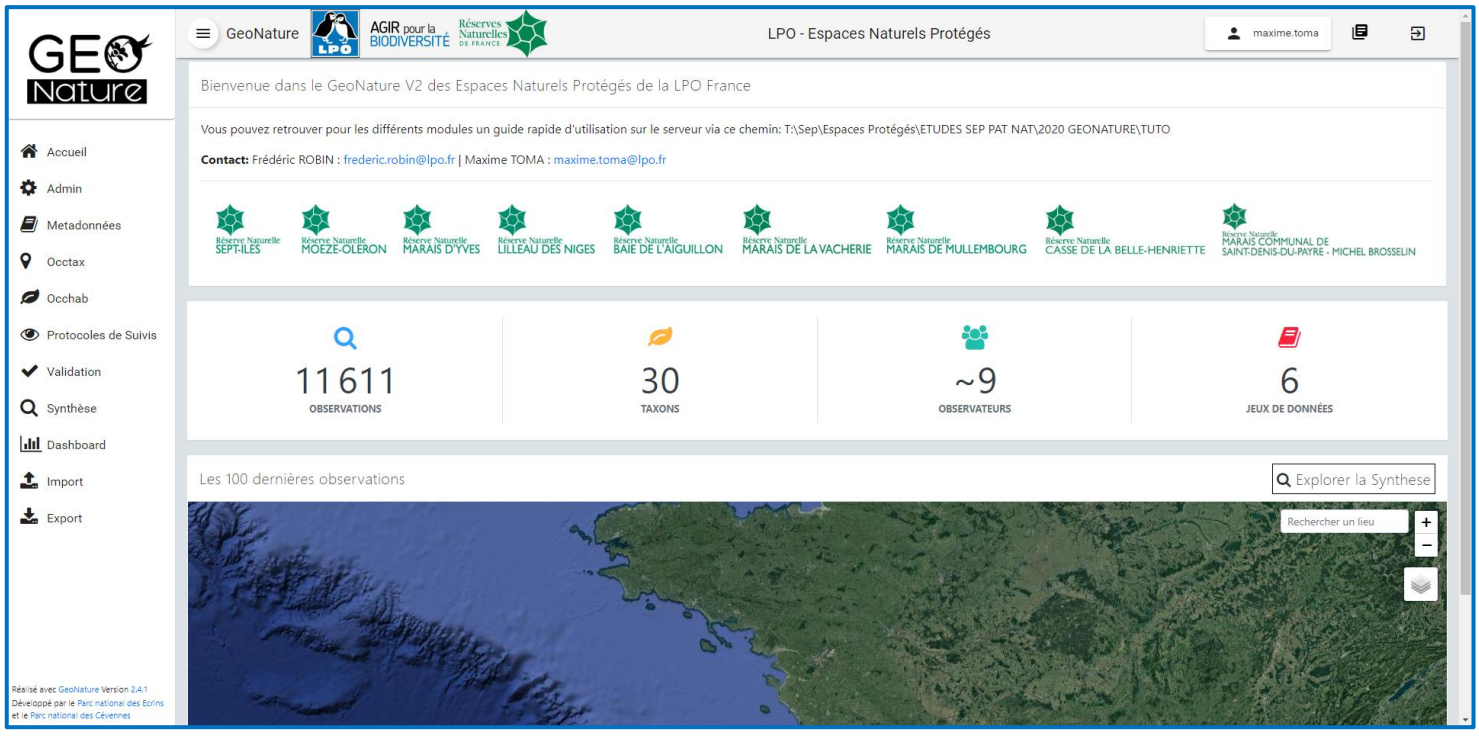


Figure 15: Interface principale de GeoNature (Source : GeoNature du SEP de la LPO)

a) AUTEURS ET CONTRIBUTEURS

L'application est aujourd'hui mise à jour par 5 auteurs : Gil Deluermoz, Theo Lechemia et Camille Monchicourt pour le Parc National des Ecrins, Amandine Sahl et Frédéric Fidon pour le Parc National des Cévennes.

Il existe également quelques contributeurs qui sont à l'œuvre pour aider dans le développement de modules ou pour les développer eux-mêmes. La liste se trouve sur la documentation officielle de GeoNature (Contributeurs).

b) GESTION DE L'OUTIL ET ARCHITECHTURE

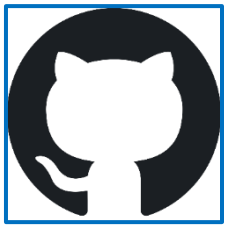


Figure 16: Logo de GitHub

Les différents fichiers et composantes de GeoNature sont disponibles en accès et téléchargement libre sur un dépôt GitHub des Parc Nationaux (https://github.com/PnX-SI/GeoNature). L'installation est détaillée dans la documentation GitHub officielle (Docs GeoNature).

Selon cette dernière, GeoNature possède une architecture modulaire et s'appuie sur plusieurs « services » indépendants pour fonctionner :

- **UsersHub** et son sous-module d'authentification Flask sont utilisés pour gérer le schéma de BDD **utilisateurs** et l'authentification. UsersHub permet une gestion centralisée de ses utilisateurs (listes, organismes, certains droits), utilisable par les différentes applications de son système d'information.
- **TaxHub** est utilisé pour la gestion des taxons au sein de GeoNature, et du schéma **taxonomie** de la BDD, et issus du référentiel TaxRef.
- Un sous-module Flask a été créé pour une gestion centralisée des nomenclatures, il pilote le schéma **ref_nomenclature**.
- **ref_geo** est le schéma de base de données qui gère le référentiel géographique. Il est utilisé pour la gestion des zonages, des communes, du MNT, du calcul automatique d'altitude et des intersections spatiales.

GeoNature a également une séparation claire entre le **backend** (interaction avec la base de données) et le **frontend** (interface utilisateur). Le backend et le frontend se lancent séparément dans GeoNature.

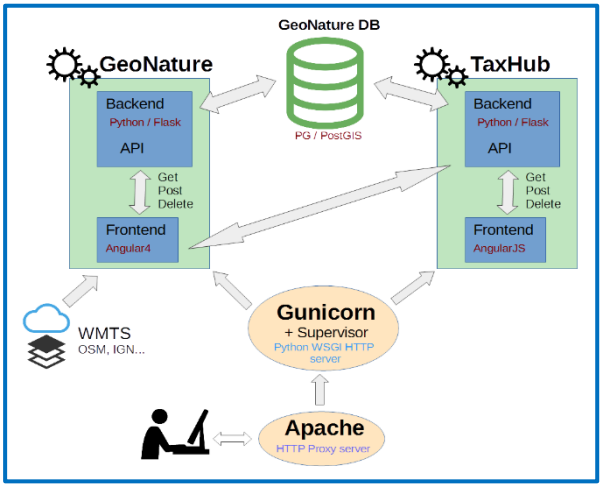


Figure 17: Schéma de l'architecture de GeoNature issu de la documentation officielle (Source : geonature.fr)

c) FONCTIONNEMENT DES MODULES

Toujours selon la documentation officielle, GeoNature a été conçu pour fonctionner en briques modulaires.

Chaque protocole, répondant à une question scientifique, est amené à avoir son propre module GeoNature comportant son modèle de base de données (dans un schéma séparé), son API et son interface utilisateur.

Les modules développés s'appuieront sur le cœur de GeoNature. Ainsi, chacune et chacun ayant les compétences suffisantes peut développer son propre module pour ses propres besoins dans GeoNature (voir la liste des modules de base et ajoutés dans la partie *III/2.b.Modules de GeoNature* et *III/2.c.Modules en cours de développement*).

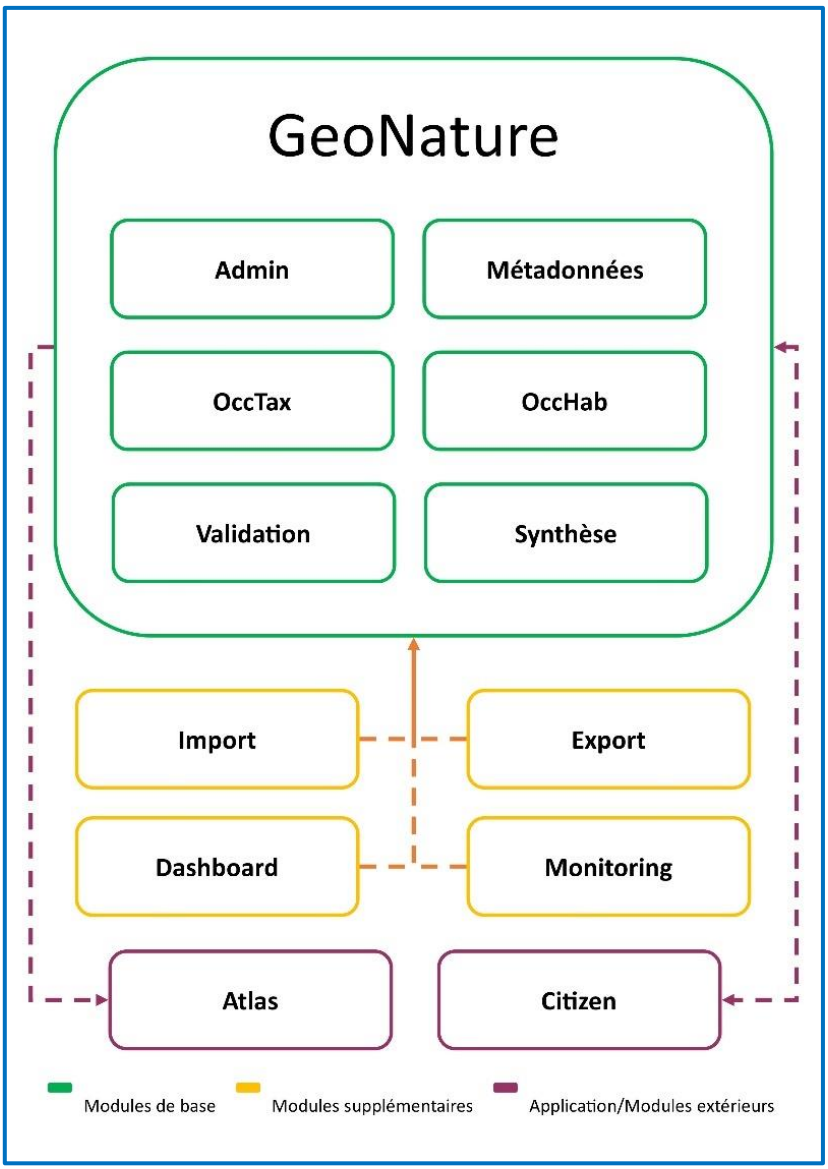


Figure 18: Schéma simplifié du fonctionnement des modules dans GeoNature (Source : Maxime TOMA)

3. Configuration des modules basiques de GeoNature

Dans cette partie, je ne présenterai pas précisément les modules (excepté pour les parties *b.*, *c.* et *d.*), mais les étapes de configuration de ceux-ci. Leurs présentations détaillées sont présentent dans les livrables ou en Annexe XXIII pour le « Guide utilisateur du module Monitoring ».

Pour les modules OccHab, OccTax, Validation, Synthèse, Dashboard, Monitoring, Export, la configuration passe par un fichier présent sur le serveur. Excepté pour Synthèse et Validation, le fichier de configuration « config_gn_module.toml » se trouve dans le dossier « config » de chaque dossier portant le nom des modules.

Pour chaque fichier de configuration existe une version avec l'extension .example répertoriant toutes les configurations possibles pour chacun. Un exemple de ces fichiers pour le module OccTax est disponible en Annexe VI.

a) CONFIGURATION GENERALE DE GEONATURE

Avant de passer à la présentation de la configuration des modules, nous allons voir celle de GeoNature de façon générale. Le fichier de configuration de GeoNature reste assez complet et regroupe même la configuration de 2 modules : « Synthèse » et « Validation ». Cependant pour ces deux derniers, nous verrons en détail cela dans les parties *IV/3.g.Validation* et *IV/3.h.Synthèse*.

Le fichier se situe dans le dossier « config » et porte le nom *config_geonature.toml*. Il est accompagné du fichier *default_config.toml.example* contenant tous les paramètres configurables possibles. Concernant les paramètres, voici une liste non exhaustive de ce que l'on peut retrouver (sans compter les 2 modules précédemment cités) :

- Identifiants de connexion à la base de données
- URLs pour toutes les API
- Le SRID local
- La langue par défaut
- Le nom de l'application
- L'activation de l'interface utilisateur de compte
- Les paramètres géographiques : centre de la carte, niveau de zoom, flux…

Il a été décider avec Frédéric ROBIN de modifier les paramètres suivants : le nom de l'application, l'activation de l'interface utilisateur de compte et les paramètres géographiques.

A noter qu'une partie de cette configuration est modifiable lors de l'installation de l'application. Ce fichier de configuration est disponible en Annexe VII.

b) TAXHUB & USERSHUB : GESTION DES TAXONS ET UTILISATEURS

La première version pour TaxHub est sortie le 9 septembre 2016 et pour UsersHub le 13 octobre 2015, tous deux sur la V1 de GeoNature.

I. TaxHub

Concernant **TaxHub**, sa configuration est peu complexe. En effet, il va gérer principalement deux paramètres définis par l'administrateur : les taxons et listes de taxons utilisés dans GeoNature. L'interface principale est composée de trois onglets : « Taxref », « Taxons » et « Listes ».

- « **Taxref** » regroupe l'ensemble des taxons du référentiel taxonomique du SINP. On peut donc faire une recherche filtrée pour ajouter un ou plusieurs taxons dans les « Taxons » utilisés dans GeoNature. Cela permet de ne pas charger et afficher tout le référentiel taxonomique à l'utilisateur lorsqu'il renseigne un taxon dans un module.
- « **Taxons** » répertorie donc tous les taxons que l'administrateur a enregistré depuis TaxRef. Cependant, ils ne peuvent être utilisés directement dans les modules de GeoNature. Ces derniers requièrent chacun une liste définie de taxons.
- « **Listes** » gère donc toutes les listes définies par l'administrateur. Une liste peut contenir ou non qu'un seul règne, classe, ordre, famille, etc. selon les besoins et modules.

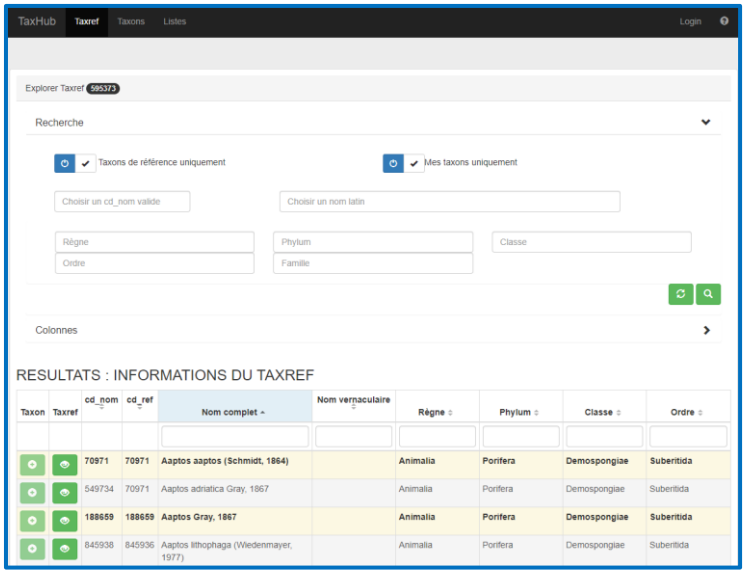


Figure 19: Interface principale de TaxHub (Source : GeoNature du SEP de la LPO)

L'outil a néanmoins un grand inconvénient : l'ajout massif de taxons.

En effet, lors de l'installation de GeoNature, seule une dizaine des taxons sont présents dans la liste principale. La documentation officielle fournit une requête SQL permettant facilement d'intégrer plus d'espèces dans « Mes taxons ». Mais l'idéal reste d'importer les taxons utilisés précédemment depuis les données historiques de la LPO.

Ainsi, après avoir listé les 3795 taxons utilisés dans les 600000 données historiques de la LPO, j'ai pu intégrer ces taxons dans « Mes taxons » ainsi que dans la liste principale (les requêtes SQL sont disponibles en Annexe VIII).

L'utilisation future de cet outil permettra néanmoins d'intégrer, au besoin, les nouveaux taxons publiés dans les prochaines versions de TaxRef.

A noter qu'une autre liste de taxons plus spécifique a été créée pour le sous-module « Comptage Mensuel Commun » (cf. *IV/4.a.Module générique de suivis*).

II. UsersHub

Concernant maintenant **UsersHub**, son interface principale est déjà plus détaillée. Elle est composée de 7 onglets :

- « **Utilisateurs** » regroupe tous les utilisateurs enregistrés. Ils permettent ou non de se connecter et d'utiliser GeoNature. Ce sont également eux qui sont utilisés en tant qu'observateurs, validateurs au sein des modules, mais doivent être renseignés dans des listes à l'instar de TaxHub pour les taxons. Possibilité de modifier les mots de passe, de modifier ou supprimer les utilisateurs.
- « **Organismes** » liste tous les organismes. Ces organismes peuvent être affectés à plusieurs utilisateurs, mais un utilisateur est affecté à un seul organisme.
- « **Groupes** » gère des groupes d'utilisateurs. Ils peuvent ensuite être gérés dans le module Admin pour une gestion commune du CRUVED des membres du groupe.
- « **Listes** » répertorie toutes les listes d'utilisateurs créées. Elles sont réutilisées par quelques modules de GeoNature.
- « **Applications** » permet de définir l'accès des utilisateurs ou des groupes aux applications GeoNature, TaxHub et UsersHub.
- « **Profils** » recueille des profils types pour la gestion d'écriture au sein des applications par exemple Administrateur, Lecteur, Modérateur, Rédacteur ou encore Validateur.
- « **Demande de compte** » permet enfin à une personne lambda de demande d'avoir un compte pour l'utilisation de GeoNature. Nous n'utiliserons cependant pas cette fonctionnalité pour la mise en place du GeoNature du SEP.

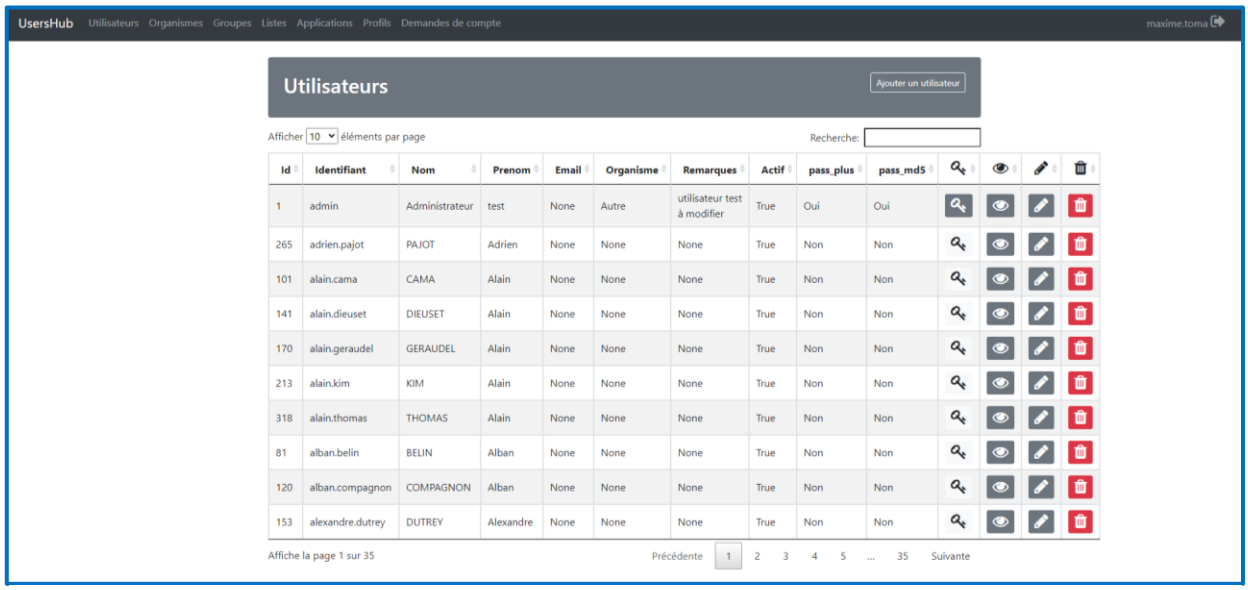


Figure 20: Interface principale de UsersHub (Source : GeoNature du SEP de la LPO)

Ainsi, comme pour TaxHub, j'ai effectué un import massif des plus de 330 utilisateurs issus de SERENA dans GeoNature (les requêtes SQL sont disponibles en Annexe IX).

L'utilisation future de cet outil permettra néanmoins d'intégrer, au besoin, de nouveaux utilisateurs (stagiaires, services civiques, écogardes, salariés, etc.) ou organismes.

A noter qu'une autre liste d'utilisateurs plus spécifique a été créée pour le sous-module « Comptage Mensuel Commun » (cf. *IV/4.a.Module générique de suivis*).

c) ADMIN : GESTION DES PERMISSIONS ET DES NOMMENCLATURES

Le module « Admin » est présent depuis la version 2.0 de GeoNature sortie le 28 février 2019. Le module est composé de deux parties : la gestion des permissions (CRUVED) et la gestion de modules spécifiques.

I. Gestion des permissions

Concernant la gestion des permissions, on doit définir pour chaque utilisateur et/ou chaque groupe ses droits CRUVED vis-à-vis de chaque module. Le terme « CRUVED » désigne les actions que l'utilisateur peut faire dans chaque module : **C**reate-Créer, **R**ead-Lire, **U**pdate-Modifier, **V**alidate-Valider, **E**xport-Exporter, **D**elete-Supprimer.

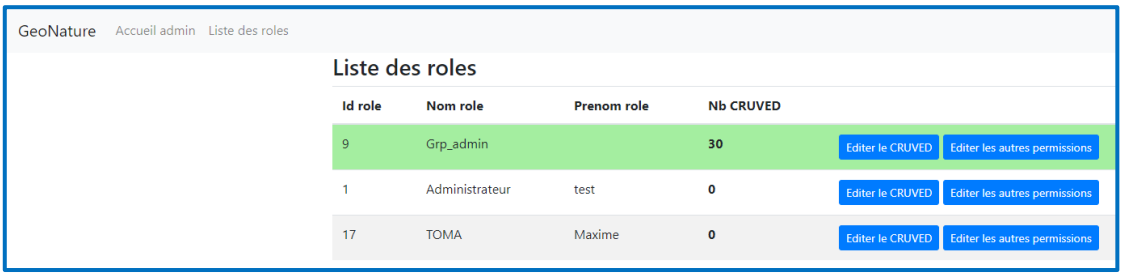


Figure 21: Interface de gestion des permissions, du module Admin (Source : GeoNature du SEP de la LPO)

Les 6 éléments du CRUVED s'accompagnent chacun d'une « portée ». Il s'agit de la portée d'action de l'utilisateur qui se décompose ainsi : « **Aucune donnée** », « **Mes données** », « **Les données de mon organisme** », et enfin « **Toutes les données** ».

Enfin 3 autres types de permissions existent en complémentarité du CRUVED. Il s'agit de permissions liées à l'affichage partiel ou total des données (Sensibilité), à la restriction de la création de données sur une région spécifique (géographique) et aux taxons autorisés (Taxonomique). Cependant, nous ne nous sommes pas concentrés dessus pour l'attribution des droits aux utilisateurs du SEP.

J'ai donc, avec l'appui de Frédéric ROBIN, créé un tableau regroupant tous les utilisateurs actifs, leurs groupes et leurs affectations aux différents types de permissions, disponible en Annexe X.

II. Gestion de modules spécifiques

Pour la gestion de modules spécifiques, on peut définir les paramètres du module « Export », mais sera détaillé dans la partie *IV/4.b.Module exports*, et du sous-module « Nomenclature ». Ce dernier n'apparait pas comme les autres modules et est géré uniquement dans le module « Admin ».

Figure 22: Interface de gestion des nomenclatures et exports du module Admin (Source : GeoNature du SEP de la LPO)

La nomenclature de GeoNature **s'apparente aux listes déroulantes** présentes dans les différents modules.

Le « type de nomenclature » est l'attribut de la liste déroulante et les « items de nomenclature » sont les différentes propositions de celle-ci. Respectant les standards SINP, l'application est déjà très fournie en nomenclature.

J'ai néanmoins dû rajouter certains items non présents dans les listes déroulantes de GeoNature. Ils ont donc été enregistrés comme étant « Non validé » ou « Validation en cours » auprès des standards SINP. Cependant, ils doivent avoir le statut « Active » pour être visibles et utilisables dans les modules utilisant ce champ.

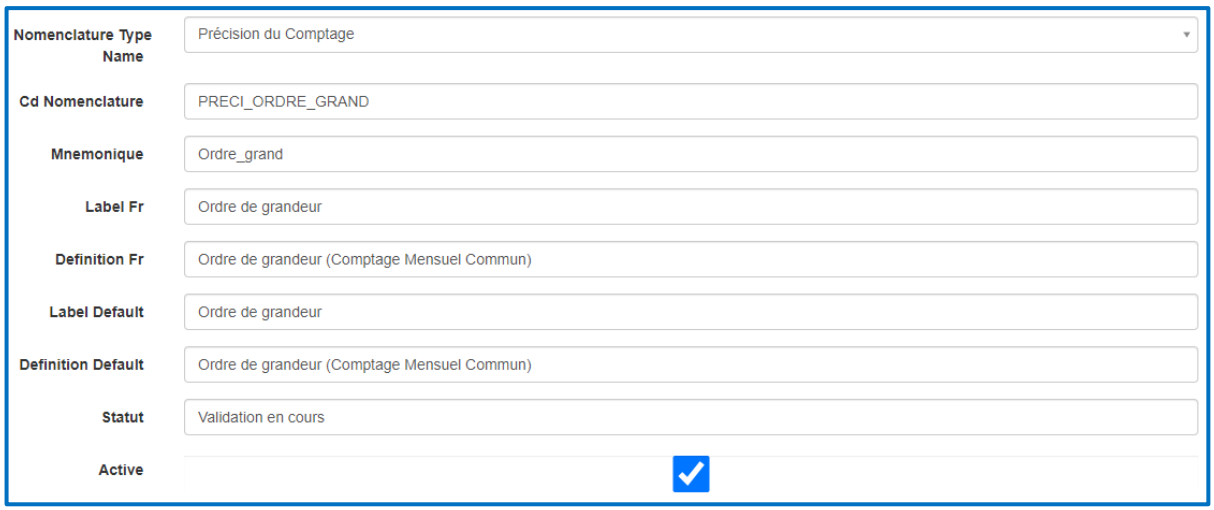


Figure 23: Exemple d'un item ayant le statut "Validation en cours" et étant actif (Module Admin)

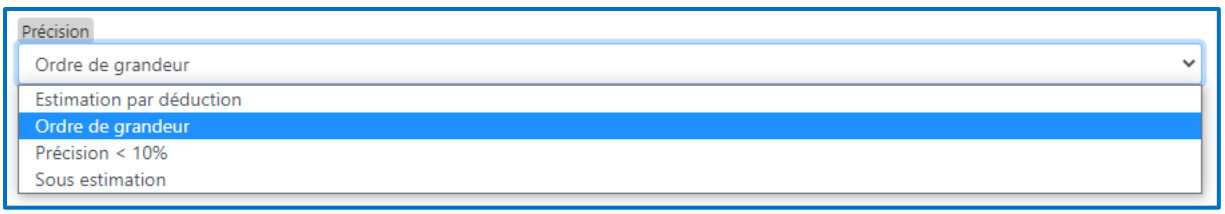


Figure 24: Exemple d'un item ayant le statut "Validation en cours" et étant actif (Module Monitoring)

La liste des nomenclatures rajoutées pour les besoins d'insertion de données historiques, ou pour la création de champs personnalisés se trouve en Annexe XI.

d) METADONNEES : GESTION DES CADRES D'ACQUISITION ET JEUX DE DONNEES

Le module « Métadonnées » est présent depuis la version 2.0 de GeoNature sortie le 28 février 2019.

Ce module va permettre de gérer les différents **cadres d'acquisitions** (CA) accueillant les différents **jeux de données** (JDD). Le cadre d'acquisition peut être vu comme étant une opération, une mission spécifique composée d'un ou plusieurs protocoles de suivis (jeu de données).

Ainsi, j'ai dû créer un à un, via l'interface du module, les cadres d'acquisition et les jeux de données nécessaires à l'accueil des données historiques ainsi qu'aux nouvelles données.

Ils sont au nombre de 3 cadres d'acquisition (Observations aléatoires, Données d'habitat, Comptage Mensuel Commun) et de 26 jeux de données (au moins 1 par réserves et par CA) regroupant 3 modules (OccTax, OccHab et Monitoring).

Une fiche de métadonnée pour le JDD « Comptage Mensuel Commun » de la RNMO est disponible en Annexe XII.

e) OCCHAB : LE MODULE DES RELEVES D'OCCURRENCES D'HABITATS

Le module « OccHab » est sorti le 27 décembre 2019 et est présent depuis la version 2.3 de GeoNature.

OccHab est l'outil de gestion d'occurrences d'habitats de GeoNature faisant suite à la sortie en 2019 du standard « Occurrences d'habitats » du SINP. Les « habitats » sont ainsi répertoriés par « stations » géoréférencées.

La configuration présente 3 réglages différents :

- La possibilité de **saisir les utilisateurs librement ou à partir des listes** du UsersHub
- **Changer la liste d'habitat par défaut** : la création d'une nouvelle liste d'habitat est à effectuer directement en base de données en liant chaque habitat de HabRef à l'id d'une liste. Une alternative à TaxHub pour les habitats n'a pas encore été mise au point.
- **Activation/désactivation de certains champs**

Nous avons fait le choix, avec Frédéric, de ne pas toucher aux réglages par défaut qui lui convenait. Ces derniers sont l'utilisation de la « Liste des observateurs du module Occtax », l'utilisation de la liste par défaut d'habitat et de l'utilisation de tous les champs.

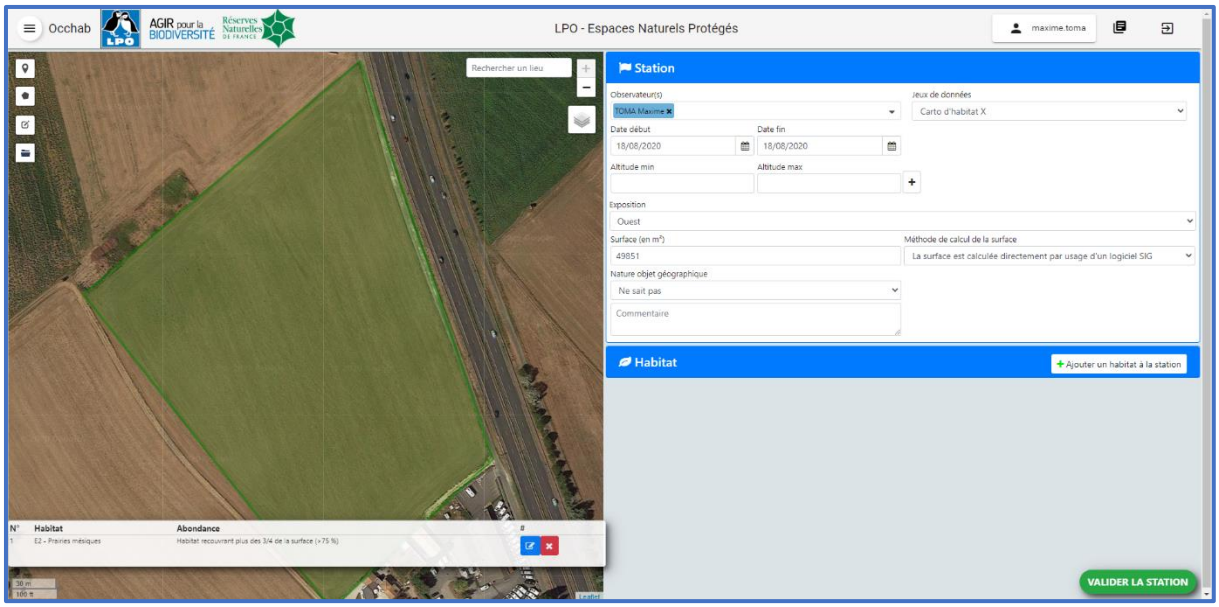


Figure 25: Interface du module OccHab (Source : GeoNature du SEP de la LPO)

f) OCCTAX : LE MODULE DES RELEVES D'OCCURRENCES TAXONOMIQUES

Le module « OccTax » est présent depuis la version 2.0 de GeoNature sortie le 28 février 2019.

OccHab est l'outil de gestion d'observations aléatoires de taxons de GeoNature basé sur le standard « Occurrences de taxons » du SINP. Ces « occurrences de taxons » sont ainsi répertoriés par « relevés » géoréférencés. Ils apparaissent également dans les modules « Synthèse » et « Validation » après création.

La configuration, très complète, présente une dizaine de réglages différents :

- La possibilité de **saisir les utilisateurs librement ou à partir des listes** du UsersHub
- **Changer la liste de taxons** par défaut
- **Activation/désactivation de certains champs**
- Le **nombre de résultats donné pour une recherche de taxon** par autocomplétion
- Mettre la **date d'aujourd'hui par défaut** lors d'une nouvelle saisie
- La valeur de **zoom minimum pour pouvoir saisir une géométrie**
- Les **champs principaux et secondaires** (« map-list ») qui devront apparaitre dans **l'interface principale**, ou accueil du module, indiquant les dernières données saisies.
- **Personnalisation des messages d'erreur**
- Les **paramètres d'export depuis OccTax** : SRID, format de fichier, colonnes sélectionnées, nombre maximum d'occurrences par fichier…

Le choix a été fait avec Frédéric ROBIN de changer un seul réglage par défaut : l'activation du champ « Statut source » qui est par défaut désactivé.

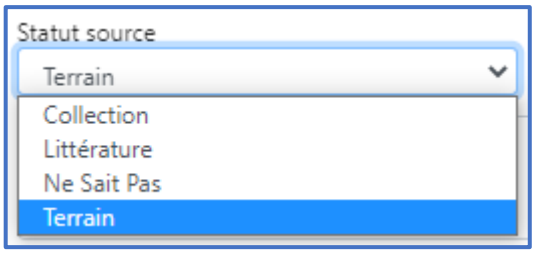


Figure 26: Liste déroulante de l'attribut "Statut source" (Source : GeoNature du SEP de la LPO)

g) VALIDATION : DONNEES A VALIDER

Le module « Validation » est présent depuis la version 2.0 de GeoNature sortie le 28 février 2019.

Il s'agit de l'outil de validation des occurrences taxonomiques. La validation s'effectue à partir des occurrences de la synthèse. Ainsi, seules les occurrences taxonomiques provenant des modules et sous-modules configurés pour s'afficher dans le module « Synthèse » sont éligibles à avoir un statut de validation dans ce module.

Statut	Définition
En attente de validation	Le travail de validation n'a pas encore été réalisé. Le statut de validation est en attente
Inconnu	Le statut de validation n'est pas connu.
Non réalisable	La donnée a été soumise à l'ensemble du processus de validation mais l'opérateur (humain ou machine) n'a pas pu statuer sur le niveau de fiabilité, notamment à cause des points suivants : état des connaissances du taxon insuffisantes, ou informations insuffisantes sur l'observation.
Invalide	La donnée a été infirmée (erreur manifeste/avérée) ou présente un trop bas niveau de fiabilité. Elle est considérée comme trop improbable (aberrante notamment au regard de l'aire de répartition connue, des paramètres biotiques et abiotiques de la niche écologique du taxon, la preuve révèle une erreur de détermination). Elle est considérée comme invalide.
Douteux	La donnée est peu vraisemblable ou surprenante mais on ne dispose pas d'éléments suffisants pour attester d'une erreur manifeste. La donnée est considérée comme douteuse.
Probable	La donnée présente un bon niveau de fiabilité. Elle est vraisemblable et crédible. Il n'y a, a priori, aucune raison de douter de l'exactitude de la donnée mais il n'y a pas d'éléments complémentaires suffisants disponibles ou évalués (notamment la présence d'une preuve ou la possibilité de revenir à la donnée source) permettant d'attribuer un plus haut niveau de certitude.
Certain - très probable	La donnée est exacte. Il n'y a pas de doute notable et significatif quant à l'exactitude de l'observation ou de la détermination du taxon. La validation a été réalisée notamment à partir d'une preuve de l'observation qui confirme la détermination du producteur ou après vérification auprès de l'observateur et/ou du déterminateur.

Figure 27: Statuts de validation des occurrences taxonomiques dans GeoNature (Source : GeoNature du SEP de la LPO)

Ainsi la configuration du module est basée sur celle du module « Synthèse ». Comme précisé au début de cette partie 3, les modules « Synthèse » et « Validation » n'ont pas de fichier de configuration qui sont propre à leurs dossiers homonymes. En effet, la configuration ces derniers, en l'occurrence du module « Synthèse », s'effectue dans le fichier principal de configuration de GeoNature (fichier geonature_config.toml).

h) SYNTHESE : REGROUPEMENT DU TRONC COMMUN DES MODULES

Le module « Synthèse » est présent depuis la version 1.0 de GeoNature sortie le 10 décembre 2014.

Ce module permet de consulter, rechercher et exporter les données provenant des différentes sources et protocoles avec leur tronc commun, basé sur le standard Occurrences de taxon du SINP. Pour des soucis pratiques, la synthèse recueille uniquement les champs répondant aux questions suivantes : Où ? Quand ? Quoi ? Qui a observé ? Dans quel cadre ?

Le module « Import », non installé pour le GeoNature du SEP, permet d'insérer directement des données extérieures dans ce module. Cependant, ces données apparaitront uniquement dans le module « Synthèse » et seront non modifiables.

La configuration, très complète, présente une dizaine de réglages différents :

- La personnalisation des filtres géographiques
- **Changer la liste de taxons** par défaut
- Le **nombre de résultats donné pour une recherche de taxon** par autocomplétion

- Les **champs principaux et secondaires** (« map-list ») qui devront apparaitre dans **l'interface principale**, ou accueil du module, indiquant les dernières données saisies.
- L'activation de la **« clusterisation » des objets** sur la carte
- **Nombre d'observations maximum à afficher sur la carte** après une recherche
- **Nombre maximum des dernières observations affichées** par défaut sur la page d'accueil de la Synthèse
- Les **paramètres d'export** : SRID, format de fichier, colonnes sélectionnées, nombre maximum d'occurrences par fichier

En accord avec Frédéric ROBIN, seul le paramètre des « filtres géographiques » a été modifié afin d'ajouter de nouveaux types de référentiels géographiques.

4. Installation et configuration de modules supplémentaires

a) MODULE MONITORING (MODULE GENERIQUE DE SUIVIS) : GESTION DE PROTOCOLES DE SUIVIS NATURALISTES

Le module générique de suivis ou « Monitoring », codéveloppé entre collaborateurs et le SI des Parcs Nationaux, est sorti le 30 juin 2020 et est présent depuis la version 2.4 de GeoNature.

La grande particularité de ce module est sa personnalisation. Il est possible avec celui-ci de créer des sous-modules personnalisés correspondant, pour chacun, à un protocole de suivis. Répondant à beaucoup d'attentes de Frédéric ROBIN quant à son architecture, ce dernier stock les « observations » selon des « visites » de « sites ».

Cependant, la première version de ce module est sortie un mois après le début de mon stage. Ayant eu un contact dès le début de celui-ci avec Camille MONCHICOURT, un des développeurs de GeoNature, j'ai su que la sortie de ce dernier était imminente et j'ai donc intégré, malgré tout, son installation et configuration dans mon planning prévisionnel (cf. *II/3.b.Diagramme de GANTT*).

Ainsi, la configuration du module passe par 3 étapes (nous prendrons pour exemple le sou-module/protocole « Comptage Mensuel Commun » de la RNMO) :

I. Les champs JSONB

Un nouveau terme fait son apparition avec ce module : **les champs JSONB**. Il s'agit d'un type de champ permettant de stocker du code JSON dans une base de données.

Ainsi, tous les champs additionnels (non-présents dans les tables de base) et spécifiques à chaque protocole se

	id_base_visit [PK] integer	data jsonb
389	144164	{"id_nomenclature_precision_comptage": null}
390	144165	{"id_nomenclature_precision_comptage": null}
391	144166	{"id_nomenclature_precision_comptage": null}
392	144167	{"id_nomenclature_precision_comptage": 558}
393	144168	{"id_nomenclature_precision_comptage": null}
394	144169	{"id_nomenclature_precision_comptage": null}

Figure 28: Exemple du champ "data" en JSONB stockant le champ personnalisé "id_nomenclature_precision_comptage" (Source : pgAdmin4)

retrouvent stockés en JSONB dans la base de données. Ils sont reliés à la table principale par son identifiant (id).

II. Configuration principale du module

Le module présente une configuration générale se répercutant sur tous les sous-modules. Cependant, on peut redéfinir pour chaque sous-module sa configuration afin d'être plus spécifique et/ou différente de la configuration principale.

Dans le fichier custom.json dans le dossier « generic », nous pouvons donc définir 4 paramètres :

- La liste de UsersHub pour l'affiliation des observateurs (observer)
- La liste de UsersHub pour l'affiliation du créateur de la donnée (digitiser)
- La liste de TaxHub pour l'affiliation des taxons
- L'id du jeu de données concerné

Il a été choisi, avec Frédéric ROBIN, de garder la configuration de base pour ce fichier principal.

III. Création d'un sous-module

Pour la création d'un nouveau sous-module, j'aurais besoin de configurer 8 fichiers. Les modifications apportées dans chacun d'entre eux se répercuteront instantanément sur GeoNature (à l'inverse de tous les autres modules). Une image peut également être définie pour illustrer le sous-module sur l'interface principal de « Monitoring », mais une « réactualisation du frontend » devra être fait via une invite de commande.

- custom.json : On y redéfinit les paramètres vus précédemment si besoin
- config.json : il s'agit de la configuration générale du sous-module. On y définit l'arborescence de ce dernier (les niveaux sites, visites, observations ou média peuvent ne pas être utiles pour certains protocoles).
- module.json : On y définit le nom et la description qui s'affichera sur l'interface principale du module « Monitoring ».
- site.json : Il s'agit de la configuration du niveau « Site ». On y définit les champs à afficher pour la consultation et les champs du formulaire de ce niveau.
- visit.json : Il s'agit de la configuration du niveau « Visite ». On y définit les champs à afficher pour la consultation et les champs du formulaire de ce niveau.
- observation.json : Il s'agit de la configuration du niveau « Observation ». On y définit les champs à afficher pour la consultation et les champs du formulaire de ce niveau.
- nomenclature.json : Ce fichier prendra en charge les nomenclatures spécifiques au sous-module et les intégrera dans le schéma « ref_nomenclatures ».
- synthese.sql : Ce fichier va configurer le module « Synthèse » afin d'y afficher les données du sous-module si l'on souhaite faire afficher ces derniers dans le module Synthèse.

La rédaction (sauf pour synthese.sql) se fait en JSON. La documentation officielle reste, cependant, accessible aux non initiés à ce langage, mais difficilement aux néophytes.

Les fichiers de configuration du sous-module « Comptage Mensuel Commun » sont disponibles en Annexe XIII.

b) MODULE EXPORTS : UTILITAIRE POUR LA GESTION D'EXPORTS PERSONNALISES

Le module « Exports », codéveloppé entre collaborateurs et le SI des Parc Nationaux, est sorti le 21 février 2020 et est présent depuis la version 2.3.2 de GeoNature.

Il permet aux utilisateurs de télécharger des données, depuis un mail qui leur est envoyé, selon un export personnalisé défini par un administrateur. Le procédé s'appuie sur les « vues » de PostgreSQL. Par défaut, l'export peut se faire au format JSON, GeoJSON, Shape, CSV et GeoPackage.

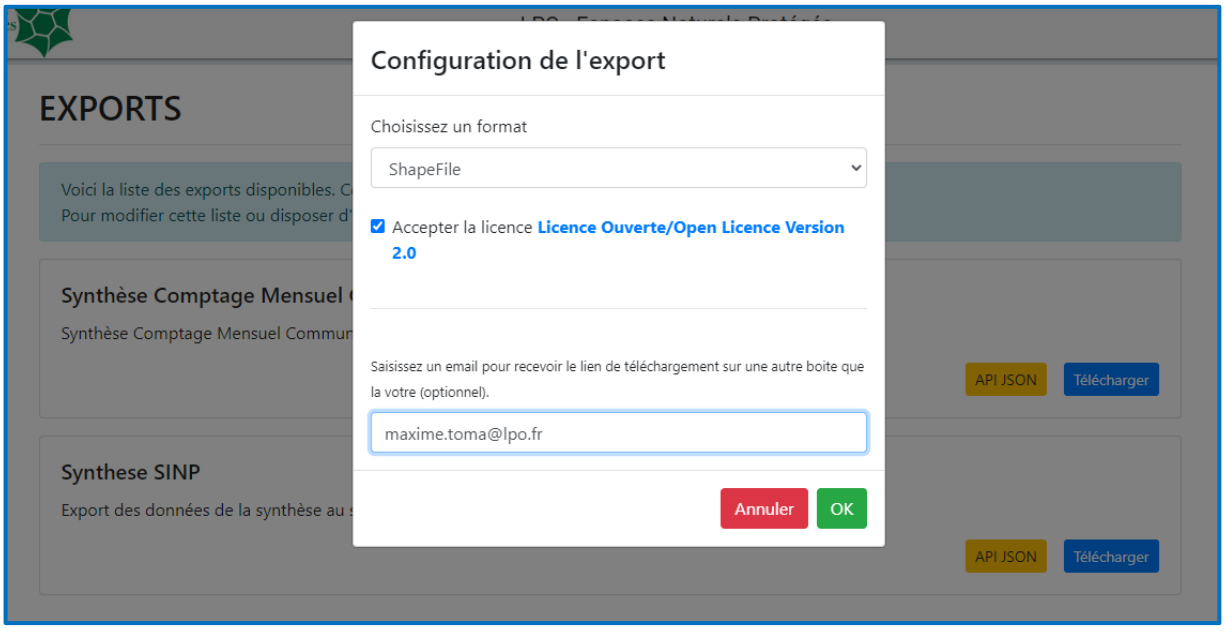


Figure 29: Interface du module "Export" lors de la création d'un export (Source : GeoNature du SEP de la LPO)

La configuration du module se fait en trois étapes :

- **Configuration du mail** : Le module demande de configurer le fichier principal de GeoNature afin d'y indiquer les identifiants de connexion, en Flask, d'une adresse mail qui fera office d'« expéditeur des exports » (cf. la partie « [MAIL] » du fichier principal de configuration de GeoNature en Annexe VII).
- **Configuration du module** : Le module a également son propre fichier de configuration avec ses propres paramètres comme l'indication des dossiers du serveur où seront stockés chaque export (automatiques ou manuels) et leur durée de stockage.
- **Configuration du ou des export(s)** : Enfin, il nous faut configurer les exports en eux-mêmes.
    - **Côté BDD** : Nous devons tout d'abord créer une vue répertoriant tous les champs dont on souhaite pour notre fichier d'export. Un exemple de requête de création d'une vue d'export pour le sous-module « Comptage Mensuel Commun » est disponible en Annexe XIV.
    - **Côté module « Admin »** : Comme indiqué dans la partie *IV/3.c.II.Gestion de modules spécifiques*, la gestion des exports passe également du côté du module « Admin ».

On va ainsi définir pour chaque export son nom, le nom du schéma et le nom de la vue, une description, le nom du champ géométrique, le SRID du champ géométrique, la privatisation ou non de l'export, et la licence ouverte utilisée (possibilité d'en créer une personnelle). Le module permet également de gérer la planification des exports ainsi que les personnes ayant accès à ce dernier s'il n'est pas public.

Nous nous sommes mis en accord avec Frédéric ROBIN pour les différents réglages du module.

c) MODULE DASHBOARD : SYNTHESE GRAPHIQUE DES DONNEES

A l'initiative d'un sujet de stage, le module « Dashboard », développé par Elsa Guilley (stagiaire au Parc national des Ecrins en 2019), est sorti le 20 février 2020 et est présent depuis la version 2.3.1 de GeoNature.

Le module permet d'afficher sous forme graphique différente statistiques, par année, entité géographique, rang taxonomique, cadre d'acquisition, taxons recontactés/non-recontactés/nouveaux par année, sur les données du module « Synthèse ».

La configuration présente 3 réglages différents :

- **Paramétrage du niveau de simplification des zonages** sur la carte "Synthèse par entité géographique"
- **Paramétrage des zonages affichables** sur la carte "Synthèse par entité géographique"
- **Paramétrage de l'échelle de graduation** des entités par classes sur la carte "Synthèse par entité géographique"

A l'instar des modules « Synthèse » et « Validation », nous avons décidé avec Frédéric de rajouter plus de filtres géographiques.

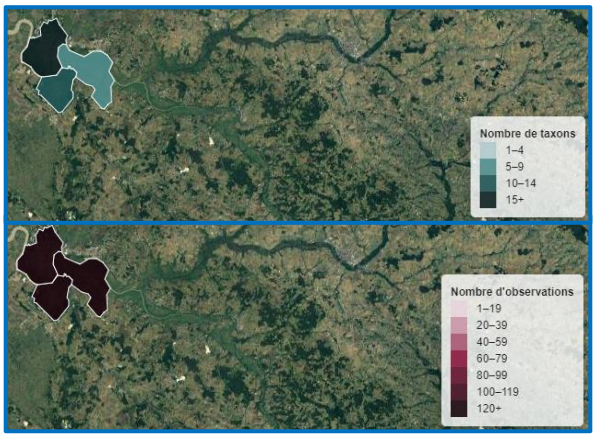


Figure 30: Extrait de la carte "Synthèse par entité géographique" montrant les 2 types de classe (Source : GeoNature du SEP de la LPO)

5. Insertion des données historiques

a) PRESENTATION RAPIDE DE LA BASE DE DONNEES

Avant de développer les méthodes appliquées pour l'insertion des données, je me dois d'introduire le modèle de la base de données de GeoNature (MCD disponible en Annexe XV). La base de données du GeoNature du SEP est aujourd'hui composée d'un schéma (lpo_imports) créé pour les besoins du plugin QGIS que l'on développera prochainement, d'un schéma regroupant toutes les données historiques intégrées (histo_imports), et de 16 autres schémas (sans compter le schéma par défaut « public ») :

- gn_commons : Contient les tables regroupant les historiques de connexion, différents paramètres pour le fonctionnement de l'application, les données de validation, les données sur les modules, un stockage des médias.
- gn_dashboard : Regroupe les tables de gestion du module Dashboard.

- gn_exports : Regroupe les tables et les vues de gestion du module Exports.
- gn_imports : Regroupe les tables de gestion du module Imports.
- gn_meta : Regroupe les données relatives aux jeux de données, et cadres d'acquisition.
- gn_monitoring : Regroupe les données de sites, visites et observations du module « Monitoring ».
- gn_permissions : Regroupe les tables de gestion du CRUVED et des autres types de permissions.
- gn_sensitivity : Regroupe les données sensibles ainsi que les critères les justifiant.
- gn_synthese : Regroupe les données présentes dans le module « Synthèse ».
- pr_occtax : Regroupe les données relatives aux relevés et occurrences de relevés du module « OccTax ».
- pr_occhab : Regroupe les données relatives aux stations et habitats du module « OccHab ».
- ref_geo : Regroupe les différents zonages, MNT, et référentiels géographiques utilisés par GeoNature.
- ref_habitats : Regroupe le référentiel d'habitats HabRef et ses tables de gestion.
- ref_nomenclatures : Regroupe toutes les nomenclatures et typologies présentes dans GeoNature.
- taxonomie : Regroupe le référentiel taxonomique TaxRef et ses tables de gestion
- utilisateurs : Regroupe la table des utilisateurs du UsersHub et ses tables de gestion

La base est également gérée par de nombreux « triggers » et fonctions permettant de contraindre les données et d'éviter les erreurs d'import disponibles ici : http://docs.geonature.fr/admin-manual.html#fonctions et http://docs.geonature.fr/admin-manual.html#triggers-vers-la-synthese.

b) L'INSERTION DES REFERENTIELS (TAXONS, UTILISATEURS, GEOGRAPHIQUES, NOMMENCLATURES)

J'ai donc commencé par intégrer tous les référentiels, présents dans l'ancienne base SERENA, dont j'aurai besoin pour l'intégration des données historiques. SERENA permet l'export des utilisateurs et des sites géographiques, ce qui n'est pas le cas pour les taxons et nomenclatures. Commençons par l'insertion des taxons.

I. Insertion des taxons

Pour commencer, j'ai dû sélectionner et distinguer tous les « *cd_nom* » des 600000 données que compose la base. Il s'agit d'un numéro unique pour identifier toutes les espèces référencées dans TaxRef. Frédéric ROBIN avait fait un export sous un fichier Excel. N'ayant pas pu avoir accès au modèle de donnée de la base Access, j'ai préféré d'utiliser le fichier Excel pour cette manipulation.

J'ai commencé par isoler la colonne des taxons « TAXO_MNHN_ID » dans un fichier CSV. J'ai ensuite créé deux tables dans le schéma taxonomie : *temp_tax* et *temp_tax_groupby*.

J'ai importé le CSV dans la première table, puis j'ai effectué un « GROUP BY cd_nom » pour l'insérer dans la seconde table. A partir de là, j'ai pu sélectionner les éléments de TaxRef en

fonction des taxons issus de SERENA et je les ai inséré dans la table *bib_noms* (équivalent de « Mes taxons » dans TaxHub) puis dans la table *cor_nom_liste* en faisant la correspondance avec la liste 100 (liste par défaut de GeoNature). Les plus de 3660 taxons ont ainsi été insérés.

Les requêtes sont disponibles en Annexe VIII.

II. Insertion des utilisateurs

Passons maintenant à l'insertion des utilisateurs. Il me fallait, en revanche, faire très attention, car Frédéric ROBIN m'a indiqué que certains utilisateurs avaient été rentrés en double et que d'autres n'était pas utilisés.

J'ai donc commencé par voir quels utilisateurs étaient présents parmi les 600000 données des réserves. Pour ce faire, j'ai exécuté une requête sous Access dans la base mère appelée « *lposep.mdb* », pour récupérer l'id des utilisateurs ayant une donnée associée puis j'ai exporté le résultat sous un fichier Excel.

J'ai ensuite supprimé à la main les doublons et, sous la directive de Frédéric ROBIN, les utilisateurs corrompus (ayant des numéros à la place des noms et prénoms) puis j'ai exporté le tout sous un fichier CSV que j'ai envoyé dans une table que j'ai créé dans la base de données.

Il ne me reste plus maintenant qu'à créer un organisme et groupe spécial et de les relier à ces utilisateurs, puis à insérer ces derniers dans le UsersHub.

Les requêtes sont disponibles en Annexe IX.

III. Insertion des nomenclatures

Pour l'insertion des nomenclatures, j'ai dû faire une correspondance à la main des items déjà présents dans la base et de ceux renseignés dans les données de SERENA. Je n'ai malheureusement fait le travail que sur les données d'observations aléatoires et sur les données du protocole « Comptage Mensuel Commun », faute de temps.

Ainsi, j'ai rentré une par une les quelques nomenclatures absentes via le module « Admin » (cf. *IV/3.c.II.Gestion_de_modules_spécifiques*). La liste des nomenclatures insérées dans GeoNature est disponible en Annexe XI.

IV. Insertion des référentiels géographiques

Enfin, pour l'insertion des référentiels géographiques, j'ai dans un premier temps inséré les données de l'INPN (Inventaire national du patrimoine naturel). Ces derniers proposent en effet sur leur site de télécharger les zonages naturels récents comme les ZNIEFF, Cœur de Parc Nationaux, Réserves Naturelles, etc...

J'ai ainsi téléchargé les données suivantes, puis les ai ajoutés dans la base de GeoNature avec le plugin DBManager de QGIS :

- Aire d'adhésion de Parc Nationaux
- Cœur de Parc Nationaux
- Parc Naturels Régionaux et Marins
- Réserves Naturelles Régionales et Nationales
- Zone Natura 2000 (ZICO, ZPS, ZSC/SIC, ZNIEFF)
- Sites acquis des Conservatoires d'espaces naturels

Je les ai ensuite ajoutés dans la table *ref_geo* via des requêtes SQL.

La LPO présente également des zonages propres à leurs protocoles de suivis et autres observations aléatoires. Ainsi, ils se distinguent en trois, voire quatre, niveaux du plus grand au plus petit : « Lieu-dit », « Unité de gestion », « Site d'inventaire » et « Points d'échantillonnage » (STEPRO uniquement). Ainsi, j'ai dû créer ces types dans la table *bib_areas_types* puis j'ai pu intégrer ces zonages en les reliant à ces nouveaux types.

Les requêtes sont disponibles en Annexe XVI.

c) L'INSERTION DES DONNEES D'HABITATS (OCCHAB)

J'ai pu ainsi continuer sur l'insertion, cette fois-ci, des habitats (ou cartographie d'habitat) dans le module OccHab. Prenons pour exemple l'insertion des données de la cartographie d'habitat de la Réserve Naturelle Nationale de Lilleau des Niges de 2012 et 2018.

Je fus assez rapidement confronté à un problème : comment insérer dans trois tables différentes, et reliées les unes aux autres par leurs identifiants, la donnée issue d'une seule table ? En effet, le schéma *pr_occhab* de la base de données se compose de trois tables principales : ***t_stations***, ***t_habitats*** et ***cor_station_observer*** (MCD simplifié ci-dessous).

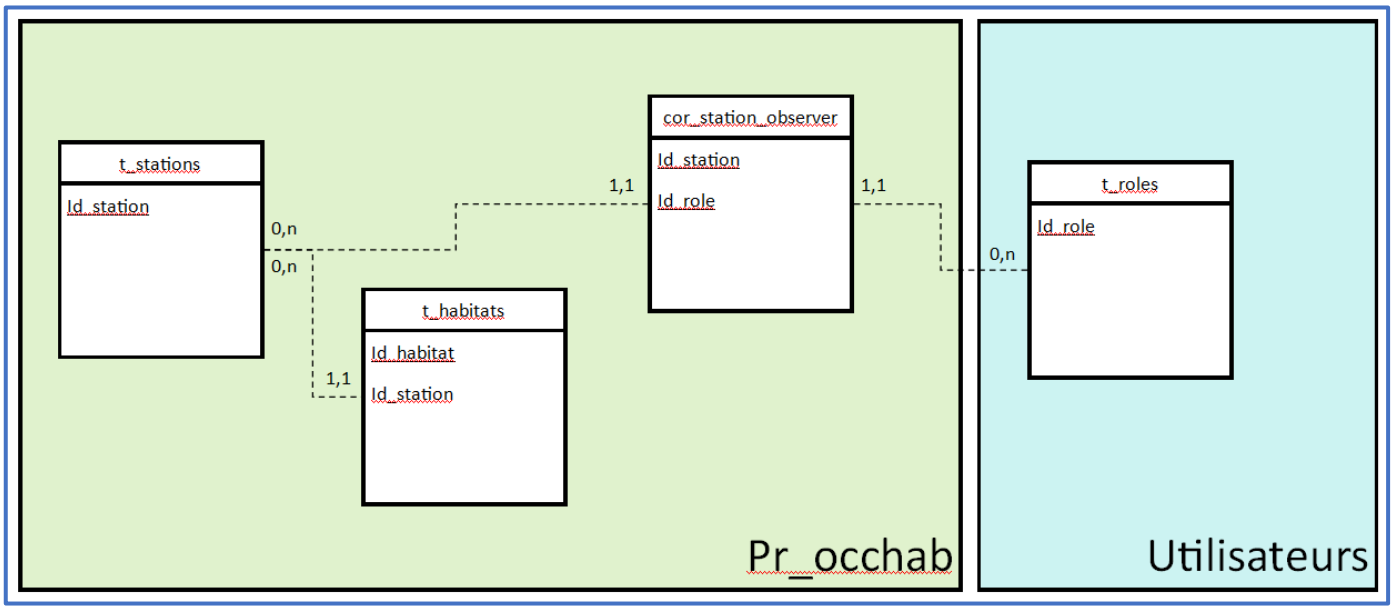


Figure 31: MCD simplifié du schéma de donnée du protocole OccHab (Source : Maxime TOMA)

- *T_stations* regroupe les stations d'habitats
- *T_habitats* regroupe ainsi les occurrences d'habitats reliés à une station

- *Cor_station_observer* fait le lien entre la table des utilisateurs (UsersHub) et une station.

Mais comment insérer ces données ? La problématique est la suivante :

J'ai une donnée initiale qui porte les stations et habitats mélangés. De ce fait, les occurrences de stations sont dupliquées si une station porte plusieurs habitats (exemple simplifié ci-dessous).

Station 1 : Les pins	Habitat 1 : Prairie mésophile
Station 1 : Les pins	Habitat 2 : Prairie hygro mésophile
Station 1 : Les pins	Habitat 3 : Jardin
Station 2 : Les hêtres	Habitat 1 : Pelouse sèche

Après avoir inséré les stations, **comment faire le lien pour faire correspondre les bonnes données d'habitats aux bonnes stations ?**

Je suis d'abord parti sur la géométrie. Elle est identique entre ma donnée initiale et celle insérée dans la table t_stations. Je n'avais plus qu'à récupérer l'id_station correspondant à la géométrie de ma donnée initiale. Cependant, ce n'est pas vraiment une méthode 100% viable et comporte des risques d'erreurs si la géométrie a été modifiée par l'application lors de l'insertion des stations, par exemple.

J'ai donc cherché à faire une relation avec l'identifiant unique (méthode plus sure). Cependant, **comment récupérer l'identifiant d'une donnée insérée ?** M'aidant des forums internet, je suis rapidement arrivé à la commande SQL « RETURNING ». Elle s'utilise à la toute fin de la requête utilisant un « INSERT INTO » et demande la colonne à retourner après l'insertion. Une partie du problème s'éclaircit, mais cela soulève une nouvelle question : comment garder ces valeurs retournées et faire un lien avec l'identifiant de la table d'origine ?

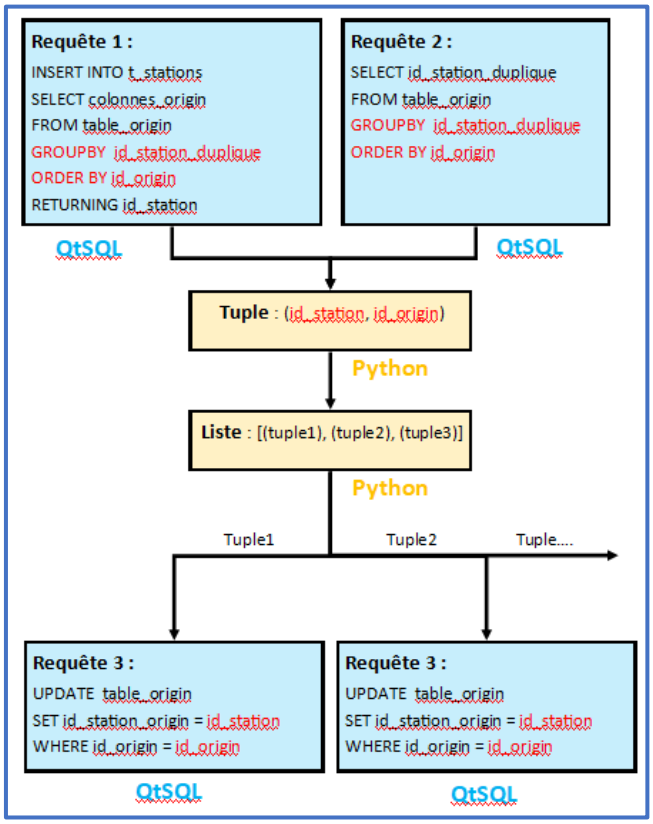


Figure 32: Schéma simplifié de la manœuvre adoptée pour l'insertion des données d'habitats dans OccHab (Source : Maxime TOMA)

Cette fois-ci, **j'ai trouvé une réponse du côté du langage Python** et plus particulièrement d'un outil utilisé pendant mon projet tutoré : QtSQL. QtSQL est un module qui permet d'exécuter des requêtes SQL sous Python. Ainsi, je vais pouvoir récupérer via cette méthode les valeurs d'une première requête (INSERT INTO avec RETURNING) en même temps que les valeurs d'une seconde requête (SELECT les ids de la table d'origine), étant triés selon les mêmes colonnes, tout en regroupant les résultats sous forme d'une liste de tuples.

Il ne me restait plus qu'à créer une nouvelle colonne dans ma table d'origine et à faire une requête UPDATE pour chaque tuple de la liste. Cette solution viable reste néanmoins assez dangereuse si l'on ne vérifie pas correctement les colonnes de tri entre les deux requêtes.

J'ai pu enfin insérer les utilisateurs et les habitats dans leurs tables respectives en faisant la relation avec les id_station enregistrés dans la nouvelle colonne de la table d'origine.

Le code python de l'insertion des données historiques de la RNMO est disponible en Annexe XVIII.

d) L'INSERTION DES DONNEES D'OBSERVATIONS ALEATOIRES (OCCTAX)

L'insertion des données dans OccTax suit le même procédé, donc, que pour OccHab. En revanche, le nombre de tables s'agrandit. Voici les tables où devront être réparties les données de la table d'origine :

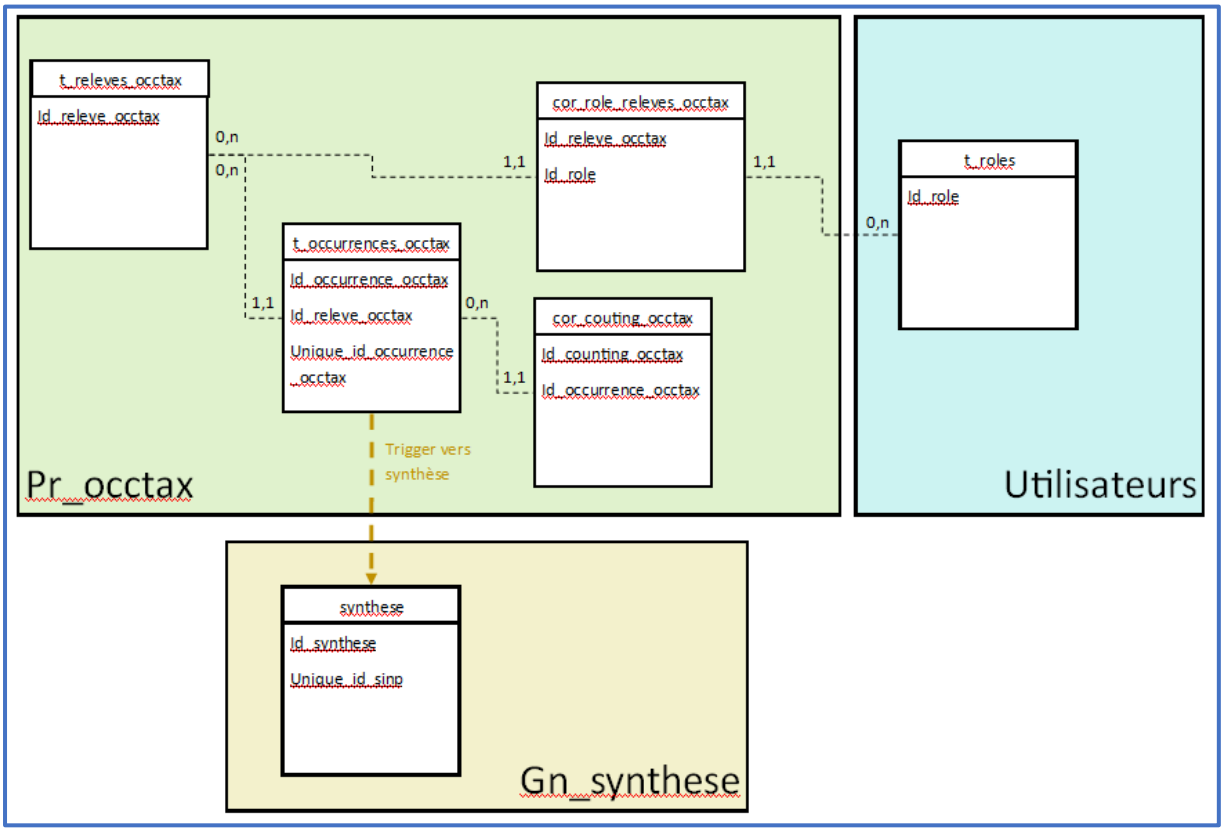


Figure 33: MCD simplifié du schéma de donnée du protocole OccTax (Source : Maxime TOMA)

- *T_releves_occtax* regroupe les différents relevés
- *T_occurrences_occtax* regroupe les occurrences de taxons appartenant à un relevé
- *Cor_counting_occtax* s'agit du dénombrement des occurrences de taxons
- *Cor_role_releves_occtax* fait le lien entre la table des utilisateurs (UsersHub) et un relevé.

J'ai donc procédé comme ceci pour l'insertion des données historiques dans OccTax :

1. **Insertion des relevés à partir de la table d'origine**. Regroupement des éléments dupliqués et tri par l'id dupliqué de façon croissante.
2. **Récupération de *l'id_releve_occtax*** correspondant à chaque insertion et, en même temps, récupération des ids dupliqué (seuls) ordonnés de façon croissante. Insertion de chaque duo d'id dans une liste de tuples.
3. **Création d'une colonne** qui accueillera l'insertion des *id_releve_occtax* du tuple en faisant la correspondance avec son id dupliqué.
4. **Insertion de la correspondance entre les rôles** (utilisateurs/observateurs) **et *l'id_releve_occtax*** issu de la nouvelle colonne.
5. **Insertion des occurrences de taxons** à partir de la table d'origine. Regroupement des éléments dupliqués et tri par l'id de la table d'origine de façon croissante.
6. **Récupération de *l'id_occurrence_occtax*** correspondant à chaque insertion et, en même temps, récupération des ids de la table d'origine (seuls) ordonnées de façon croissante. Insertion de chaque duo d'id dans une deuxième liste de tuples.
7. **Création d'une deuxième colonne** qui accueillera l'insertion des *id_occurrence_occtax* du tuple en faisant la correspondance avec son id.
8. **Insertion des éléments du dénombrement** dans la table *cor_counting_occtax* selon *l'id_occurrence_occtax* de la nouvelle colonne. Un trigger fait automatiquement une insertion à ce moment-là des éléments des 3 tables dans la table *synthese*.
9. **Insertion des éléments de validation** dans la table *t_validations* selon l'identifiant unique de la table *t_occurrences_occtax*. Cet identifiant unique est de type UUID et est l'attribut de liaison entre le module de synthèse (et donc le module de validation) et l'occurrence à valider.

Le code python de l'insertion des données historiques de la STEPRO est disponible en Annexe XVII.

e) L'INSERTION DES DONNEES DE PROTOCOLE DE SUIVIS (MONITORING)

Enfin, l'insertion des données pour les protocoles de suivis fait appel à un nouveau type d'attribut : le JSONB (cf. *IV/4.a.I.Les champs JSONB*).

En effet, le module Monitoring propose à ses utilisateurs de créer ses propres champs personnalisés selon le protocole créé. Ses derniers sont donc stockés sous le langage JSON, lui-même stocké dans la base donnée dans une colonne au format JSONB.
Ces protocoles sont donc répartis en sous-modules. Un sous-module est égal à un protocole et un jeu de donnée. Pour mieux comprendre, voici ci-dessous un extrait du guide d'utilisation, que j'ai créé, du module :

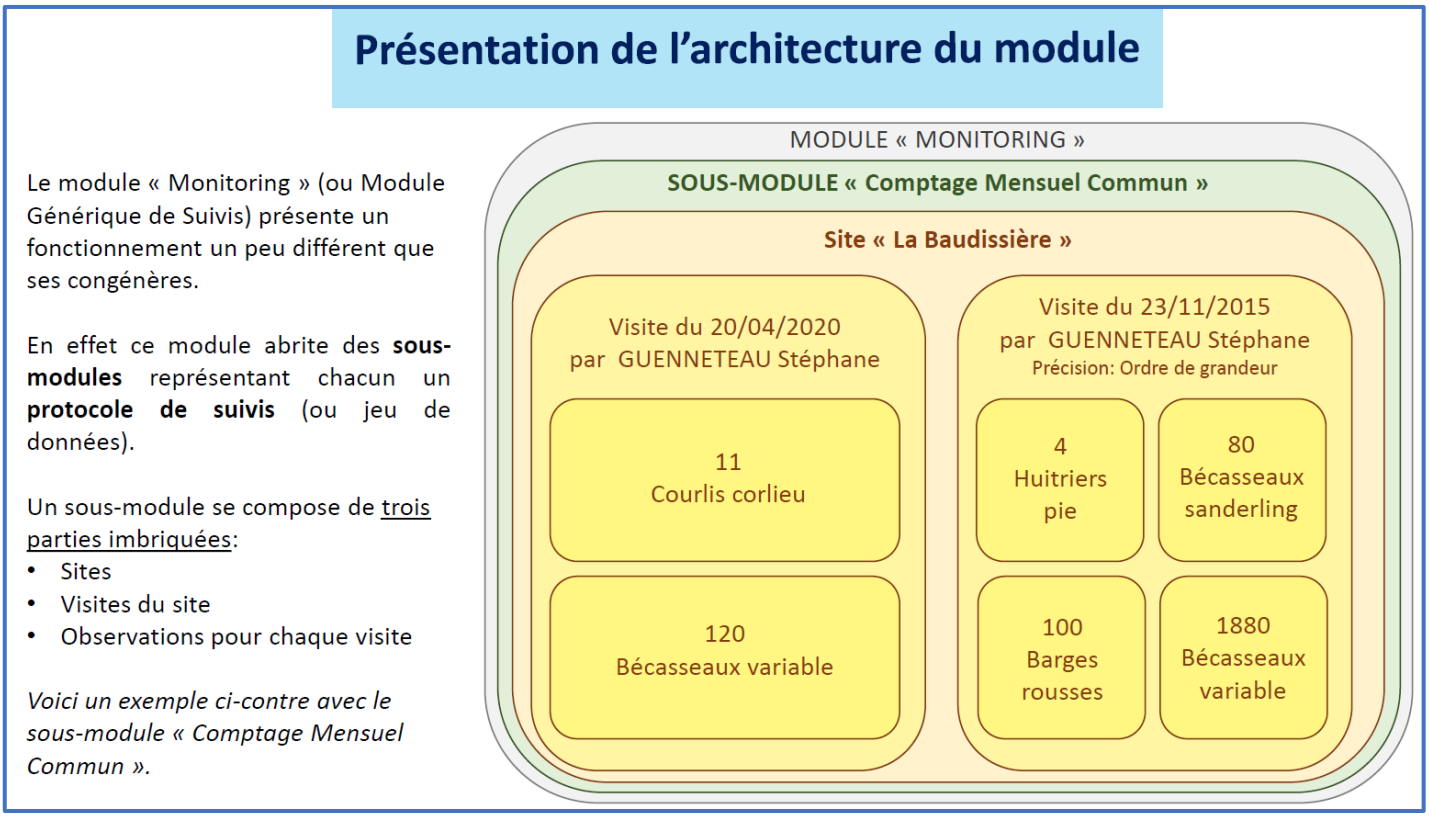


Figure 34: Extrait du guide d'utilisation du module Monitoring créé pour le service SEP de la LPO France (Source : Maxime TOMA)

Ainsi, il m'a fallu rechercher sur le site officiel de PostgreSQL comment marchait ce type de donnée. J'ai pu alors avoir un aperçu des fonctions utilisables dans une requête SQL pour ce genre d'attribut.

Ensuite, j'ai découvert les tables que composait le module. Elles sont au nombre de 10 (cf. MCD ci-après) :

- T_base_sites regroupe les données des sites
- T_site_complements regroupe les champs additionnels en JSONB des sites
- T_base_visits constitue toutes les visites d'un site
- T_visit_complements constitue les champs additionnels en JSONB des visites
- T_observations accueille toutes les observations d'un site
- T_observation_complements accueille les champs additionnels en JSONB des observations
- T_observation_details accueille les champs additionnels en JSONB du détail des observations
- Cor_site_module fait le lien entre les sites et leurs sous-modules de la table t_modules
- Cor_site_area fait le lien entre les sites et le référentiel géographique pour connaitre les aires intersectées
- Cor_visit_observer fait le lien entre les visites et les observateurs de la table t_roles

Les tables des sites, visites et observations sont ainsi en quelque sorte doublées pour accueillir les champs « *data* » ou « *additionnal_data* » en JSONB.

La manœuvre d'insertion reste assez identique à celle pour OccTax :

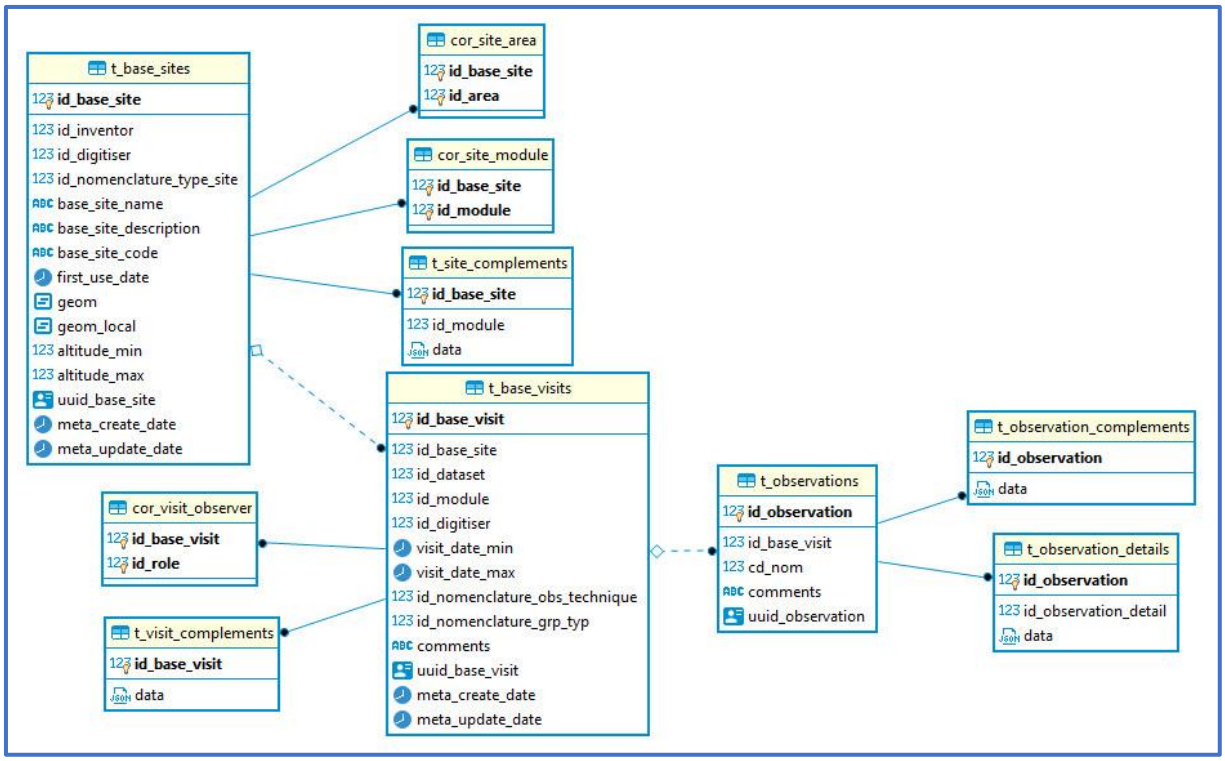


Figure 35: MCD officiel du module Monitoring (Source : GitHub de GeoNature)

1. **Insertion des sites** :

Les sites ont été préalablement intégrés au référentiel géographique de GeoNature à partir des fichiers shape des typologies de site des réserves. Ces derniers sont décomposés en 3 niveaux allant de grands polygones de sites à des polygones plus précis : *Lieux-dit*, *Unité de gestion* et *Sites d'inventaires*. Un quatrième niveau plus précis encore, appelé *Points d'échantillonnage*, a aussi été créé pour la STEPRO.

J'ai donc fait une relation entre les noms (ou code) des sites du référentiel et ceux indiqués dans la table d'origine du protocole (qui devaient être strictement les mêmes). J'ai ainsi récupéré la géométrie, le nom et ai créé ces sites dans le sous-module.

2. **Insertion des vistes à partir de la table d'origine**. Regroupement des éléments dupliqués et tri par l'id dupliqué de façon croissante.
3. **Récupération de *l'id_base_visit*** correspondant à chaque insertion et, en même temps, récupération des ids dupliqué (seuls) ordonnés de façon croissante. Insertion de chaque duo d'id dans une liste de tuples.
4. **Création d'une colonne** qui accueillera l'insertion des *id_base_visit* du tuple en faisant la correspondance avec son id dupliqué.
5. **Insertion de la correspondance entre les rôles** (utilisateurs/observateurs) **et *l'id_base_visit*** issu de la nouvelle colonne.
6. **Insertion de la correspondance entre les champs additionnels et *l'id_base_visit***, issu de la nouvelle colonne, dans la table *t_visit_complements*.
7. **Insertion des observations** à partir de la table d'origine. Tri par l'id de la table d'origine de façon croissante.

8. **Récupération de *l'id_observation*** correspondant à chaque insertion et, en même temps, récupération des ids de la table d'origine (seuls) ordonnées de façon croissante. Insertion de chaque duo d'id dans une deuxième liste de tuples.
9. **Création d'une deuxième colonne** qui accueillera l'insertion des *id_observation* du tuple en faisant la correspondance avec son id.
10. **Insertion de la correspondance entre les champs additionnels et *l'id_observation,*** issu de la nouvelle colonne, dans la table *t_observation_complements*.
11. L'insertion des éléments dans la table *synthese* se fait cette fois-ci manuellement en lançant une commande. Le problème devrait être fixé par les développeurs de GeoNature pour une prochaine mise à jour. Attention, il faut cependant bien configurer le fait que le sous-module est compatible avec le module Synthèse (voir *IV/4.a.III.Création d'un sous-module*)
12. **Insertion des éléments de validation** dans la table *t_validations* selon l'identifiant unique de la table *t_observations*. Cet identifiant unique est de type UUID et est l'attribut de liaison entre le module de synthèse (et donc le module de validation) et l'occurrence à valider.

NB : La requête d'insertion des champs additionnels dans la colonne *data* s'est faite en deux parties à chaque fois. La première si la valeur à entrer est non-nulle, et l'autre si elle est nulle.

Le code python de l'insertion des données historiques du protocole « Comptage Mensuel Commun » de la RNMO est disponible en Annexe XIX.

6. Réalisation d'un plugin QGIS d'import massif de données

Nous allons maintenant passer à la partie de la réalisation du plugin QGIS. Celui-ci a pour vocation d'intégrer des données dans les modules compatibles de GeoNature (OccTax, Monitoring…), de consulter l'historique des insertions tout en ayant la possibilité de supprimer ces dernières en cas d'erreur et de pouvoir modifier ses propres identifiants de connexion à la base de données.

Attention, **l'utilisation de ce plugin doit être fait avec un rôle PostgreSQL de la base ayant les droits du « super-utilisateur »**.

Un langage spécifique sera potentiellement employé dans cette partie, dont je veillerai à les définir, pour permettre la bonne compréhension de toutes les informations, même pour une personne novice.

Un fichier .sql, complémentaire à ce plugin, permet la création d'un schéma spécifique et de la table de l'historique des imports, et est disponible dans les livrables.

a) DIFFERENCES APPORTES PAR LES NOUVELLES VERSIONS

Pour la réalisation de ce plugin, j'ai été contraint d'utiliser PyQt5, non manipulés, en cours et la version 3.7 de Python, compatibles avec QGIS 3.10.

En effet, la base de GeoNature étant sous une version PostgreSQL 11.7, je ne peux plus utiliser les classes de PyQt4. Cependant, la façon de coder reste la même, mais il est important de faire quelques recherches sur les nouveautés apportées par ces versions.

Ces différences concernent les points suivants :

1. L'installation d'un plugin : Le chemin d'accès aux plugins a changé sur la version 3 de QGIS. Ainsi, on peut les retrouver via le chemin : *C:\Users\USER\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins*.
2. La gestion des signaux et slots : La gestion des signaux et des slots a également changé sur la version 5 de PyQt. Vous pourrez retrouver tous les liens consultés dans la partie Bibliographie/Webographie.
3. Les classes PyQGIS : Certaines classes de PyQGIS ont changé. On peut notamment noter le changement de la classe QgsMapLayerRegistry pour la classe QgsProject.

b) CREATION DES FICHIERS DU PLUGIN

Pour commencer, j'ai dû créer le dossier par défaut du plugin que j'ai décidé d'appeler « GeoPIM_SEP », « PIM » étant l'anagramme de « Plugin d'Import Massif » et « Geo » faisant référence à GeoNature. Il est composé des fichiers suivants :

Forms	Contient les .ui (user interface) qui définissent l'interface utilisateur du plugin. Ils sont issus des maquettes IHM.
Help	Il contient deux fichiers PDF : *Guide_Administrateur_GeoPIM*, guide pour installer le plugin QGIS « GeoPIM » ainsi que connecter la base à QGIS, et *Guide_Utilisateur_GeoPIM*, guide d'utilisation du plugin « GeoPIM », étape par étape.
Icons	Il contient les fichiers .PNG des différentes images qui apparaissent dans les menus du plugin. Ce n'est en revanche pas directement ces images qui sont utilisées dans les interfaces graphiques des formulaires du plugin. Ce sont des fichiers de ressources qui les contiennent.
Scripts	Ce dernier dossier regroupe les scripts SQL qui devront être utilisés à l'installation du plugin par l'administrateur.
Dialog.py	Les fichiers de dialogue définissent les fonctions qui sont associées aux form.py contenus dans 'forms'. Ils permettent le fonctionnement de l'interface et le dialogue avec la BD.
Metadata.txt	Ce document contient les informations essentielles sur le plugin comme les créateurs et leur contact, les versions de QGIS qui assurent son bon fonctionnement ainsi que son utilité.
CHANGELOG	Répertorie une description de toutes les versions sorties.
Param_bdd.py	Fichier contenant les paramètres de la base de données.

En parallèle, j'ai créé **un fichier de ressources**. Ce dernier sert à empaqueter des fichiers (images, sons, texte...) au sein du même exécutable. Cela permet de stocker et utiliser au même endroit les icônes d'un projet. J'ai ainsi placé ce fichier dans le dossier « forms » reprenant les chemins des images du répertoire « Icons ». Sur *Qt Designer*, il est ensuite possible d'appeler ce fichier pour ajouter les icônes aux différents formulaires et permet un partage du plugin sans crainte que les icônes ne soient plus sur les formulaires.

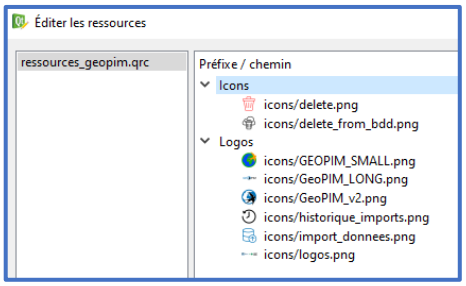


Figure 36: Contenu du fichier de ressources (Source : QtDesigner)

c) CREATION DES MAQUETTES INTERFACES HOMME-MACHINE (IHM) ET DES FICHIERS .UI SOUS QTDESIGNER

I. Création des maquettes

Je dois à présent montrer à Frédéric ROBIN à quoi ressemblera l'outil. Pour ce faire, je dois créer des maquettes du futur plugin afin qu'il voie à quoi ressembleront les différents formulaires qui seront créés. Ils sont disponibles en Annexe XX.

Étant obligé de faire la création du plugin en deux temps, j'ai fait valider auprès de Frédéric ROBIN les IHM pour la partie « Historique » et « OccTax » le mercredi 8 juillet 2020, et la partie sur « Monitoring » le 25 août 2020. Les IHM étant validés par écrit par le commanditaire, je peux entamer la réalisation des formulaires sur *QtDesigner*.

II. Reproduction des maquettes sous QtDesigner

QtDesigner, ou *QtCreator*, est un outil libre disponible avec l'installation de QGIS. Il permet la réalisation d'interfaces graphiques pour les utilisateurs (ou *Graphic User Interface* - GUI) composées de QtWidgets.

Les QtWidgets se présentent sous beaucoup de formes allant de la liste déroulante à la case à cocher, en passant par les boites de texte, les calendriers, etc. En somme, il s'agit d'éléments graphiques à disposer sur une fenêtre afin de demander à l'utilisateur de renseigner les différentes informations à rentrer dans une ou plusieurs tables attributaires.

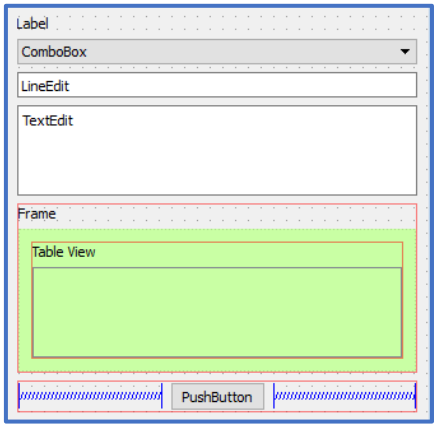


Figure 37: Récapitulatif des QWidgets utilisés pour le plugin "GeoPIM" (Source : QtDesigner)

Voici tous les QtWidgets et leurs différentes utilisations dans mes formulaires :

LineEdit	Le LineEdit permet à l'utilisateur de rentrer une courte suite de caractères. *Exemple : Nom de la* table *à intégrer*

ComboBox	La ComboBox renseigne à l'utilisateur une liste d'éléments présents notamment dans une table attributaire. *Exemple : Liste des modules*
TextEdit	Le TextEdit quant à lui permet à l'utilisateur de rentrer une longue suite de caractères. *Exemple : Messages de log lors de l'insertion des données*
Label	Le Label est juste là pour indiquer une information supplémentaire à l'utilisateur. Généralement, il indique à quoi correspond un QtWidget adjacent.
PushButton	Ce sont les boutons cliquables et les plus communs dans QtDesigner. *Exemple : Bouton de lancement du script d'import*
Frame	Le Frame permet de regrouper un certain nombre de QtWidget ensemble, en apportant un aspect esthétique supplémentaire. *Exemple : Widgets à activer/désactiver dans certaines conditions*
TableView	La TableView permet l'affichage d'une table attributaire sous la forme d'un tableur. On peut notamment cacher certaines colonnes ou encore rendre directement éditable celui-ci ou non. *Exemple : Visualisation de l'historique des imports*
Spacer	Le Spacer permet d'agencer le formulaire.
ProgressBar	La ProgressBar permet l'affichage d'une barre de progression pour simuler un temps de chargement ou définir une valeur. *Exemple : Temps de réalisation de l'import de données*
Layouts	Les layouts permettent une mise en place ordonnée des différents widgets à l'intérieur de celui-ci. Il permet également un positionnement et un redimensionnement dynamique des widgets selon le redimensionnement manuel d'une fenêtre. Grid dispose les Widgets selon une grille et Form selon un formulaire.

III. Application d'une nomenclature

Afin de gérer au mieux tous les éléments d'un formulaire sous python, je dois déjà adopter une bonne nomenclature sous *QtDesigner*.

Ainsi, pour chaque type de widgets, j'ai utilisé le même préfixe. Celui-ci fut ensuite accompagné le plus souvent du nom de colonne de l'attribut de la table souhaitée (ex : cmbox_id_module). Voici ci-contre la liste des préfixes de ma nomenclature. Je retrouve ensuite plus facilement par autocomplétion les QWidgets grâce à l'environnement PyCharm (permettant de développer du code en python).

QT	Renommé	
Label	**lbl**	
QListWidget	**qlist**	
Pushbutton	**btn**	
LineEdit	**le**	
QdateEdit	**dat**	
QCombobox	**cmbox**	
QWidget	**w**	
Table View	**tab**	
btradio	**rad**	
time	**time**	
textEdit	**text**	
spinbox	**spin**	**dbspin**
checkBox	**chkbox**	

Figure 38: Nomenclature utilisée pour QtDesigner

IV. Compilation des fichiers .ui

Pour utiliser les QWidgets dans le développement du code, je dois compiler les fichiers .ui obtenus. Le fichier .py compilé décrira sous python comment sont agencés les éléments du formulaire. Ce fichier .py sera ensuite appelé par le fichier de dialog qui fera le lien entre la forme graphique du formulaire et la base de données.

Il existe plusieurs possibilités pour compiler un fichier .ui en .py, mais j'ai décidé d'utiliser un exécutable en .bat lançant en cascade la commande pour tous les formulaires.

```
C:\Python37\Scripts\pyuic5.exe import_form.ui > import_form.py
C:\Python37\Scripts\pyuic5.exe import_ajout_form.ui > import_ajout_form.py
C:\Python37\Scripts\pyuic5.exe historique_form.ui > historique_form.py

C:\Python37\Scripts\pyrcc5.exe ressources_geopim.qrc > ressources_geopim_rc.py
```

Figure 39: Code du fichier compile.bat

d) CREATION DES MENUS / SOUS-MENUS ET DU FICHIER PRINCIPAL GEOPIM_PLUGIN.PY

Passons à présent à la création du fichier principal du plugin nommé sobrement *geopim_plugin.py*. Celui-ci, appelé par le fichier __init__.py à l'ouverture du plugin par QGIS, permet de répertorier toutes les ouvertures de formulaires associés à des QActions, et de former un QMenu ainsi qu'une toolbar.

Un QMenu peut répertorier plusieurs QActions qui peuvent, eux-mêmes, être utilisés dans une toolbar. Mais pour mieux comprendre, voici un exemple avec le plugin GeoPIM (en bleu une partie de l'interface de QGIS, en vert la "toolbar" et en rouge le QMenu et les QActions sous forme de menu déroulant) :

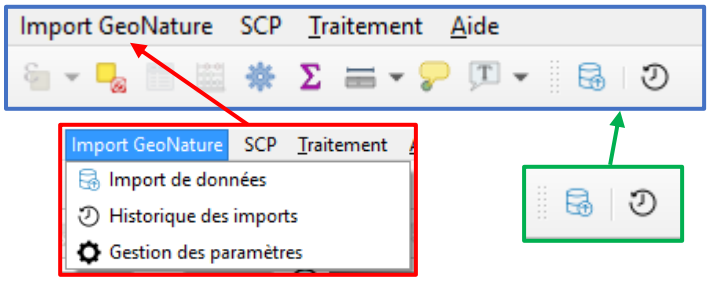


Figure 40: Présentation du QMenu du plugin GeoPIM (Source : QGIS)

Le code du fichier, disponible en Annexe XXI, est composé de plusieurs classes et méthodes permettant l'ouverture et fermeture des différents dialogs/formulaires :

GeoNatureLPOImportPlugin	Classe principale du plugin. Elle contient les 4 méthodes suivantes :
initGui	Cette méthode va définir le QMenu et les « signaux » et « slot » des QActions. Le « signal » est l'élément déclencheur qui interpelle une méthode pour résultat (« slot »).

gereImport	Elle a pour rôle d'appeler la classe *ImportDialog* présente dans le fichier *import_dialog.py*, importé au tout début du code. Elle est appelée quand le QAction « Import de données » a été cliqué.
gereHistorique	Elle a pour rôle d'appeler la classe *HistoriqueDialog* présente dans le fichier *historique_dialog.py*, importé au tout début du code. Elle est appelée quand le QAction « Historique des imports » a été cliqué.
gereParam	Elle a pour rôle d'appeler la classe *GestionParamDialog* présente dans le fichier *gestion_param_dialog.py*, importé au tout début du code. Elle est appelée quand le QAction « Gestion des paramètres » a été cliqué.

e) CREATION DES FICHIERS TYPES (.CSV / .SHP)

Afin de contrôler l'intégration des données et que cette dernière reste simple d'utilisation, je dois créer un fichier « type » standardisé pour tous les utilisateurs du GeoNature SEP. En effet, il faut que ce dernier présente le même nombre de colonnes, les mêmes valeurs de liste déroulante que sur GeoNature, etc.

J'ai également décidé de gérer ces listes déroulantes avec leurs valeurs et non leurs identifiants, pour l'affichage « utilisateur », afin que le fichier soit le plus simple d'utilisation. En revanche, j'ai créé une feuille alternative (« Imports ») qui prendra les identifiants au lieu des valeurs et qui sera la feuille à enregistrer au format CSV (UTF-8) pour l'import.

Ainsi, le fichier « type » pour l'insertion dans le module OccTax et dans le sous-module « Comptage Mensuel Commun », disponibles dans les livrables du stage, ont été créés sous l'extension .xlsx et possèdent 5 feuilles :

- Observations aléatoires (ou Comptage Mensuel Commun) : Feuille principale, contenant toutes les données des relevés, remplie par les utilisateurs.
- Imports : Feuille qui sera à enregistrer en CSV (UTF-8) pour l'import dans GeoNature.
- Sites : Feuille regroupant tous les sites utilisés par la LPO.
- Nomenclatures : Feuille regroupant toutes les items de nomenclatures utilisés en fonction du module/sous-module.
- Utilisateurs : Feuille regroupant tous les utilisateurs actifs de UsersHub.
- Taxons : Feuille regroupant tous les taxons des listes utilisées en fonction du module/sous-module de TaxHub.

La feuille principale fait appel aux autres feuilles pour la création de certaines cellules au format de liste. De plus, certaines de ces listes ont même été configurées pour que les valeurs de liste s'ajoutent par autocomplétion.

Par conséquent, pour chaque nouvelle insertion dans la base de données, l'utilisateur devra **sauvegarder la feuille principale au format « .csv »** en veillant bien qu'il soit bien **encodé en**

UTF-8. Une fois les données insérées, l'utilisateur devra repartir sur un nouveau fichier « type » vierge pour éviter d'insérer deux fois les mêmes données. A noter que sur les nouvelles versions d'Excel, il existe la possibilité d'enregistrer au format « CSV UTF-8 ».

Une mise à jour régulière des valeurs de liste est indispensable pour éviter toute erreur lors des insertions.

Voyons maintenant le fonctionnement de chaque fichier de dialogue.

f) CREATION DES FICHIERS DE DIALOGUES PAR MODULE

I. Gestion des paramètres

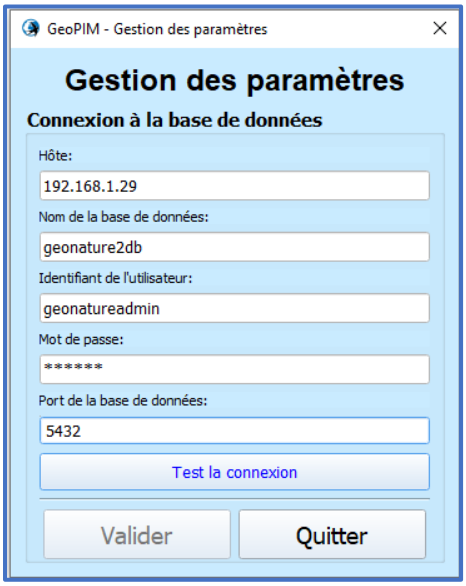


Figure 41: Boite de dialogue de la "Gestion des paramètres" (Source : GeoPIM)

Le fichier *gestion_param_dialog.py* a la charge de la gestion et de l'affichage des paramètres de la base de données. Il fait appel aux éléments enregistrés du fichier *param_bdd.py*.

Un bouton permet de « Tester la connexion » avant de faire apparaitre le bouton « Valider » et de cliquer dessus.

Les modifications sont alors directement apportées dans le fichier *param_bdd.py* si la connexion à la base s'est correctement faite avec ces identifiants. Attention toutefois, le mot de passe n'est pas visible en clair sur le plugin, mais l'est dans les fichiers de configuration.

La modification sera cependant active que si vous relancez QGIS manuellement ou via le plugin « Plugin Reloader ».

II. Historique

La fenêtre de gestion de « l'Historique des imports » se présente sous la forme d'un QTableView. Ce dernier est lié à la table du schéma *lpo_imports* de la base de données.

Deux boutons sont également disponibles, après avoir sélectionné un import : le premier, supprime uniquement la table servant pour l'import de données, mais ne donne plus la possibilité de supprimer ces données au sein du module où elles ont été intégrées. Le deuxième donne cette possibilité.

Seuls les attributs du nom de la table, du JDD, du module et de la date et heure de l'import sont visibles. La table est, cependant, composée de 2

Figure 42: Boite de dialogue de "l'Historique des imports" (Source : GeoPIM)

autres attributs booléens désignant si la table a été supprimée ou si la table et les données dans le module ont été supprimées. L'import n'apparaitra plus sur la fenêtre, mais restera quand même visible dans la table de la base de données pour la traçabilité.

III. Dialogue principal des imports

Figure 43: Boite de dialogue principale pour l'import de données (Source : GeoPIM)

Il s'agit de la fenêtre de dialogue principale du module. C'est à partir de celle-ci que l'on va définir le module dans lequel les données seront insérées, le fichier (.csv/.shp) d'origine, le jeu de données associé et enfin le nom de la table d'insertion. Ce dernier permettra d'identifier l'insertion dans la fenêtre « Historique des imports ».

Une vérification s'effectue lors de la rédaction du nom de la table pour vérifier qu'elle n'existe pas déjà dans la base de données. Une autre vérification s'effectue également sur le nom du module. Si ce dernier n'est pas disponible pour l'insertion de données, un message d'avertissement apparaitra.

Également, chaque QComboBox est réglé par défaut sur une valeur nulle. Ainsi, le jeu de données et le nom de la table ne peuvent être saisis que si le fichier a été choisi, et lui-même ne peut être saisi que si le module l'a été.

Une fois que tous les éléments sont valides, le bouton « Valider » s'active et l'on peut passer à l'import dans le module.

IV. Import dans le module OccTax

Le code du fichier de dialogue des imports dans le module « OccTax » se base sur le travail effectué sur les scripts d'intégration des données historiques.

Cette fenêtre s'affiche après avoir appuyé sur le bouton « Valider » de la boite de dialogue vue précédemment. Elle est composée de deux boutons, d'un QProgressBar et d'un QTextEdit. De plus, on ne peut plus accéder aux autres éléments de QGIS tant que cette fenêtre est active.

Le bouton « Lancer l'import » va exécuter le script d'intégration. En parallèle, vont s'afficher les messages de gestion du script dans le QTextEdit pour savoir où il en est dans l'intégration des données, et le QProgressBar indiquera son pourcentage. Enfin, le QTextEdit passera en vert si l'intégration est un succès ou en

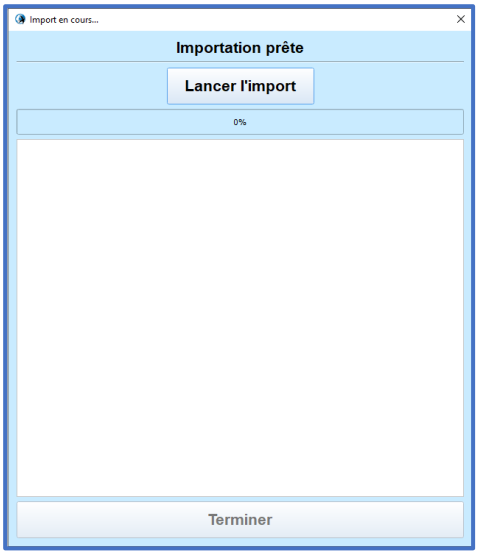


Figure 44: Boite de dialogue des imports dans le module "OccTax" et le sous-module "Comptage Mensuel Commun" (Source : GeoPIM)

rouge si c'est un échec et affichera la localisation du problème. Enfin, le bouton « Terminer » sera cliquable.

Concernant le code, on va en premier créer la nouvelle table d'insertion nommée précédemment. Ensuite, on intègre les données via la requête COPY. Cette dernière ne fonctionne pas comme une requête normale si on souhaite intégrer la donnée sur un serveur distant (comme c'est le cas ici). Ainsi, je passe par une commande exécutée sur une invite de commande. Cela a pour effet d'ouvrir temporairement l'invite pendant l'import. Enfin, on transforme les X et Y en champ géométrique (geom).

Les données sont ensuite insérées par niveau : table des relevés, table de liaison entre les observateurs et les relevés, table des occurrences de relevé, table de liaison entre le dénombrement et les occurrences de relevé (plusieurs triggers s'occupent de créer à ce stade les données dans les tables de synthèse) et enfin table de validation des données.

Une fois les données correctement insérées, le nom de la table, le nom du module, le nom du JDD et la date et heure d'import sont insérés dans la table de « l'historique des imports ».

Un contrôle permanent est fait pour vérifier que les requêtes se sont correctement exécutées. Dans le cas contraire, le script est stoppé et la table d'insertion et les premières données intégrées sont supprimées.

V. Import dans le module Monitoring (« Comptage Mensuel Commun »)

Le code du fichier de dialogue des imports dans le sous-module « Comptage Mensuel Commun » se base sur le travail effectué sur les scripts d'intégration des données historiques.

Il s'agit de la même fenêtre utilisée pour l'import dans « OccTax » et a donc le même fonctionnement.

Concernant le code d'insertion, il est en revanche basé sur les scripts d'intégration des données historiques de la RNMO pour ce protocole (voir Annexe XIX)

Tous les fichiers du plugin sont disponibles dans les livrables. Cependant, le code du fichier d'import dans le sous-module « Comptage Mensuel Commun »
import_monitoring_cmc_dialog.py est disponible en Annexe XXII.

7. Personnalisation de l'interface principale de GeoNature & Rédaction des guides

a) PERSONNALISATION GRAPHIQUE DE GEONATURE (FRONTEND)

GeoNature permet également de personnaliser graphiquement l'interface principale de l'application. En effet, sont ainsi personnalisables :

- **Les logos de l'application** : Ils seront placés à côté du nom de l'application. Ils sont définis par le fichier *logo_structure.png* dans le chemin « frontend/src/images ».

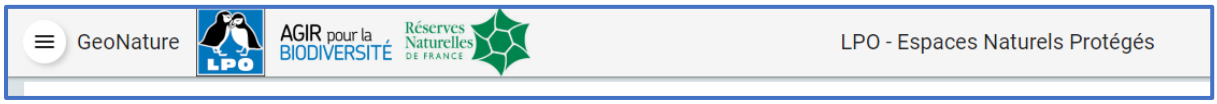


Figure 45: En-tête de GeoNature présentant les logos et nom de l'application (Source : GeoNature du SEP de la LPO)

- **L'image de la page de connexion** : On peut également changer l'image affichée sur la page de connexion de GeoNautre.

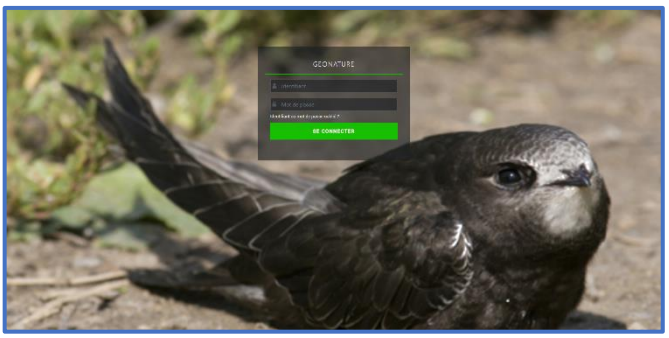


Figure 46: Page de connexion de GeoNature (Source : GeoNature du SEP de la LPO)

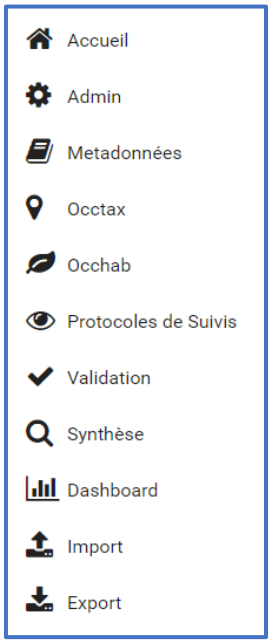


Figure 47: Menu des modules dans GeoNature (Source : GeoNature du SEP de la LPO)

- **Les pictogrammes et le nom affiché des modules** : Les pictogrammes et noms affichés des modules sont également personnalisables. Cependant, ils ne sont modifiables que via la base de données en modifiant les données de la table « t_modules ».

- **La bannière d'introduction** : La bannière d'introduction est située en dessous du nom et logo de l'application. Elle permet d'afficher un message de présentation que l'on peut personnaliser. Cette personnalisation se fait via un fichier HTML à modifier.

Figure 48: Bannière d'introduction de l'application GeoNature (Source : GeoNature du SEP de la LPO)

- **Le fichier de style (CSS) :** Enfin, un fichier en CSS permet de personnaliser ce que l'on veut en termes de taille, de design de bouton, de couleurs... Il faut, en revanche, une connaissance des classes et index issus du code HTML principal de l'application.

En accord avec Frédéric ROBIN, nous avons apporté une personnalisation graphique sur tous les points cités précédemment.

b) REDACTION DES GUIDES UTILISATEUR ET ADMINISTRATEUR DE GEONATURE

Afin de faciliter la prise en main de ce nouvel outil, il m'a été demandé par Frédéric ROBIN de réaliser **un guide utilisateur par module** à destination des utilisateurs, ainsi qu'**un guide administrateur** pour la gestion, postérieure à mon stage, de GeoNature.

Ainsi, chaque guide d'utilisation va faire une présentation du module, de son utilité, et des manipulations à effectuer. Un exemple avec le guide utilisateur du module « Monitoring » est disponible en Annexe XXIII.

Pour le guide administrateur, j'ai choisi de faire référence à la documentation officielle de GeoNature, qui explique déjà très bien les manipulations à effectuer, au lieu de tout recopier. Cependant, certains aspects, voir modules, sont encore absents de cette documentation officielle comme les modules « Admin », « Métadonnées » notamment. Ainsi, je les ai rajoutés dans le guide administrateur afin qu'il soit le plus complet possible. De plus, j'ai rajouté les manipulations à effectuer pour se connecter en FTP et SSH au serveur. Ce guide est disponible dans les livrables du projet.

c) REDACTION DES GUIDES UTILISATEUR ET ADMINISTRATEUR DU PLUGIN

Parallèlement aux guides de GeoNature, il me fallait également rédiger ceux du plugin QGIS. Ces derniers seront essentiels pour l'utilisation du plugin.

C'est pourquoi j'ai rédigé un premier manuel « administrateur », disponible en dans les livrables du projet, présentant l'installation du plugin ainsi que le tutoriel de connexion de la base sous le logiciel Qgis, mais également sous le logiciel pgAdmin4 conformément à la commande. Une dernière partie présentera la composition du plugin, et son principe de fonctionnement.

Enfin, j'ai rédigé un second manuel à destination des futurs utilisateurs de ce plugin, disponible en Annexe XXIV. Il présente une description sommaire de l'interface du plugin sous le logiciel Qgis, mais également le fonctionnement de chacune des interfaces.

V/ BILAN DU DEPLOIEMENT DE GEONATURE

Pour cette dernière grande partie, je vais développer un bilan global et critique de mon travail effectué au sein du service Espaces Protégés de la LPO. Nous verrons ainsi les difficultés que j'ai rencontré, les coûts que cela a engendrés et la continuité du projet.

1. Difficultés rencontrées

a) CONFIGURATION DES MODULES PRINCIPAUX

La difficulté principale que j'ai pu avoir sur la configuration des modules principaux serait mon manque de compétences sur les outils Flask et Angular utilisés par GeoNature. Cependant, la documentation officielle est adaptée pour les non-initiés et permet donc d'appréhender plus facilement cette configuration.

b) REALISATION DU PLUGIN

Pour la réalisation du plugin, j'ai dû faire face à un problème majeur. En effet, au vu de la version 11.7 de PostgreSQL, j'ai ainsi développé le plugin en Python 3.7 compatible avec PyQt5 et QGIS 3.10. Je n'ai cependant pas vu ces versions de PyQt et PyQGIS en cours et j'ai dû me renseigner sur des forums et sur les documentations officielles concernant les changements apportés depuis la version vue avec Alain LAYEC en cours. Finalement, ces changements n'affectèrent pas beaucoup mon code au vu des fonctions, méthodes et aspects que je devais utiliser et développer et qui n'avaient, pour la plupart, pas changé.

Une autre difficulté concerne le compile.py fourni par M. LAYEC en cours. Ce dernier ne marchait plus pour ma part et j'ai ainsi dû créer un nouveau fichier (en .bat) pour une compilation groupée des fichiers .ui.

c) INSTALLATION ET CONFIGURATION DES MODULES COMPLEMENTAIRES

Concernant les modules complémentaires, et plus spécifiquement le module Monitoring, la difficulté qui s'est posée est qu'il s'agit d'un nouveau module. Il n'est, de ce fait, pas encore complet et nécessite quelques améliorations afin d'ajouter des nouvelles et anciennes données de certains protocoles. De plus, la SHF (Société Herpétologique de France) développe un module exclusif à tous les protocoles standardisés de l'herpétofaune. Ainsi, toutes les données historiques du SEP de la LPO n'ont pas pu être intégrées dans GeoNature.

d) LE SERVICE INFORMATIQUE DE LA LPO ET LE CRASH DU SERVEUR

Enfin, ma plus grande difficulté reste le crash fin juillet du serveur de la LPO abritant le GeoNature du SEP et de la charge de travail du Service informatique.

Ce dernier est composé de trois personnes : Serge THOMAS (Responsable du service), Cédric GAUDEFROY (Administrateur Systemes et réseaux), et François BOUCHET (Responsable de projet Web et référent sur la mise en place et la maintenance de GeoNature).

François BOUCHET est donc le seul à avoir les compétences requises pour la gestion technique et l'administration du serveur. Or, le crash est survenu la dernière semaine de juillet, lors de ses congés, laissant le serveur inopérant jusqu'à début août. Par la suite, François BOUCHET n'a pas eu la priorité sur notre problème de la part de sa hiérarchie au vu de la charge de travail conséquente du SI à cause de la crise sanitaire.

Par ailleurs, François s'occupe de toutes les problématiques « web/serveur » de la LPO France et de certaines LPO et partenaires extérieures. La situation du serveur est restée ainsi jusqu'à fin-août/début-septembre avant que François soit arrêté pour raison médicale rendant impossible l'installation du nouveau serveur avant mon départ de la structure.

Afin de ne pas perdre un mois de travail, j'ai pris l'initiative d'installer la dernière version de GeoNature sur une machine virtuelle, en m'appuyant sur le cours de Julien HUBEBRT sur Lizmap. J'ai ainsi pu continuer à remplir mes objectifs, mais dans des conditions beaucoup moins favorables, et sans aucune visibilité pour Frédéric ROBIN et les membres des réserves qui ne pouvaient pas accéder à mon GeoNature.

2. Coûts

Au cours de la mise en place de GeoNature pour les Espaces Protégés de la LPO France, différents coûts ont pu être relevés.

a) FINANCIERS

Les coûts financiers du projet durant mon stage sont les suivants :

Type de coût	Prix
Coût du stagiaire	2000,7 € (soit 3,9€ x 513h)
Coût du personnel aidant* (Frédéric ROBIN)	Environ 5000€
Coût du personnel aidant* (François BOUCHET)	Indéterminable
Coût du personnel aidant* (Techniciens de réserves)	Environ 1500 €
Coût de l'application GeoNature	0€
Total :	**Environ 8500,7 €**

** Le coût du personnel aidant correspond au temps de sollicitation multiplié par le coût horaire de cet agent. Au total, 13 personnes m'ont apporté leur aide durant le projet.*

La situation actuelle (crise sanitaire) fait qu'il est difficile de calculer les frais de structure (personnes étant en télétravail le plus souvent, etc..).

b) TEMPORELS

Les coûts en termes de temps pour mettre en place l'application GeoNature comprennent les 16 semaines de stage que j'ai pu réaliser, soit 513 heures.

De plus, nous pouvons comptabiliser le temps de travail passé sur le projet selon le personnel aidant : environ 672 heures (4j * 6 mois) pour Frédéric ROBIN et environ 44 heures (11 x 4h) pour le personnel des réserves. Nous pouvons faire également mention des 3 heures (1h x 3j) de réunion avec le service informatique.

3. Continuité du projet

a) INSTALLATION ET CONFIGURATION DE ATLAS, MOBILE, CITIZEN ET D'AUTRES MODULES EN COURS DE DEVELOPPEMENT

Un premier aspect de la continuité du projet est l'installation et la configuration d'autres modules sortis ou en cours de développement. En effet, certains modules seraient intéressants à utiliser comme le module Atlas et le module Citizen pour une diffusion et une collecte de données naturalistes grand-public, le module de la SHF toujours en cours de développement et permettant de saisir des données venant de protocoles herpétologiques, ou encore le module Occtax-Mobile qui permet la saisie sur le terrain d'observations aléatoires de taxons via un smartphone ou une tablette.

b) CREATION D'AUTRES SOUS-MODULES DE MONITORING

Une autre grande continuité est la création de nouveaux sous-modules du module Monitoring. Cette création permettrait de saisir et importer des données d'autres protocoles que je n'ai pas eu le temps d'instaurer dans GeoNature. Une mise à jour serait cependant à attendre pour certains protocoles (comme le « Distsance Sampling ») concernant la possibilité de mettre en place une géométrie sur les données de visites et d'observations.

c) INTEGRATION DES AUTRES DONNEES HISTORIQUES

Faisant écho aux points précédents, je n'ai pas pu importer toutes les données historiques des réserves. Il restera donc à les insérer dans les différents modules, dont ceux à paraître prochainement.

d) MISE A JOUR DES MODULES ET DU PLUGIN

Enfin, une maintenance de l'outil reste à faire afin de mettre à jour les différents modules et l'application principale, ainsi que le plugin pour importer des données dans de nouveaux modules ou pour appliquer certaines modifications dues aux mises à jour de ces derniers.

CONCLUSION

Le service Espaces Protégés de la Ligue pour la Protection des Oiseaux de France cherche aujourd'hui à revoir leurs méthodes de saisie de données naturalistes spatialisées. Ainsi, les Systèmes d'Informations Géographiques apparaissent au cœur de ce renouveau.

L'outil open source GeoNature est apparu comme une alternative solide, actuelle et gratuite du logiciel SERENA qu'ils utilisaient précédemment. Le mode de fonctionnement de SERENA ne convient plus pour les protocoles actuels de la LPO. De plus, GeoNature jouit aujourd'hui d'une réputation de plus en plus rependue au sein des structures, dont beaucoup d'associations, naturalistes et permet ainsi un retour d'expérience assez exhaustif pour l'équipe de développement de l'outil, toujours aussi active. L'intérêt de cette réputation y est double : respectant les standards français du SINP, l'échange de ces données se retrouve grandement facilité notamment auprès des organismes de l'État. Étant open source, chacun peut ainsi aider l'équipe de développement pour améliorer certains aspects de l'application voire même le développement de nouveaux outils. L'outil reste cependant assez incomplet pour la LPO France aujourd'hui, mais reste attentif et intéressé aux actualités de GeoNature.

J'ai ainsi pu, au travers de cette expérience, découvrir cet outil et proposer à l'équipe du SEP de la LPO France l'installation et la configuration de modules nécessaires à la saisie de différents protocoles ainsi qu'à l'import de données historiques, datant pour certaines des années 80, complété par le développement d'un plugin QGIS. Cela m'a permis de mettre en application et développer mes connaissances et compétences acquises durant la Licence professionnelle SIG de La Rochelle.

Je me suis également adapté aux problèmes qui se sont présentés devant moi comme le développement du plugin dans une version ultérieure à celle vue en cours, ainsi que le crash serveur qui a failli mettre en péril la fin de mon stage. Grâce au cours de Julien HUBERT sur Lizmap, j'ai installé en autonomie sur une machine virtuelle une version de GeoNature. J'ai alors pu continuer mon travail sans dépendre des délais imposés par le service informatique. Le service informatique n'était pas toujours disponible sur le projet de GeoNature à cause d'une charge de travail plus conséquente due à la crise sanitaire.

Les conditions actuelles (départ du directeur de pôle, arrêt maladie de ma chef de service et du responsable web du service informatique) font qu'il n'y a plus de GeoNature opérant. Néanmoins, tous mes travaux pourront être intégrés au nouveau serveur quand il sera disponible. Ils permettront aux utilisateurs du GeoNature du SEP de saisir des observations aléatoires, des données d'habitats, des données de comptage, etc.

Pour finir, cette expérience a été très enrichissante pour moi, tant sur l'aspect humain, relationnel, que technique. Frédéric ROBIN et l'ensemble du service ont été pour moi moteurs de motivation et m'ont partagé leurs connaissances sur la nature, sur le fonctionnement de l'association et en globalité sur la réalité du monde professionnel. Cela m'a également permis de conduire un projet en situation et de voir les contraintes et les stratégies à mettre en œuvre pour les contourner.

TABLE DES FIGURES

Figure 1: Carte de répartition des antennes LPO (Source : lpo.fr) 8
Figure 2: Lieutenant Hémery portant deux macareux moines (Source : lpo.fr) 9
Figure 3: Organigramme simplifié de la LPO (Source : Maxime TOMA) 10
Figure 4: Organigramme détaillé du Service Espaces Protégés (Source : lpo.fr) 11
Figure 5: Image de présentation de SERENA (Source : http://www.serena-rnf.net/v2/) 11
Figure 6: Logo de GeoNature (Source : geonature.fr) 12
Figure 7: Répartition des différentes réserves naturelles gérées de la LPO (Source : Maxime TOMA) 13
Figure 8: Niveau 1 de l'Organigramme des Tâches....18
Figure 9: Planning GANTT prévisionnel 19
Figure 10: Planning GANTT définitif 20
Figure 11: Logos de Flask et Angular 21
Figure 12: GeoNature-atlas du Parc National du Mercantour (Source : http://biodiversite.mercantour-parcnational.fr/) 23
Figure 13: GeoNature-citizen du Parc National du Mercantour (Source : https://obs.mercantour-parcnational.fr/home)....23
Figure 14: Logo du logiciel R....28
Figure 15: Interface principale de GeoNature (Source : GeoNature du SEP de la LPO) 28
Figure 16: Logo de GitHub....29
Figure 17: Schéma de l'architecture de GeoNature issu de la documentation officielle (Source : geonature.fr) 29
Figure 18: Schéma simplifié du fonctionnement des modules dans GeoNature (Source : Maxime TOMA) 30
Figure 19: Interface principale de TaxHub (Source : GeoNature du SEP de la LPO)....32
Figure 20: Interface principale de UsersHub (Source : GeoNature du SEP de la LPO) 33
Figure 21: Interface de gestion des permissions, du module Admin (Source : GeoNature du SEP de la LPO)....34
Figure 22: Interface de gestion des nomenclatures et exports du module Admin (Source : GeoNature du SEP de la LPO)....35
Figure 23: Exemple d'un item ayant le statut "Validation en cours" et étant actif (Module Admin)....35
Figure 24: Exemple d'un item ayant le statut "Validation en cours" et étant actif (Module Monitoring) 35
Figure 25: Interface du module OccHab (Source : GeoNature du SEP de la LPO)....37
Figure 26: Liste déroulante de l'attribut "Statut source" (Source : GeoNature du SEP de la LPO) 37
Figure 27: Statuts de validation des occurrences taxonomiques dans GeoNature (Source : GeoNature du SEP de la LPO)....38
Figure 28: Exemple du champ "data" en JSONB stockant le champ personnalisé "id_nomenclature_precision_comptage" (Source : pgAdmin4) 39
Figure 29: Interface du module "Export" lors de la création d'un export (Source : GeoNature du SEP de la LPO) 41
Figure 30: Extrait de la carte "Synthèse par entité géographique" montrant les 2 types de classe (Source : GeoNature du SEP de la LPO) 42
Figure 31: MCD simplifié du schéma de donnée du protocole OccHab (Source : Maxime TOMA)....45
Figure 32: Schéma simplifié de la manœuvre adoptée pour l'insertion des données d'habitats dans OccHab (Source : Maxime TOMA) 46
Figure 33: MCD simplifié du schéma de donnée du protocole OccTax (Source : Maxime TOMA) 47

Figure 34: Extrait du guide d'utilisation du module Monitoring créé pour le service SEP de la LPO France (Source : Maxime TOMA) 49
Figure 35: MCD officiel du module Monitoring (Source : GitHub de GeoNature) 50
Figure 36: Contenu du fichier de ressources (Source : QtDesigner) 53
Figure 37: Récapitulatif des QWidgets utilisés pour le plugin "GeoPIM" (Source : QtDesigner) 53
Figure 38: Nomenclature utilisée pour QtDesigner 54
Figure 39: Code du fichier compile.bat 55
Figure 40: Présentation du QMenu du plugin GeoPIM (Source : QGIS) 55
Figure 41: Boite de dialogue de la "Gestion des paramètres" (Source : GeoPIM) 57
Figure 42: Boite de dialogue de "l'Historique des imports" (Source : GeoPIM) 57
Figure 43: Boite de dialogue principale pour l'import de données (Source : GeoPIM) 58
Figure 44: Boite de dialogue des imports dans le module "OccTax" et le sous-module "Comptage Mensuel Commun" (Source : GeoPIM) 58
Figure 45: En-tête de GeoNature présentant les logos et nom de l'application (Source : GeoNature du SEP de la LPO) 60
Figure 46: Page de connexion de GeoNature (Source : GeoNature du SEP de la LPO) 60
Figure 47: Menu des modules dans GeoNature (Source : GeoNature du SEP de la LPO) 60
Figure 48: Bannière d'introduction de l'application GeoNature (Source : GeoNature du SEP de la LPO) 60

BIBLIOGRAPHIE ET WEBOGRAPHIE

Compréhension du projet / de l'application

Sites WEB

- Documentation de GeoNature – [en ligne] Disponible sur http://docs.geonature.fr/ [consulté entre mai et septembre 2020]
  - Installation Globale
  - Manuel Utilisateur
  - Manuel Administrateur
  - Import Niveau 1
  - Import Niveau 2
  - Développement
  - Compatibilité
- Documentation de GeoNature par module – [en ligne] Disponible sur https://github.com/PnX-SI/GeoNature#modules-et-projets-liés [consulté entre mai et septembre 2020]
  - UsersHub
  - TaxHub
  - GeoNature-atlas
  - GeoNature module Export
  - GeoNature module Import
  - GeoNature module Dashboard
  - GeoNature module Validation
  - GeoNature module générique de suivis
- Forum StackOverflow.com [en ligne] – Disponible sur https://stackoverflow.com/questions/ [consulté entre mai et septembre 2020]
  - I am using office and Flask-mail : https://stackoverflow.com/questions/54600601/i-am-using-office-and-flask-mail

Tickets Github

- GeoNature
  - Notification de mise à jour : https://github.com/PnX-SI/GeoNature/releases
  - JSONB vers colonnes brutes : https://github.com/PnX-SI/Ressources-techniques/blob/master/PostgreSQL/pivot_query.sql
  - Champs personnalisés : https://github.com/PnX-SI/GeoNature-citizen/issues/181
  - Script d'import pour différents modules : https://github.com/PnX-SI/gn_module_suivi_flore_territoire/blob/develop/docs/import-data.md
  - Ajout d'un auteur de la métadonnée : https://github.com/PnX-SI/GeoNature/issues/921
- UsersHub
  - https://github.com/PnX-SI/UsersHub/blob/master/data/usershub.sql#L61
- Synthèse
  - Pourquoi la synthèse à plat ?https://github.com/PnX-SI/GeoNature/issues/924
- GeoNature module générique de suivis

- Script de création : https://github.com/PnX-SI/gn_module_monitoring/blob/develop/data/schema_suivis_generique.sql#L17
- Trigger Monitoring vers Synthèse : https://github.com/PnX-SI/gn_module_monitoring/issues/14
- MLD : https://github.com/PnX-SI/gn_module_monitoring/issues/3#issuecomment-582853124

Base de données / SQL

Ouvrages

- 2D3DGIS, 2019, Formation PostgreSQL - Guide pratique

Sites WEB

- Forums PostgreSQL.fr – [en ligne] Disponible sur https://forums.PostgreSQL.fr/ [consulté entre mai et septembre 2020]
- Site de PostgreSQL.org – [en ligne] Disponible sur https://www.PostgreSQL.org/docs/ [consulté entre mai et septembre 2020]
- Forum Lazarus [en ligne] – Getting last inserted ID : https://forum.lazarus.freepascal.org/index.php?topic=36162.0
- Forum developpez.com [en ligne] – Comment récupérer la liste des tables et des champs : https://www.developpez.net/forums/d66633/bases-donnees/PostgreSQL/recuperer-liste-tables-champs/
- Forum de StackOverflow.com [en ligne] – Disponible sur https://stackoverflow.com/questions/ [consulté entre mai et septembre 2020]
  - lastInsertId does not work in PostgreSQL : https://stackoverflow.com/questions/10492566/lastinsertid-does-not-work-in-PostgreSQL
  - INSERT INTO … FROM SELECT … RETURNING id mappings : https://stackoverflow.com/questions/29256888/insert-into-from-select-returning-id-mappings
  - How to get a json object as column in PostgreSQL? https://stackoverflow.com/questions/39945308/how-to-get-a-json-object-as-column-in-PostgreSQL
  - PostGIS - convert multipolygon to single polygon : https://stackoverflow.com/questions/21719941/postgis-convert-multipolygon-to-single-polygon
  - postgres default timezone : https://stackoverflow.com/questions/6663765/postgres-default-timezone

Réalisation du plugin QGIS / Elaboration des scripts python

Ouvrages

- LAYEC Alain, 2019-2020. Programmation PYTHON Niveau I – Guide de formation
- LAYEC Alain, 2019-2020. Programmation PyQt Niveau I – Guide de formation
- LAYEC Alain, 2019-2020. Programmation PyQGis Niveau I – Guide de formation

Sites WEB

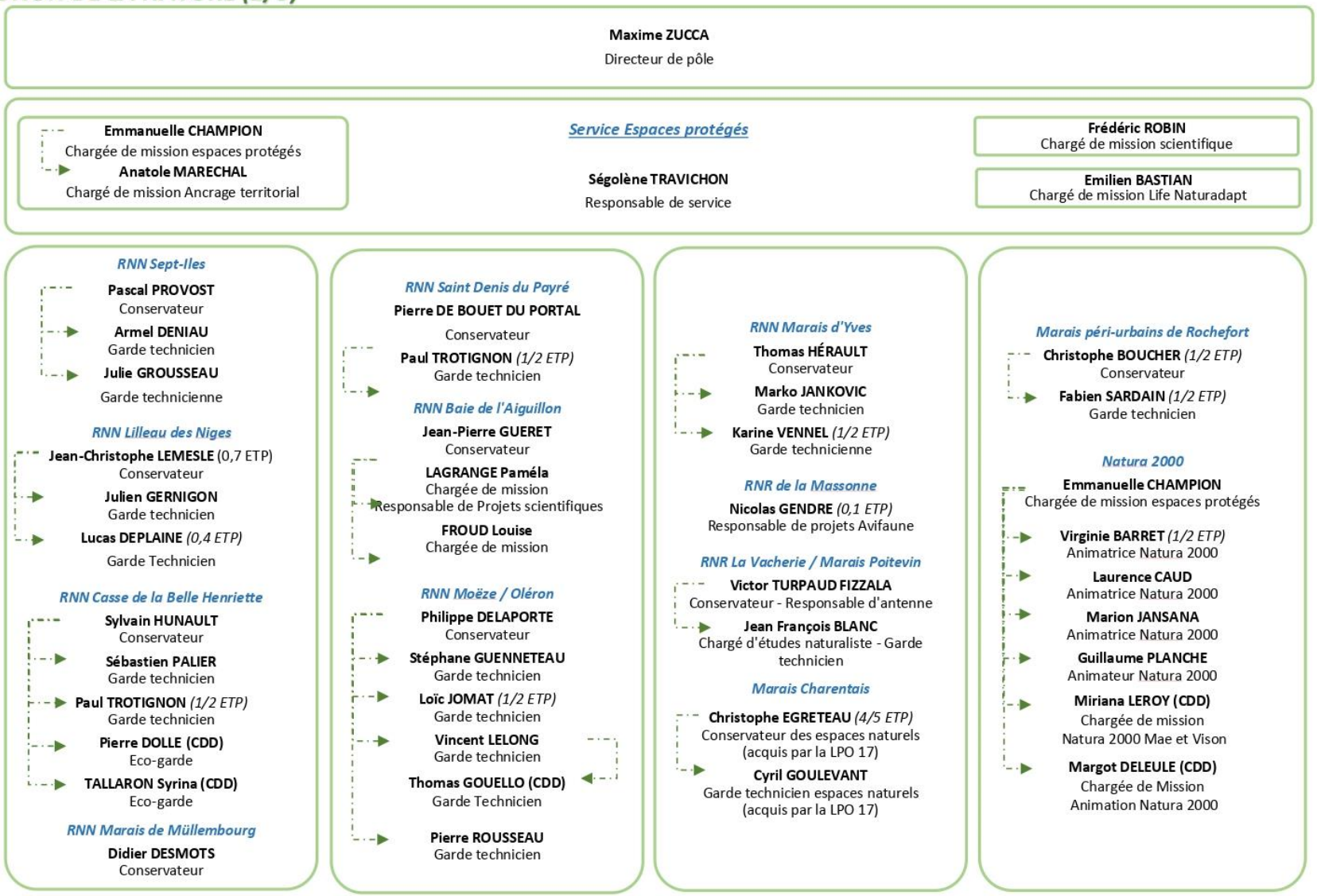
- Documentation PyQt – [en ligne] Disponible sur https://doc.qt.io/qt-5/index.html [consulté entre mai et septembre 2020]
  - QSqlQuery Class : https://doc.qt.io/qt-5/qsqlquery.html
  - Executing SQL Statements : https://doc.qt.io/qt-5/sql-sqlstatements.html
  - TableView : https://forum.qt.io/topic/89942/sql-table-view-removed-element-stays-on-the-view, https://forum.qt.io/topic/91333/update-qtableview-after-remove-a-row/6, https://doc.qt.io/qt-5/qsqltablemodel.html#setFilter
- Forum OpenClassRoom (Python) – [en ligne]
  Disponible sur https://openclassrooms.com/forum/categorie/langage-python [consulté entre mai et septembre 2020]
- Forum developpez.com [en ligne] :
  - Conversion fichier .ui en .py (Python 3) : https://www.developpez.net/forums/d1054389/autres-langages/python/gui/pyqt/convertir-fichier-ui-py/
  - ImportError: No module named PyQt4: https://www.developpez.net/forums/d586795-2/autres-langages/python/gui/pyqt/importerror-no-module-named-pyqt4/#post5857476
- Forum StackOverflow.com [en ligne] – Disponible sur https://stackoverflow.com/questions/ [consulté entre mai et septembre 2020]
  - Is there a portable way to get the current username in Python? https://stackoverflow.com/questions/842059/is-there-a-portable-way-to-get-the-current-username-in-python
  - Query to return Count using QtSql : https://stackoverflow.com/questions/24783926/query-to-return-count-using-qtsql

TABLE DES ANNEXES

ANNEXE I : ORGANIGRAMME DE LA LPO (SEP) 72
ANNEXE II : CAHIER DES CHARGES DU STAGE 73
ANNEXE III : ORGANIGRAMME DES TACHES (SADT) – NIVEAUX 0, 1 ET 2 77
ANNEXE IV : DIAGRAMME DE GANTT PREVISIONNEL 85
ANNEXE V : DIAGRAMME DE GANTT DEFINITIF 86
ANNEXE VI : FICHIER DE CONFIGURATION DU MODULE OCCTAX 89
ANNEXE VII : FICHIER DE CONFIGURATION GENERAL DE GEONATURE 92
ANNEXE VIII : SCRIPT SQL – INSERSETION MASSIF DE TAXONS 94
ANNEXE IX : SCRIPT SQL – INSERTION MASSIF D'UTILISATEURS DE SERENA 94
ANNEXE X : TABLEAU DES PERMISSIONS (CRUVED) 95
ANNEXE XI : LISTE DES NOMENCLATURES RAJOUTES 97
ANNEXE XII : FICHE DU JEU DE DONNEES « COMPTAGE MENSUEL COMMUN » DE LA RNMO 98
ANNEXE XIII : FICHIERS DE CONFIGURATION DU SOUS-MODULE « COMPTAGE MENSUEL COMMUN » 99
ANNEXE XIV : SCRIPT SQL – CREATION D'UNE VUE POUR LE MODULE « EXPORT » 104
ANNEXE XV : MODELE CONCEPTUEL DE LA BASE DE DONNEES DE GEONATURE 105
ANNEXE XVI : SCRIPT SQL – INSERTION DES REFERENTIELS GEOGRAPHIQUES 107
ANNEXE XVII : SCRIPT PYTHON – INSERTION DES DONNEES D'OBSERVATIONS ALEATOIRES POUR LA STEPRO 110
ANNEXE XVIII : SCRIPT PYTHON – INSERTION DES DONNEES HABITATS POUR LA RNMO 114
ANNEXE XIX : SCRIPT PYTHON – INSERTION DES DONNEES « COMPTAGE MENSUEL COMMUN » DE LA RNMO DANS LE MODULE MONITORING 118
ANNEXE XX : MAQUETTE DE L'INTERFACE HOMME-MACHINE DU PLUGIN « GEOPIM » 124
ANNEXE XXI : SCRIPT PYTHON PRINCIPAL DU PLUGIN 137
ANNEXE XXII : SCRIPT PYTHON D'INSERTION DE DONNEES DANS OCCTAX 139
ANNEXE XXIII : GUIDE UTILISATEUR DU MODULE MONITORING 151
ANNEXE XXIV : GUIDE UTILISATEUR DU PLUGIN « GEOPIM » 162
ANNEXE XXV : FICHES RECAPITULATIVES DE TRAVAIL 175

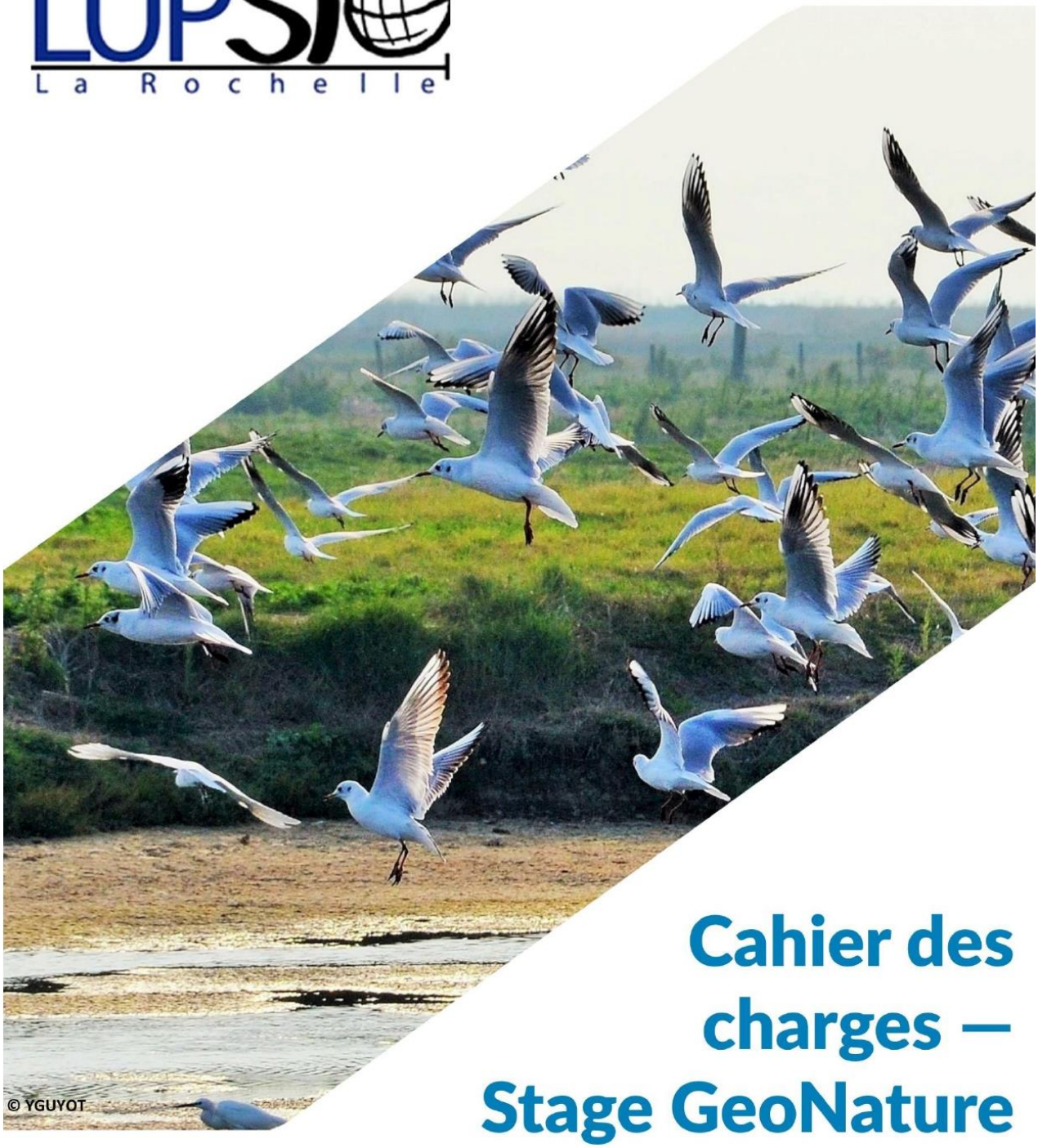
ANNEXE I : ORGANIGRAMME DE LA LPO (SEP)

PÔLE PROTECTION DE LA NATURE (2/3)



ANNEXE II : CAHIER DES CHARGES DU STAGE

LUPSIC La Rochelle



Cahier des charges — Stage GeoNature

Edition du 12 Juin 2020

LPO — AGIR pour la BIODIVERSITÉ — GEO Nature — Réserves Naturelles DE FRANCE

I/ Le contexte de la commande

Les données historiques des Réserves Naturelles gérées par la LPO étaient saisies sur un logiciel nommé **SERENA**.

Cet outil est, comme son anagramme l'indique, un **S**ystème d'**E**change de données pour les **R**éseaux d'**E**spaces **NA**turels développé par Réserves Naturelles de France et permettait la saisie de donnée naturalistes selon des protocoles de suivis. La base était gérée sous Access et le logiciel permettait la liaison entre celle-ci et l'utilisateur, auquel on affichait directement un formulaire de saisie.

Nouvel outil open-source repris en 2017 par les Parcs Nationaux français, **GeoNature** est une application permettant de regrouper l'ensemble des données provenant des protocoles Faune et Flore, de saisir dans différents protocoles et de consulter l'ensemble de ces données dans une application de synthèse.

1) La commande

Le logiciel SERENA devenu trop obsolète et inadapté pour la gestion de données géospatialisées, Frédéric ROBIN en charge de la gestion des Réserves Naturelles de la LPO souhaite intégrer les anciennes et nouvelles données récoltées avec l'outil GéoNature. La commande traduit donc d'une intégration et adaptation des données saisies sur SERENA vers GeoNature, le tout via l'implémentation et configuration de « modules » propres à celui-ci. Enfin, un nettoyage des données sera à effectuer avant import qui se fera via un plugin QGIS. En complément, l'intégration et configuration de modules complémentaires serait apprécié (atlas, mobile, dashboard…).

2) Les intervenants

La LPO (Ligue pour la Protection des Oiseaux) sera représenté par Frédéric ROBIN, coordinateur des réserves naturelles gérées par la LPO, et maître du stage, dont voici les coordonnées :

> Nom de l'organisme : **LPO, Ligue pour la Protection des Oiseaux - BirdLife France**
> Contact : **Frédéric ROBIN**
> Type d'activité **: Association de protection de l'environnement**
> Adresse : **Fonderies Royales – CS 90263 – 17305 Rochefort Cedex**
> Adresses mail : **frederic.robin@lpo.fr**

Le travail sera réalisé par Maxime TOMA de la Licence Professionnelle Systèmes d'Information Géographique de La Rochelle (LUPSIG) dans le cadre d'un stage professionnel du 25 mai au 11 septembre 2020. Les coordonnées de l'étudiant sont les suivants :

> Stagiaire : **Maxime Toma**
> Adresses mail : **maxime.toma@gmail.com, maxime.toma@lpo.fr**

II/ Les objectifs du travail et résultats attendus

L'objectif principal du projet est de :

Mettre en place l'outil GéoNature pour la gestion et le suivi de protocoles des Réserves Naturelles administrées par la LPO.

Les livrables attendus par le commanditaire sont les suivants :

- Une **intégration des données historiques** en adaptant les champs aux composantes de GeoNature (Utilisateurs, Taxons, Sites, Relevés…).
- Création d'un **plugin QGIS** configuré pour l'import de données massives pour les différents modules intégrés.
- Un **Guide d'administration personnalisé de l'outil** présentant tous ses aspects de gestion de GéoNature de la part de l'admin.
- Des **Guides d'utilisation** pour les différents protocoles et réserves selon les besoins.
- Une **Personnalisation de l'interface graphique** de GéoNature et **une configuration ses modules**.
- Un **rapport de stage** remis par l'étudiant pour l'université.
- Une **soutenance de stage** (lieu et date à définir).

Les différents outils utilisés seront décrits dans les « Méthodes de travail ».

III/ Méthodes de travail

Pour réaliser ce travail, la LPO fournira les documents/données suivant.e.s :

- Données bancarisées sur fichier Excel, Shape et Access des suivis effectués (base SERENA, .shp, .mdb, .xls, .map)
- Données complémentaires SIG (shape, référentiels cartographiques, etc…)
- Les accès pour l'administration de l'outil GeoNature

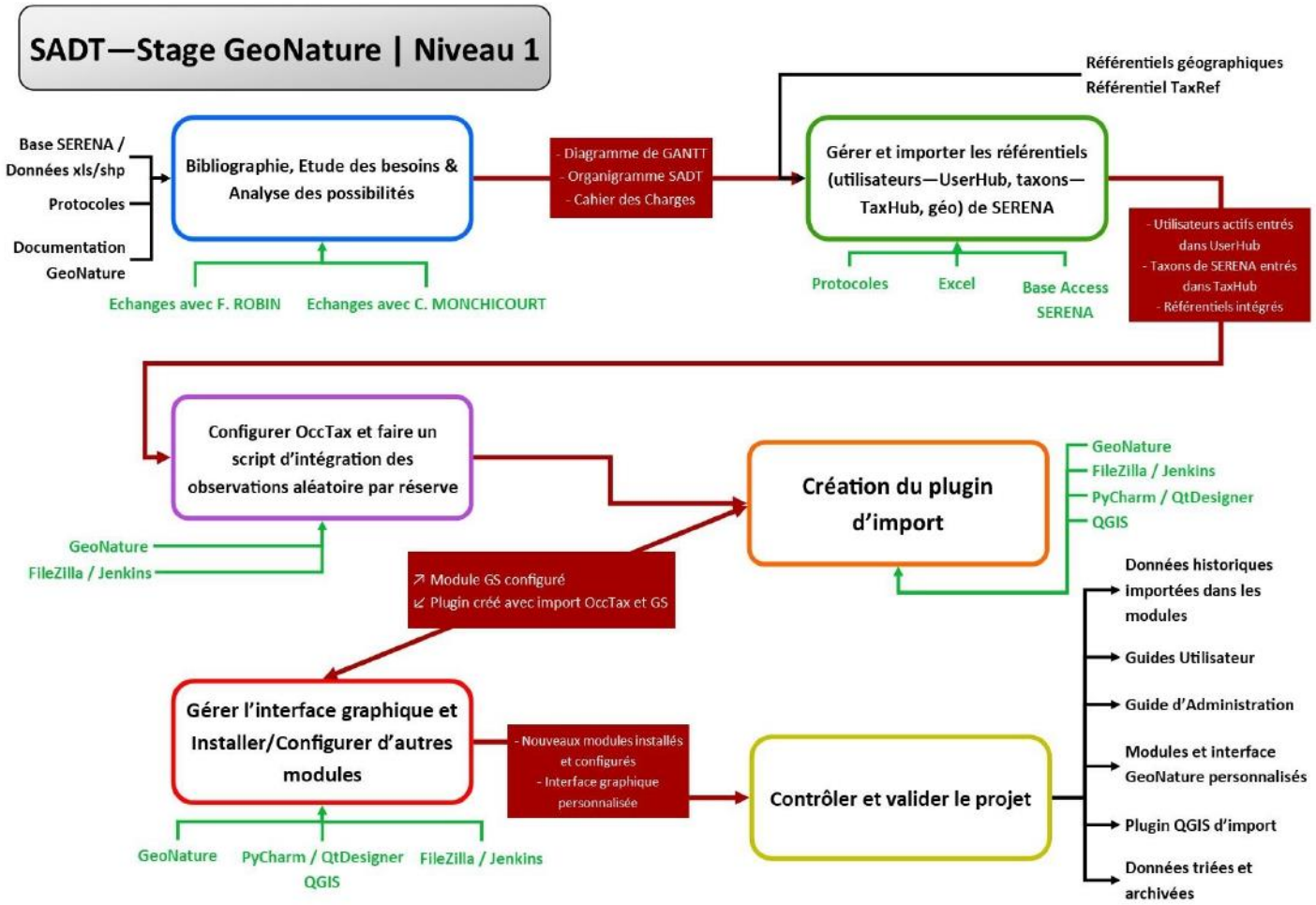
Feront l'objet d'une validation écrite (mail) auprès du maître de stage :

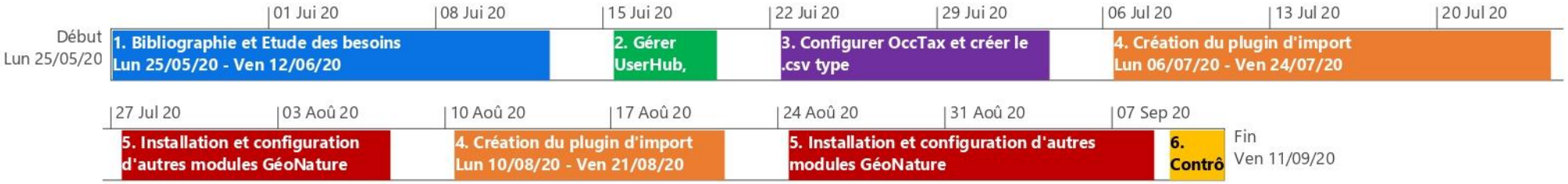
- **Validation du présent Cahier des Charges** ci-présent (fin Juin au plus tard).
- **Validation de l'IHM du plugin QGIS** (début juillet)

Liste des outils utilisés pendant le projet :

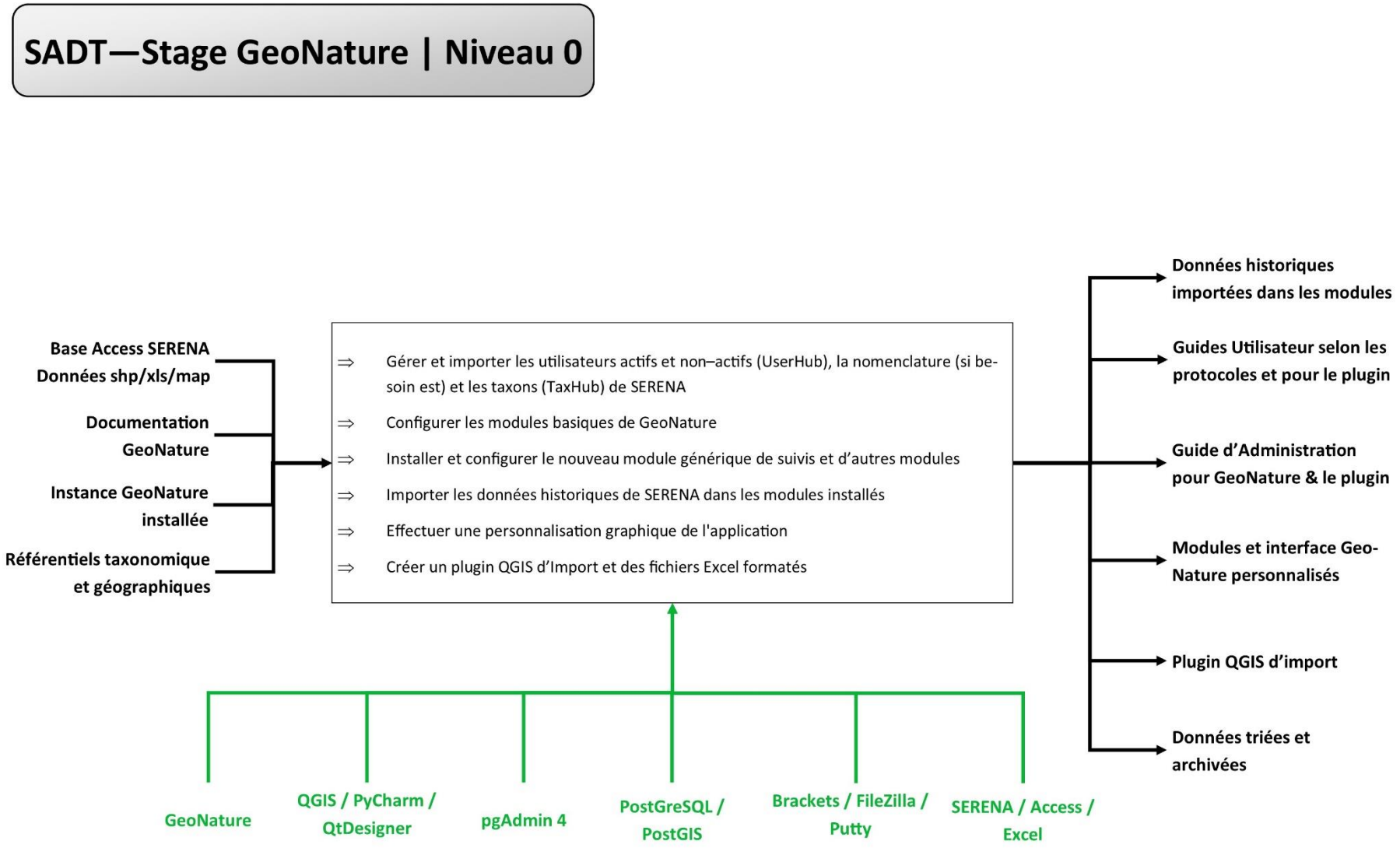
- **PostGreSQL** et **pgAdmin4**, pour l'administration de la base de données.
- **PostGIS**, extension de PostGreSQL permettant l'hébergement et le traitement de données portant une géométrie.
- **GéoNature**, outil open-source développé par les Parcs Nationaux français pour la saisie, la gestion, la synthèse et la diffusion d'observations faune et flore.
- **PyCharm, PyQT4, QTCreator et PyQGIS,** pour la création du plugin sur QGIS 2.18.
- **Jenkins,** serveur d'automatisation open source qui permet l'envoi de commandes shells de manière fiable et contrôlée.
- **FileZilla,** client FTP, FTPS et SFTP permettant la modification de fichiers hébergés.
- **Word, PowerPoint,** pour la rédaction des guides.
- **Excel,** pour le nettoyage des données et pour faire les .csv formatés

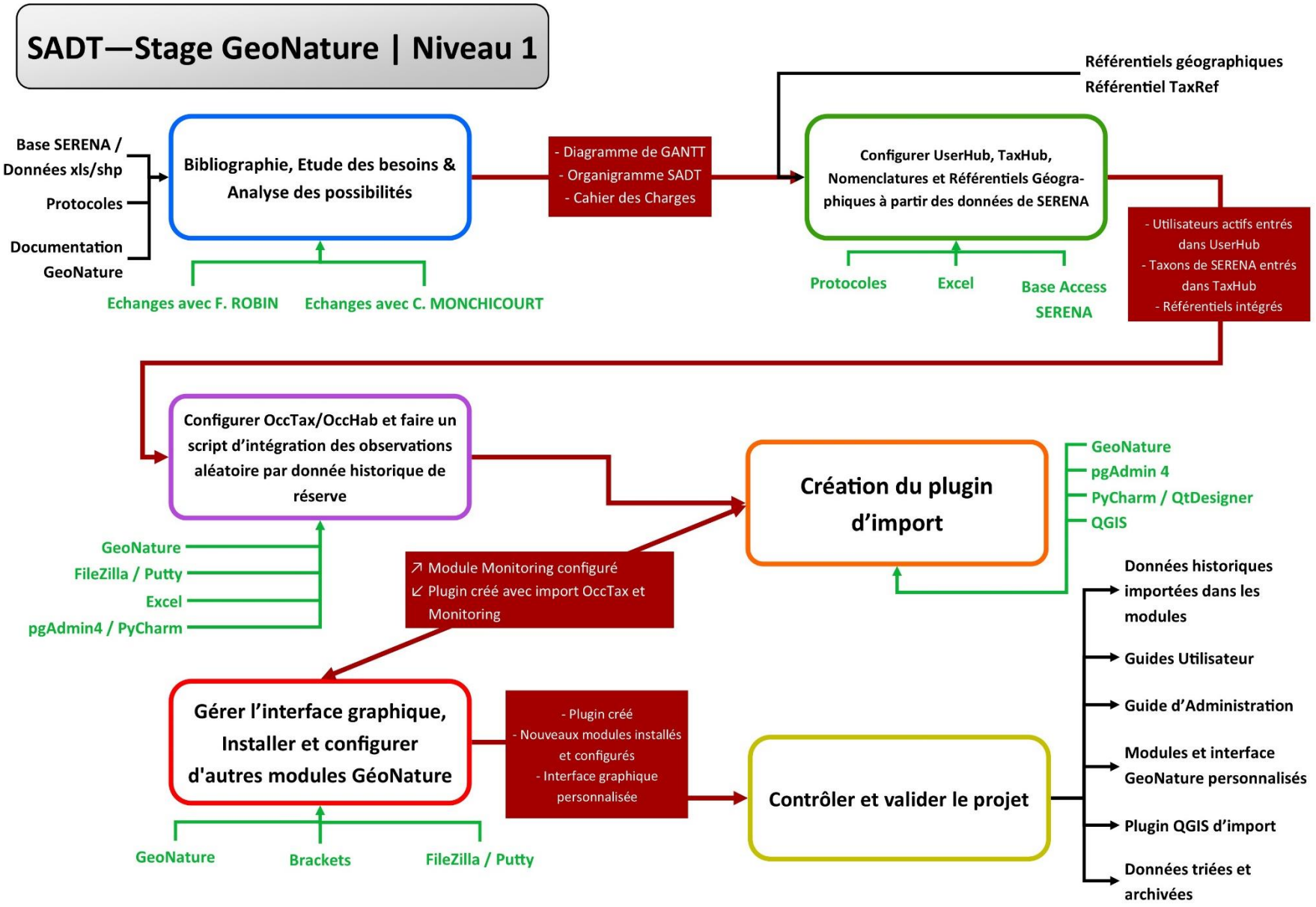
Planning prévisionnel et organigramme des tâches à réaliser :

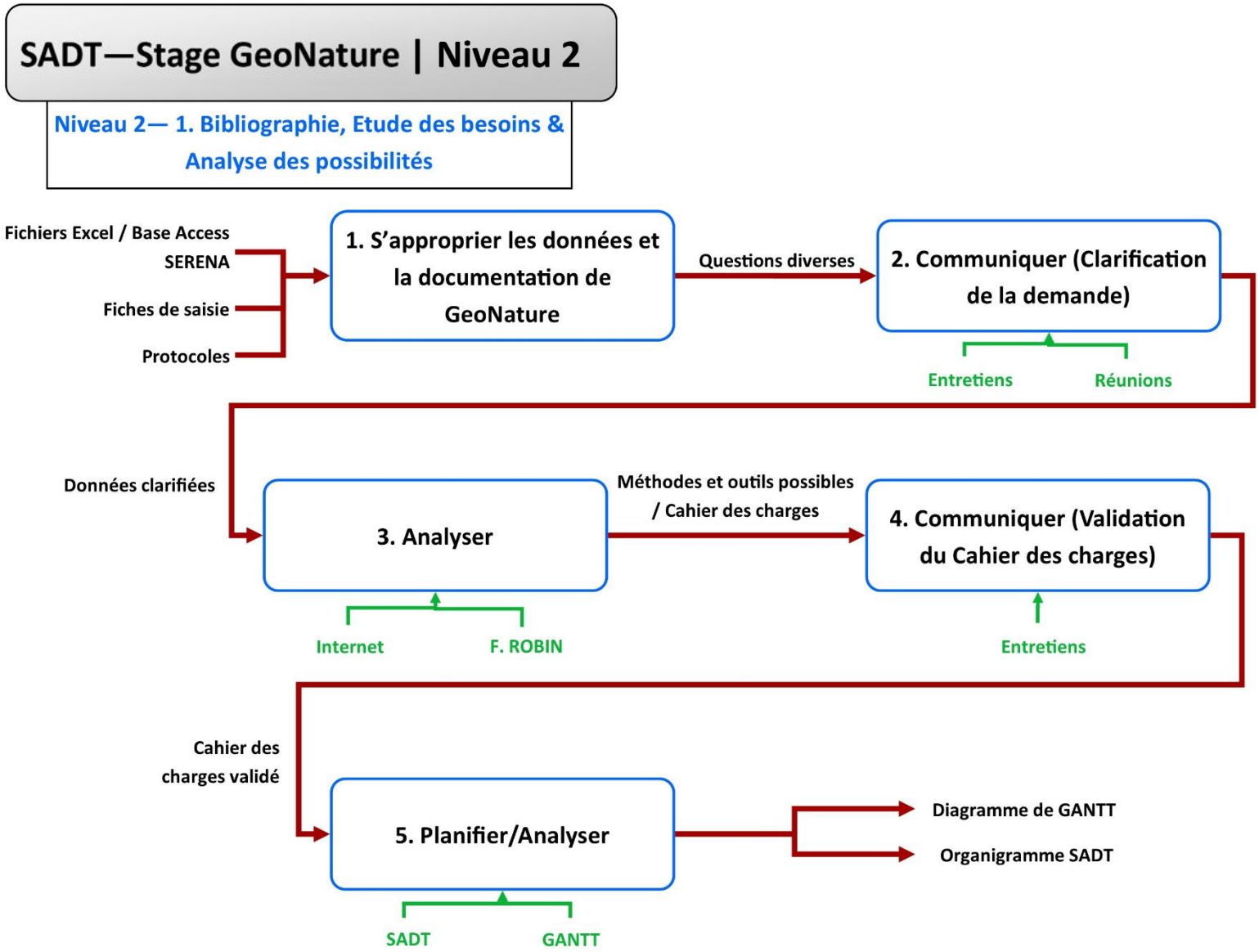


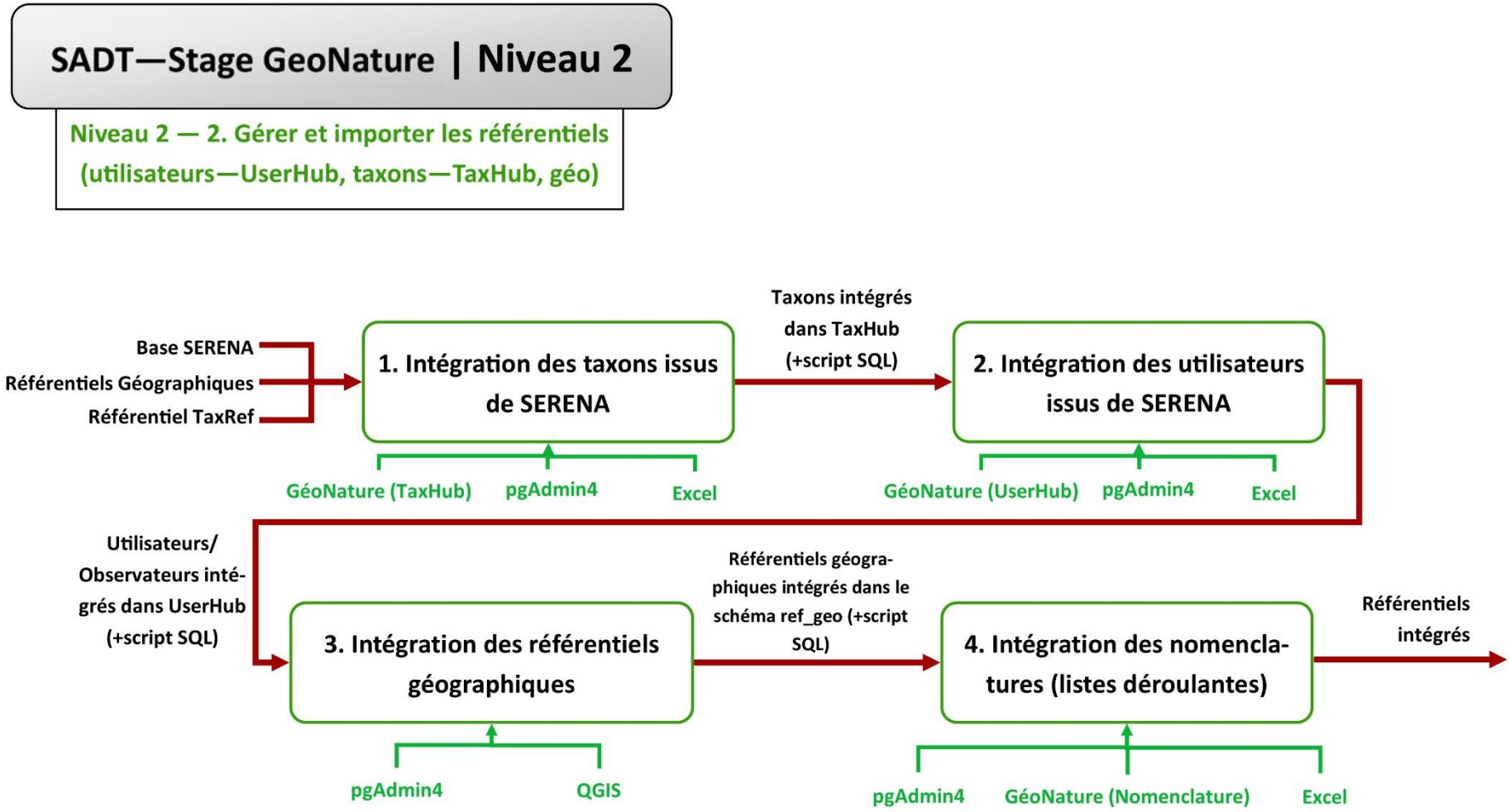


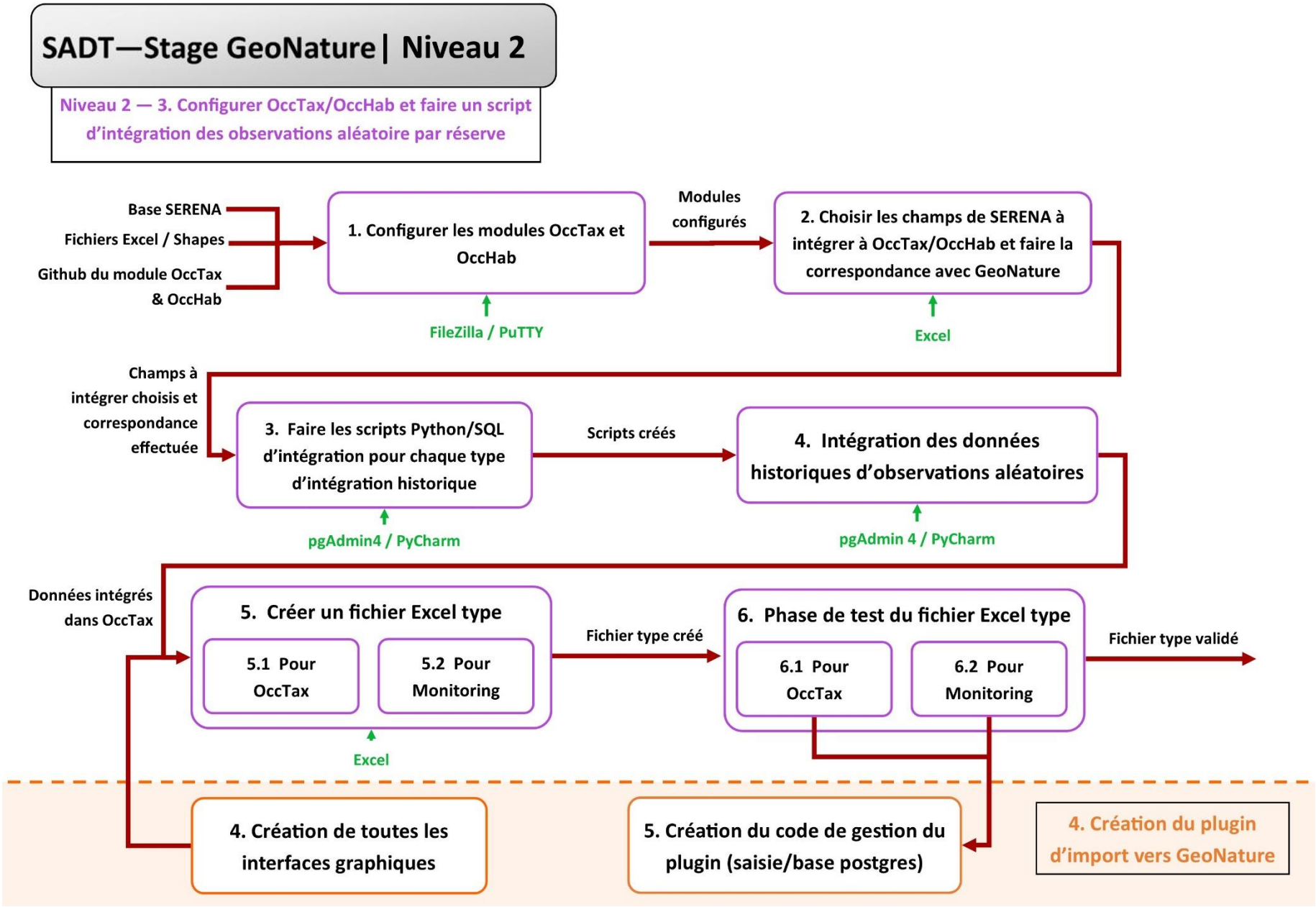
ANNEXE III : ORGANIGRAMME DES TACHES (SADT) – NIVEAUX 0, 1 ET 2

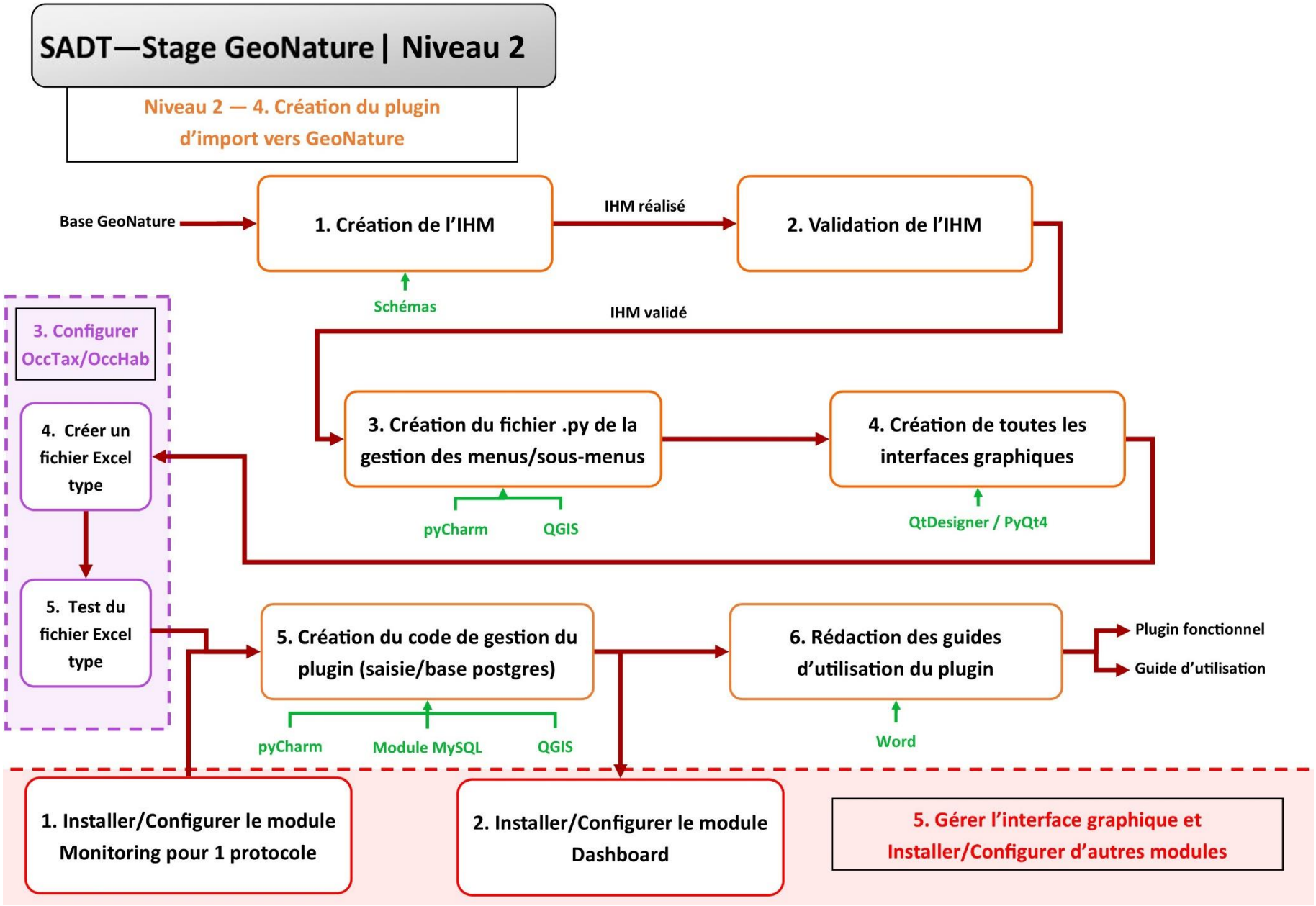


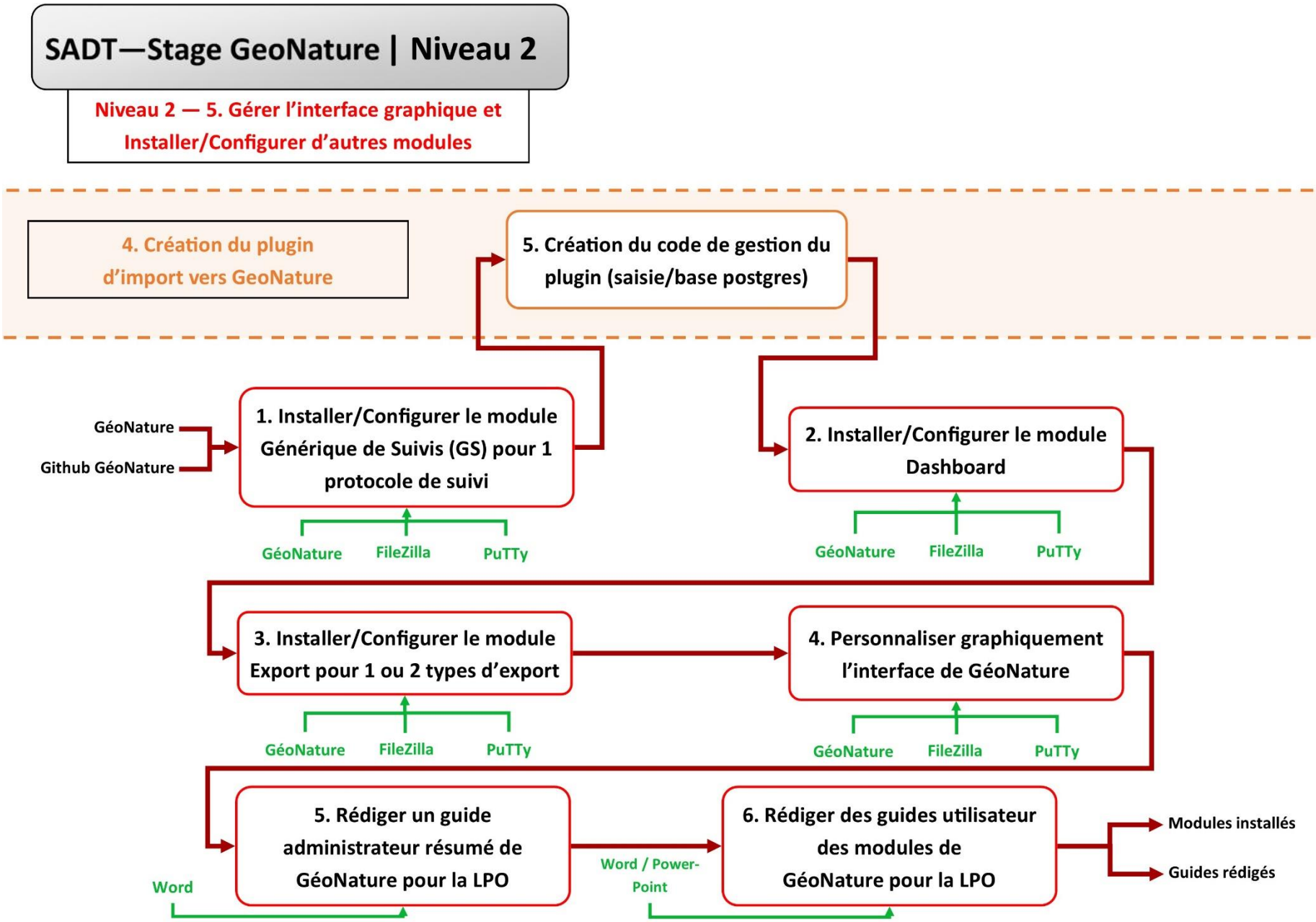

SADT—Stage GeoNature | Niveau 1
Base SERENA / Données xls/shp
Protocoles
Documentation GeoNature
Bibliographie, Etude des besoins & Analyse des possibilités
Echanges avec F. ROBIN
Echanges avec C. MONCHICOURT
- Diagramme de GANTT
- Organigramme SADT
- Cahier des Charges
Référentiels géographiques
Référentiel TaxRef
Configurer UserHub, TaxHub, Nomenclatures et Référentiels Géographiques à partir des données de SERENA
Protocoles
Excel
Base Access SERENA
- Utilisateurs actifs entrés dans UserHub
- Taxons de SERENA entrés dans TaxHub
- Référentiels intégrés
Configurer OccTax/OccHab et faire un script d'intégration des observations aléatoire par donnée historique de réserve
GeoNature
FileZilla / Putty
Excel
pgAdmin4 / PyCharm
↗ Module Monitoring configuré
↙ Plugin créé avec import OccTax et Monitoring
Création du plugin d'import
GeoNature
pgAdmin 4
PyCharm / QtDesigner
QGIS
Gérer l'interface graphique, Installer et configurer d'autres modules GéoNature
GeoNature
Brackets
FileZilla / Putty
- Plugin créé
- Nouveaux modules installés et configurés
- Interface graphique personnalisée
Contrôler et valider le projet
Données historiques importées dans les modules
Guides Utilisateur
Guide d'Administration
Modules et interface GeoNature personnalisés
Plugin QGIS d'import
Données triées et archivées

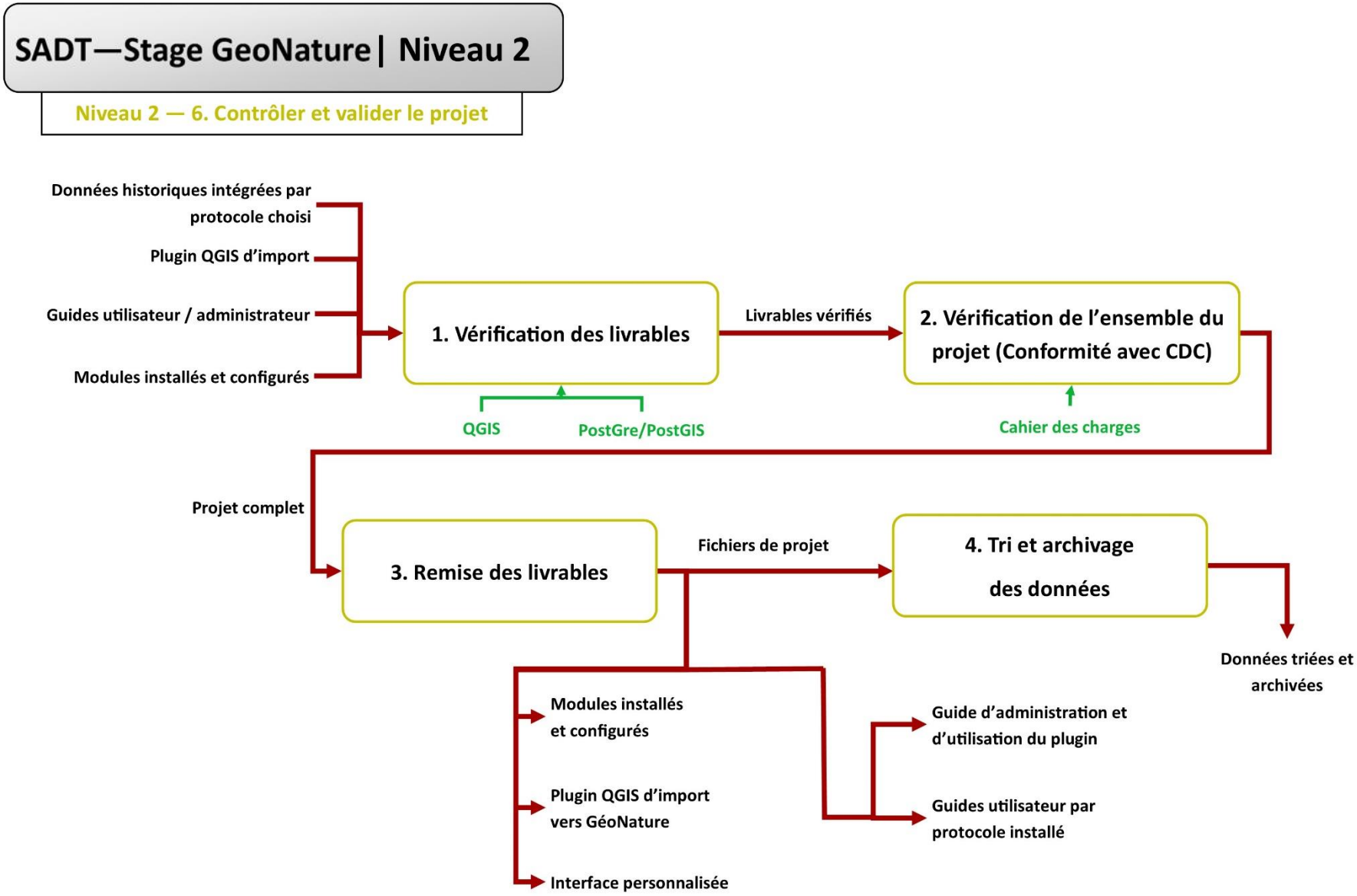

SADT—Stage GeoNature | Niveau 2
Niveau 2— 1. Bibliographie, Etude des besoins & Analyse des possibilités
Fichiers Excel / Base Access
SERENA
Fiches de saisie
Protocoles
1. S'approprier les données et la documentation de GeoNature
Questions diverses
2. Communiquer (Clarification de la demande)
Entretiens
Réunions
Données clarifiées
3. Analyser
Internet
F. ROBIN
Méthodes et outils possibles / Cahier des charges
4. Communiquer (Validation du Cahier des charges)
Entretiens
Cahier des charges validé
5. Planifier/Analyser
SADT
GANTT
Diagramme de GANTT
Organigramme SADT


SADT—Stage GeoNature | Niveau 2
Niveau 2 — 2. Gérer et importer les référentiels (utilisateurs—UserHub, taxons—TaxHub, géo)
Base SERENA
Référentiels Géographiques
Référentiel TaxRef
1. Intégration des taxons issus de SERENA
GéoNature (TaxHub)
pgAdmin4
Excel
Taxons intégrés dans TaxHub (+script SQL)
2. Intégration des utilisateurs issus de SERENA
GéoNature (UserHub)
pgAdmin4
Excel
Utilisateurs/ Observateurs intégrés dans UserHub (+script SQL)
3. Intégration des référentiels géographiques
pgAdmin4
QGIS
Référentiels géographiques intégrés dans le schéma ref_geo (+script SQL)
4. Intégration des nomenclatures (listes déroulantes)
pgAdmin4
GéoNature (Nomenclature)
Excel
Référentiels intégrés

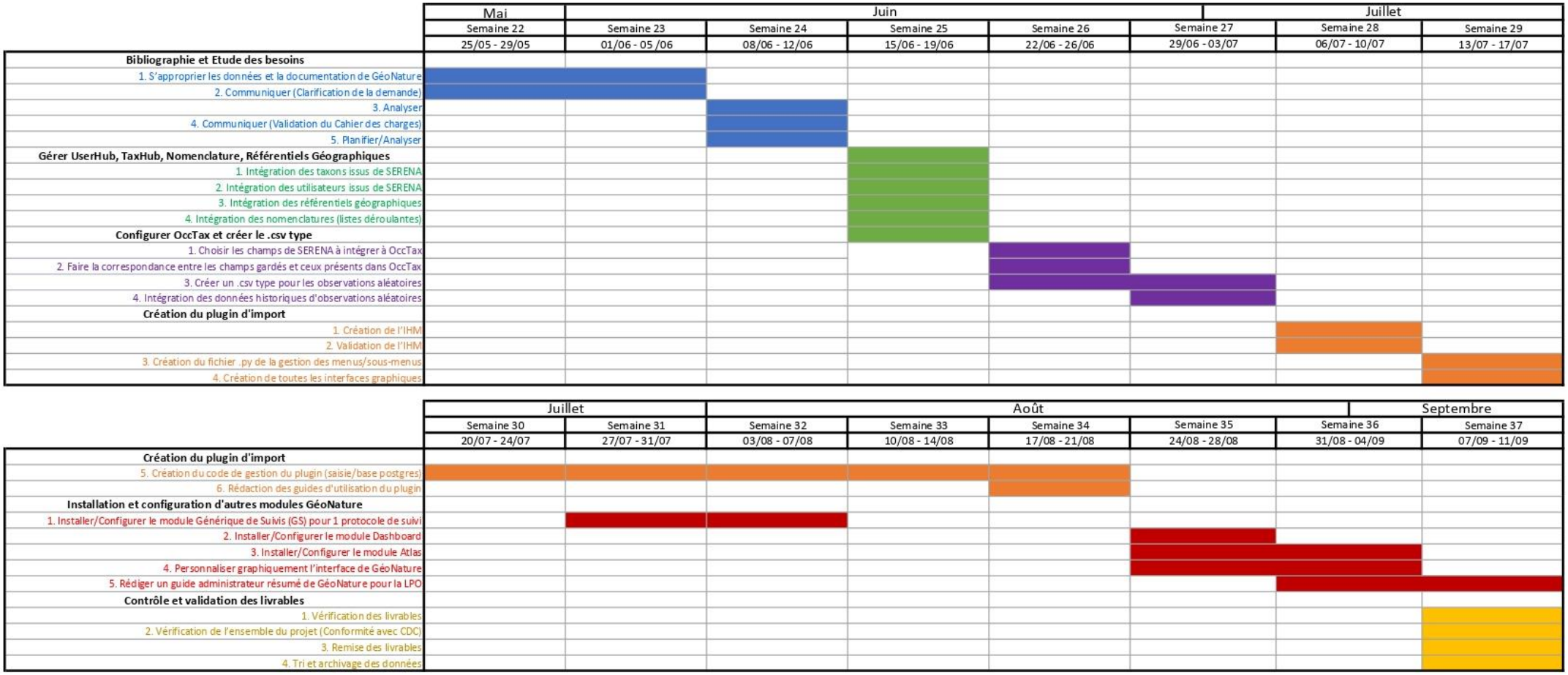

SADT—Stage GeoNature | Niveau 2
Niveau 2 — 3. Configurer OccTax/OccHab et faire un script d'intégration des observations aléatoire par réserve
Base SERENA
Fichiers Excel / Shapes
Github du module OccTax & OccHab
1. Configurer les modules OccTax et OccHab
FileZilla / PuTTY
Modules configurés
2. Choisir les champs de SERENA à intégrer à OccTax/OccHab et faire la correspondance avec GeoNature
Excel
Champs à intégrer choisis et correspondance effectuée
3. Faire les scripts Python/SQL d'intégration pour chaque type d'intégration historique
pgAdmin4 / PyCharm
Scripts créés
4. Intégration des données historiques d'observations aléatoires
pgAdmin 4 / PyCharm
Données intégrés dans OccTax
5. Créer un fichier Excel type
5.1 Pour OccTax
5.2 Pour Monitoring
Excel
Fichier type créé
6. Phase de test du fichier Excel type
6.1 Pour OccTax
6.2 Pour Monitoring
Fichier type validé
4. Création de toutes les interfaces graphiques
5. Création du code de gestion du plugin (saisie/base postgres)
4. Création du plugin d'import vers GeoNature


SADT—Stage GeoNature | Niveau 2
Niveau 2 — 4. Création du plugin d'import vers GeoNature
Base GeoNature
1. Création de l'IHM
IHM réalisé
2. Validation de l'IHM
Schémas
IHM validé
3. Configurer OccTax/OccHab
4. Créer un fichier Excel type
5. Test du fichier Excel type
3. Création du fichier .py de la gestion des menus/sous-menus
pyCharm
QGIS
4. Création de toutes les interfaces graphiques
QtDesigner / PyQt4
5. Création du code de gestion du plugin (saisie/base postgres)
pyCharm
Module MySQL
QGIS
6. Rédaction des guides d'utilisation du plugin
Word
Plugin fonctionnel
Guide d'utilisation
1. Installer/Configurer le module Monitoring pour 1 protocole
2. Installer/Configurer le module Dashboard
5. Gérer l'interface graphique et Installer/Configurer d'autres modules

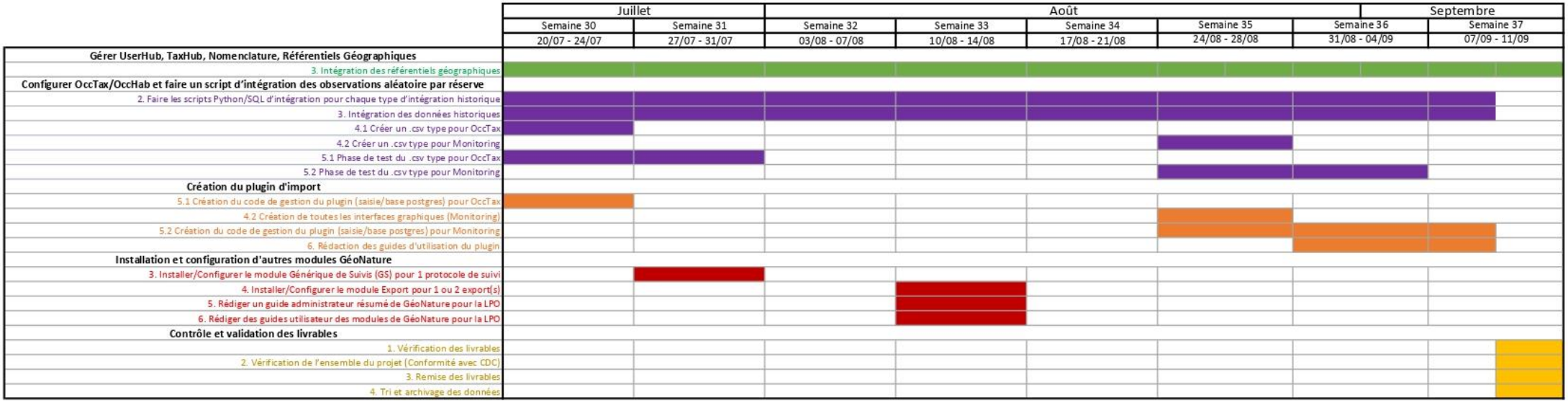

SADT—Stage GeoNature | Niveau 2
Niveau 2 — 5. Gérer l'interface graphique et Installer/Configurer d'autres modules
4. Création du plugin d'import vers GeoNature
5. Création du code de gestion du plugin (saisie/base postgres)
GéoNature
Github GéoNature
1. Installer/Configurer le module Générique de Suivis (GS) pour 1 protocole de suivi
GéoNature
FileZilla
PuTTy
2. Installer/Configurer le module Dashboard
GéoNature
FileZilla
PuTTy
3. Installer/Configurer le module Export pour 1 ou 2 types d'export
GéoNature
FileZilla
PuTTy
4. Personnaliser graphiquement l'interface de GéoNature
GéoNature
FileZilla
PuTTy
5. Rédiger un guide administrateur résumé de GéoNature pour la LPO
Word
Word / Power-Point
6. Rédiger des guides utilisateur des modules de GéoNature pour la LPO
Modules installés
Guides rédigés


SADT—Stage GeoNature | Niveau 2
Niveau 2 — 6. Contrôler et valider le projet
Données historiques intégrées par protocole choisi
Plugin QGIS d'import
Guides utilisateur / administrateur
Modules installés et configurés
1. Vérification des livrables
QGIS
PostGre/PostGIS
Livrables vérifiés
2. Vérification de l'ensemble du projet (Conformité avec CDC)
Cahier des charges
Projet complet
3. Remise des livrables
Fichiers de projet
4. Tri et archivage des données
Données triées et archivées
Modules installés et configurés
Plugin QGIS d'import vers GéoNature
Interface personnalisée
Guide d'administration et d'utilisation du plugin
Guides utilisateur par protocole installé

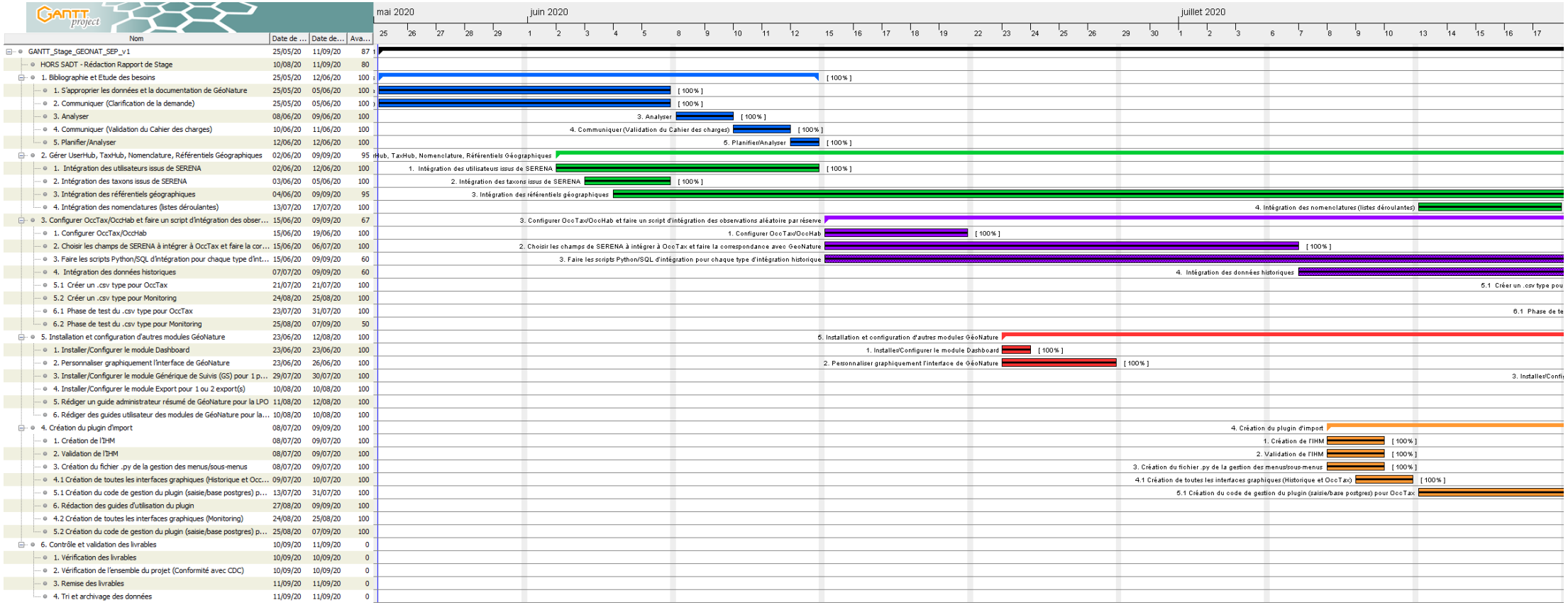
ANNEXE IV : DIAGRAMME DE GANTT PREVISIONNEL



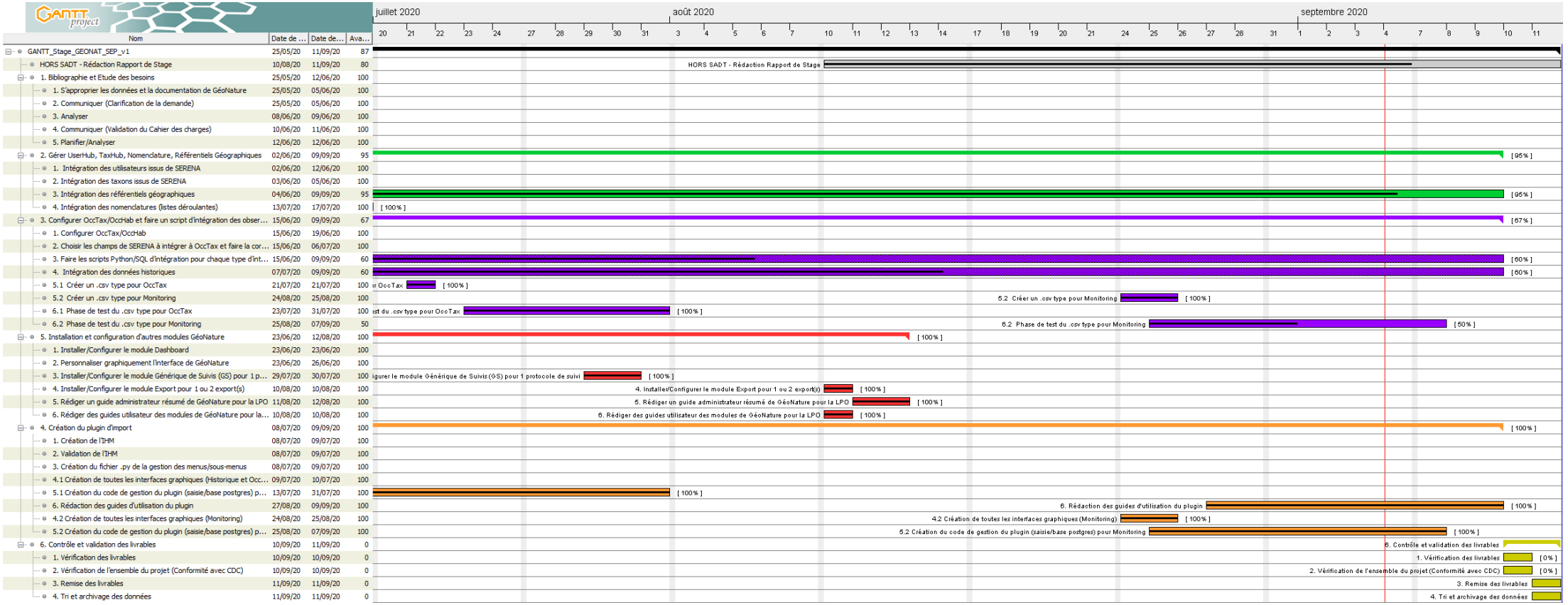
ANNEXE V : DIAGRAMME DE GANTT DEFINITIF



Version GANTT Project 1/2 :



Version GANTT Project 2/2 :



ANNEXE VI : FICHIER DE CONFIGURATION DU MODULE OCCTAX

Toutes les configurations possibles pour le module OccTax.

```
# ----------- OCCTAX AVAILABLE AND DEFAULT PARAMETER ------------
# You can override parameters defautl values in conf_gn_module.toml file
# /!\ All parameters which are not in a section (between []) must be AT THE TOP OF THE FILE

# Switch the form input in free text input (true) OR in SELECT input (false)
observers_txt = false

# id of the observator list -- utilisateurs.t_menus
id_observers_list = 1

# id of the taxon list -- taxonomie.bib_listes. Use for the taxonomy search in the occtax
forml
id_taxon_list = 100
# number of results in the autocomplete taxon search
taxon_result_number = 20

# Add a validation rule for digital proof. Must begin with a http:// OR https://
digital_proof_validator = true

# set default form date picker with the date of today
DATE_FORM_WITH_TODAY = true

# -------- MAPLIST PARAMETER ------------

# Zoom level on the map FROM which you can add point/line/polygon
releve_map_zoom_level = 6

# Columns which are default display in the list
default_maplist_columns = [
    { prop = "taxons", name = "Taxon(s)" },
    { prop = "date_min", name = "Date début", max_width = "100" },
    { prop = "observateurs", name = "Observateur(s)" },
    { prop = "dataset_name", name = "Jeu de données" }
]

# Available columns which can be add manualy by user
available_maplist_column = [
     { prop = "altitude_max", name = "altitude_max" },
    { prop = "altitude_min", name = "altitude_min" },
    { prop = "comment", name = "Commentaire" },
    { prop = "date_max", name = "Date fin" },
    { prop = "date_min", name = "Date début" },
    { prop = "id_dataset", name = "ID dataset" },
    { prop = "id_digitiser", name = "ID rédacteur" },
    { prop = "id_releve_occtax", name = "ID relevé" },
    { prop = "observateurs", name = "observateurs" },
    { prop = "taxons", name = "taxons" }
]

# Message of the list of releve
list_messages = { emptyMessage = "Aucune donnée à afficher", totalMessage = "observations au
total" }

# --------------- EXPORT PARAMETER ---------------------

# Name of the VIEW based export
export_VIEW_name = 'export_occtax_sinp'
```

```
# Name of the geometry columns of the VIEW
export_geom_columns_name = 'geom_4326'

# Name of the primary key column of the VIEW
export_id_column_name = 'permId'

# Name of the columns with observers_txt
export_observer_txt_column = 'obsId'

# SRID for the shapefile
export_srid = 4326

# Export available format (Only csv, geojson and shapefile is possible)
export_available_format = ['csv', 'geojson', 'shapefile']

# Custom message to display in the export modal
export_message = "<p> <b> Attention: </b> </br>  Vous vous apprêtez à télécharger les données
de la <b>recherche courante. </b> </p>"

# Max observations number the user is allowed to export at once
MAX_EXPORT_NUMBER = 50000

# Columns to display in the exports
export_columns =  [
  "permId",
  "statObs",
  "nomCite",
  "dateDebut",
  "dateFin",
  "heureDebut",
  "heureFin",
  "altMax",
  "altMin",
  "cdNom",
  "cdRef",
  "dateDet",
  "comment",
  "dSPublique",
  "statSource",
  "idOrigine",
  "jddId",
  "refBiblio",
  "obsMeth",
  "ocEtatBio",
  "ocNat",
  "ocSex",
  "ocStade",
  "ocBiogeo",
  "ocStatBio",
  "preuveOui",
  "ocMethDet",
  "preuvNum",
  "preuvNoNum",
  "obsCtx",
  "permIdGrp",
  "methGrp",
  "typGrp",
  "denbrMax",
  "denbrMin",
  "objDenbr",
  "typDenbr",
  "obsId",
  "obsNomOrg",
```

```
  "detId",
  "detNomOrg",
  "orgGestDat",
  "WKT",
  "natObjGeo"
 ]

# afficher l'outil pour pointer un marker à partir des coordonnées X/Y
ENABLE_GPS_TOOL = true
# Activer l'outil leaflet filelayer qui permet de charger des fichier GPS, GeoJSON, et KML
ENABLE_UPLOAD_TOOL = true


# ------------- FORM PARAMETER ---------------

# Allow to hide OR display some fields. If hidden, database default value is used
[form_fields]
    date_min = true
    date_max = true
    hour_min = true
    hour_max = true
    altitude_min = true
    altitude_max = true
    obs_technique = true
    group_type = true
    comment_releve = true
    obs_method = true
    bio_condition = true
    bio_status = true
    naturalness = true
    exist_proof = true
    observation_status = true
    diffusion_level = false
    blurring = false
    determiner = true
    determination_method = true
    sample_number_proof = true
    digital_proof = true
    non_digital_proof = true
    source_status = false
    comment_occ = true
    life_stage = true
    sex = true
    obj_count = true
    type_count = true
    count_min = true
    count_max = true
```

ANNEXE VII : FICHIER DE CONFIGURATION GENERAL DE GEONATURE

Les données surlignées en noir sont des données sensibles et ont été remplacés.

```
##########################################
# GeoNature backend global configuration file
##########################################

# Database
SQLALCHEMY_DATABASE_URI = "PostgreSQL://geonatureadmin:[redacted]@localhost:5432/geonature2db"
URL_APPLICATION = 'http://[redacted]/geonature'
API_ENDPOINT = 'http://[redacted]/geonature/api'
API_TAXHUB = 'http://[redacted]/taxhub/api'

# Remplacer par une clé alétoire complexe
SECRET_KEY = 'super secret key'

# Code EPSG des géométries dans la BDD
LOCAL_SRID = '2154'

# Langue principale par défaut
DEFAULT_LANGUAGE='fr'


#################################################################################
###### Si vous souhaitez surcoucher certains paramètres,
###### compléter les sections ci-dessous à partir du modèle default_config.toml.example
#################################################################################

appName = 'LPO - Espaces Naturels Protégés'

# Configuration liée aux ID de BDD
[BDD]

# Configuration générale du frontend
[FRONTEND]

# Configuration de la Synthese
[SYNTHESE]
    AREA_FILTERS = [
        { label = "Communes", id_type = 25 },
        { label = "Réserves Naturelles", id_type = [5,6] },
    ]

[USERSHUB]
    URL_USERSHUB = 'http://[redacted]/usershub' # sans slash final
    # Administrateur de mon application
    ADMIN_APPLICATION_LOGIN = 'admin'
    ADMIN_APPLICATION_PASSWORD = 'admin'

[ACCOUNT_MANAGEMENT]
    ENABLE_USER_MANAGEMENT = true

[MAIL_CONFIG]
    MAIL_SERVER = "smtp.office365.com"
    MAIL_PORT = 587
    MAIL_USE_TLS = true
    MAIL_USE_SSL = false
    MAIL_USERNAME = "[redacted]@lpo.fr"
    MAIL_PASSWORD = "[redacted]"
    MAIL_DEFAULT_SENDER = "[redacted]@lpo.fr"
    MAIL_ASCII_ATTACHMENTS = false
```

```
# Configuration cartographique
[MAPCONFIG]
    # Coordonnées par défaut du centre de la carte à afficher
    # Attention : les coordonnées sont au format [Y, X]
    # Cf Leaflet configuration (https://leafletjs.com/reference-1.4.0.html#latlng-l-latlng)
    CENTER = [46.97747407252, -1.58492983016]

    # Zoom par défaut
    ZOOM_LEVEL = 7

[[MAPCONFIG.BASEMAP]]
    attributions = "GoogleSatellite"
    name = "google"
    url = "//{s}.google.com/vt/lyrs=s&x={x}&y={y}&z={z}"
    subdomains = ["mt0", "mt1", "mt2", "mt3"]
[[MAPCONFIG.BASEMAP]]
    name = "OpenstreetMap"
    url = "//{s}.tile.openstreetmap.fr/hot/{z}/{x}/{y}.png"
    attributions = "OSM contributors"
```

ANNEXE VIII : SCRIPT SQL – INSERSETION MASSIF DE TAXONS

```
-- ATTENTION à utiliser que lors de la première execution ou si on veut renouveler le
tout
DELETE FROM taxonomie.cor_nom_liste;
DELETE FROM taxonomie.bib_noms;

CREATE TABLE IF NOT EXISTS taxonomie.temp_tax (id SERIAL NOT NULL, cd_nom INTEGER)
CREATE TABLE IF NOT EXISTS taxonomie.temp_tax_groupby (id SERIAL NOT NULL, cd_nom
INTEGER)

INSERT INTO taxonomie.temp_tax_groupby (cd_nom)
SELECT cd_nom FROM taxonomie.temp_tax
GROUP BY cd_nom

-- Insertion depuis TaxRef vers bib_noms (taxons spécifiques à l'organisme) avec la
condition qu'ils soient dans la liste des taxons importés de SERENA
INSERT INTO taxonomie.bib_noms(cd_nom,cd_ref,nom_francais)
SELECT t.cd_nom, t.cd_ref, t.nom_vern
FROM taxonomie.taxref t
INNER JOIN taxonomie.temp_tax_groupby temp ON temp.cd_nom = t.cd_nom
WHERE t.cd_nom NOT IN (SELECT DISTINCT cd_nom FROM taxonomie.bib_noms);

-- Insertion dans une liste depuis bib_nom (SELECT 100 désigne l'id de la liste)
INSERT INTO taxonomie.cor_nom_liste (id_liste,id_nom)
SELECT 100, n.id_nom FROM taxonomie.bib_noms n
WHERE n.id_nom NOT IN (SELECT DISTINCT id_nom FROM taxonomie.cor_nom_liste);
```

ANNEXE IX : SCRIPT SQL – INSERTION MASSIF D'UTILISATEURS DE SERENA

```
-- Requête pour SERENA
SELECT DISTINCT OBSE_OBSV_ID FROM RNF_OBSE
INNER JOIN RNF_SRCE ON RNF_OBSE.OBSE_OBSV_ID = RNF_SRCE.SRCE_ID;

-- Création de l'organisme 16 dans UsersHub: Contributeur extérieur

-- Insertion des utilisateurs issus de SERENA dans la table t_roles
INSERT INTO utilisateurs.t_roles (groupe, nom_role, prenom_role, identifiant,
id_organisme)
SELECT 'false', UPPER(nom), prenom, LOWER(CONCAT(prenom,'.',nom)), 16
FROM gn_imports.serena_users;

-- Liaison avec le groupe 60: Contributeur extérieur
INSERT INTO utilisateurs.cor_roles (id_role_groupe, id_role_utilisateur)
SELECT 60, id_role FROM utilisateurs.t_roles
WHERE id_organisme = 16;

-- Suppression des caractères spéciaux dans l'identifiant
UPDATE utilisateurs.t_roles
SET identifiant =
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(identifiant, 'é',
'e'),'à','a'),'ï','i'),'î','i'),'ê','e'),'ë','e'),'è','e'), ' ', ''),'..','.')
WHERE id_organisme = 16;
```

ANNEXE X : TABLEAU DES PERMISSIONS (CRUVED)

	GeoNature						Admin						Metadonnées						Synthese						OccTax					
USER/GROUPE	C	R	U	V	E	D	C	R	U	V	E	D	C	R	U	V	E	D	C	R	U	V	E	D	C	R	U	V	E	D
Grp_admin	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Groupe_RNMO_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_RNMO_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Groupe_RNMY_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_RNMY_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Groupe_RNBH_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_RNBH_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Groupe_RNMULL_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_RNMULL_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Groupe_RNSDP_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_RNSDP_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Groupe_RNLDN_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_RNLDN_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Groupe_RNBA_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_RNBA_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Groupe_RNVACH_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_RNVACH_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Groupe_RN7ILES_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_RN7ILES_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Groupe_ACQUI_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_ACQUI_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Groupe_STEPRO_p	0	3	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	3	0	0	2	0	2	2	2	0	2	2
Groupe_STEPRO_t	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	1	0	1	2	1	0	1	1
Grp_LPO	0	3	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	3	0	0	0	0	2	2	1	0	2	1
Groupe_Contributeur_exterieur																														

Aucune donnée	0
Mes données	1
Données de mon organisme	2
Toutes les données	3
Aucune permission	

	OccHab						Validation						Validation						Monitoring						Monitoring - Comptage Mensuel Commun RNMO					
USER/GROUPE	C	R	U	V	E	D	C	R	U	V	E	D	C	R	U	V	E	D	C	R	U	V	E	D	C	R	U	V	E	D
Grp_admin	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Groupe_RNMO_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	3	3	3	3	3	3
Groupe_RNMO_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	2	3	1	0	1	1
Groupe_RNMY_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNMY_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNBH_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNBH_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNMULL_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNMULL_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNSDP_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNSDP_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNLDN_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNLDN_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNBA_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNBA_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNVACH_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RNVACH_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RN7ILES_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_RN7ILES_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_ACQUI_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_ACQUI_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_STEPRO_p	0	3	0	0	3	0	0	2	0	2	0	0	0	2	0	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Groupe_STEPRO_t	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Grp_LPO	0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	3	0	0	0	0
Groupe_Contributeur_exterieur																														

ANNEXE XI : LISTE DES NOMENCLATURES RAJOUTES

id_type	typ_mnemonique	typ_label_default	nom_mnemonique	nom_label_default
7	ETA_BIO	État biologique de l'observation	Blessé	Blessé
13	STATUT_BIO	Statut biologique	Couvaison	Couvaison
13	STATUT_BIO	Statut biologique	Chante	Chante
13	STATUT_BIO	Statut biologique	Fuite	Fuite
13	STATUT_BIO	Statut biologique	Vol de migration	Vol de migration
13	STATUT_BIO	Statut biologique	Accouplement	Accouplement
13	STATUT_BIO	Statut biologique	Alarme	Alarme
13	STATUT_BIO	Statut biologique	Vigilance	Vigilance
13	STATUT_BIO	Statut biologique	Construction de nid	Construction de nid
13	STATUT_BIO	Statut biologique	Combat, Défense	Combat, Défense
13	STATUT_BIO	Statut biologique	Déplacement (marche, vol, nage)	Déplacement (marche, vol, nage)
13	STATUT_BIO	Statut biologique	Dortoir	Dortoir
13	STATUT_BIO	Statut biologique	Mise bas	Mise bas
13	STATUT_BIO	Statut biologique	Nourrissage des jeunes	Nourrissage des jeunes
13	STATUT_BIO	Statut biologique	Parade	Parade
13	STATUT_BIO	Statut biologique	Ponte	Ponte
13	STATUT_BIO	Statut biologique	Repos, Reposoirs	Repos, Reposoirs
10	STADE_VIE	Stade de vie : stade de développement du sujet	Juvénile volant	Juvénile volant
10	STADE_VIE	Stade de vie : stade de développement du sujet	Juvénile non-volant	Juvénile non-volant
116	TYPE_SITE	Type de sites	Pt_comptage	Point de comptage mensuel commun
125	PRECI_COMPTAGE	Précision du Comptage	Sous_estim	Sous estimation
125	PRECI_COMPTAGE	Précision du Comptage	Preci_inf_10	Précision < 10%
125	PRECI_COMPTAGE	Précision du Comptage	Ordre_grand	Ordre de grandeur
125	PRECI_COMPTAGE	Précision du Comptage	Estim_deduc	Estimation par déduction

Extrait du fichier *Tableau_des_nomenclatures_rajoutes.xlsx*

ANNEXE XII : FICHE DU JEU DE DONNEES « COMPTAGE MENSUEL COMMUN » DE LA RNMO

Jeu de données
Comptage Mensuel RNMO

Fiche descriptive

Identification

Indentifiant SINP :
6d3fbab9-07c8-4f8f-b1af-cdeb6a6e47de
Instance : Comptage Mensuel RNMO
Nom court : Comp.Mensu.RNMO
Description : Comptage Mensuel RNMO
Type de données : Occurrences de Taxons

Objectifs

Inventaire pour étude d'espèces ou de communautés

Territoires

Domaine continental
Domaine marin

Cadre de référence

Comptage Mensuel Commun
80def5a5-bf28-41df-9431-67e4d738ffa8

Processus de collecte

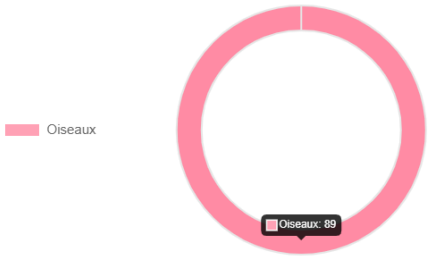
Observation directe : Vue, écoute, olfactive, tactile
Observation directe : Vue, écoute, olfactive, tactile

Contacts

Contact principal : LPO

Producteur du jeu de données : Réserve Naturelle Nationale de Moëze-Oléron

Répartition des espèces



Cadre d'acquisition initial

80def5a5-bf28-41df-9431-67e4d738ffa8
Comptage Mensuel Commun

ANNEXE XIII : FICHIERS DE CONFIGURATION DU SOUS-MODULE « COMPTAGE MENSUEL COMMUN »

custom.json

```
{

  "__CODE_LIST_INVENTOR": "obsocctax",
  "__CODE_LIST_OBSERVER": "obsocctax",
  "__ID_COMPONENT_TAXONOMY": "100",
  "__ID_DATASET_VISIT": 1,
  "__SYNTHESE": true

}
```

config.json

```
{
  "tree": {
    "module": {
      "site": {
        "visit": {
          "observation": null
        }
      }
    }
  },
  "data": {
    "nomenclature": ["PRECI_COMPTAGE"],
    "user": ["__CODE_LIST_INVENTOR", "__CODE_LIST_OBSERVER"]
  }
}
```

module.json

```
{
    "module_label":"Comptage Mensuel Commun [RNMO]",
    "module_desc":"Module pour les comptages mensuels communs de la RNMO"
}
```

nomenclature.json


```
{
  "types": [
    {
      "mnemonique": "PRECI_COMPTAGE",
      "label_default": "Précision du Comptage",
      "definition_default": "Précision du Comptage (Comptage Mensuel
Commun)"
    }
  ],
  "nomenclatures": [
    {
      "type":"PRECI_COMPTAGE",
      "cd_nomenclature": "PRECI_ESTIM_DEDUC",
      "mnemonique": "Estim_deduc",
      "label_default": "Estimation par déduction",
      "definition_default": "Estimation par déduction (Comptage Mensuel
Commun)"
    },
    {
      "type":"PRECI_COMPTAGE",
      "cd_nomenclature": "PRECI_ORDRE_GRAND",
      "mnemonique": "Ordre_grand",
      "label_default": "Ordre de grandeur",
      "definition_default": "Ordre de grandeur (Comptage Mensuel Commun)"
    },
    {
      "type":"PRECI_COMPTAGE",
      "cd_nomenclature": "PRECI_INF_10",
      "mnemonique": "Preci_inf_10",
      "label_default": "Précision < 10%",
      "definition_default": "Précision < 10% (Comptage Mensuel Commun)"
    },
    {
      "type":"PRECI_COMPTAGE",
      "cd_nomenclature": "PRECI_SOUS_ESTIM",
      "mnemonique": "Sous_estim",
      "label_default": "Sous estimation",
      "definition_default": "Sous estimation (Comptage Mensuel Commun)"
    },
    {
      "type":"TYPE_SITE",
      "cd_nomenclature": "COMPT_PT_O",
      "mnemonique": "Pt_comptage",
      "label_default": "Point de comptage mensuel commun",
      "definition_default": "Point de comptage (Comptage Mensuel Commun)"
    }
  ]
}
```

site.json


```
{
  "geometry_type": "Polygon",
  "display_properties": [
    "base_site_name",
    "base_site_code",
    "last_visit",
    "nb_visits"
  ],
  "display_list": [
    "base_site_name",
    "base_site_code",
    "last_visit",
    "nb_visits"
  ],
  "specific": {
    "id_nomenclature_type_site": {
      "type_widget": "text",
      "attribut_label": "Type site",
      "type_util": "nomenclature",
      "value": {
        "code_nomenclature_type": "TYPE_SITE",
        "cd_nomenclature": "COMPT_PT_O"
      },
      "hidden": true
    },
    "id_inventor": {
      "hidden": true
    },
    "base_site_code": {
      "required": false
    },
    "first_use_date": {
      "required": false
    }
  }
}
```

visit.json

```
{
  "display_properties": [
    "visit_date_min",
    "observers",
    "id_nomenclature_precision_comptage",
    "comments",
    "nb_observations"
  ],
  "display_list": [
    "visit_date_min",
    "observers",
    "id_nomenclature_precision_comptage"
  ],
  "keep": [
    "visit_date_min",
    "observers",
    "id_nomenclature_precision_comptage",
    "comments"
  ],
  "specific": {
    "id_base_site": {
      "type_widget": "site",
      "required": true,
      "attribut_label": "Choix du site"
    },
    "visit_date_min": {
      "required": true,
      "attribut_label": "Date"
    },
    "id_nomenclature_precision_comptage": {
      "type_widget": "nomenclature",
      "attribut_label": "Précision",
      "code_nomenclature_type": "PRECI_COMPTAGE",
      "type_util": "nomenclature",
      "required": false
    }
  }
}
```

observation.json

```
{
  "display_properties": [
    "cd_nom",
    "nombre_individu",
    "comments"
  ],
  "specific": {
    "nombre_individu": {
      "type_widget": "text",
      "attribut_label": "Nombre d'individus",
      "required": true,
      "min": 0,
      "max": 100000000
    }
  }
}
```

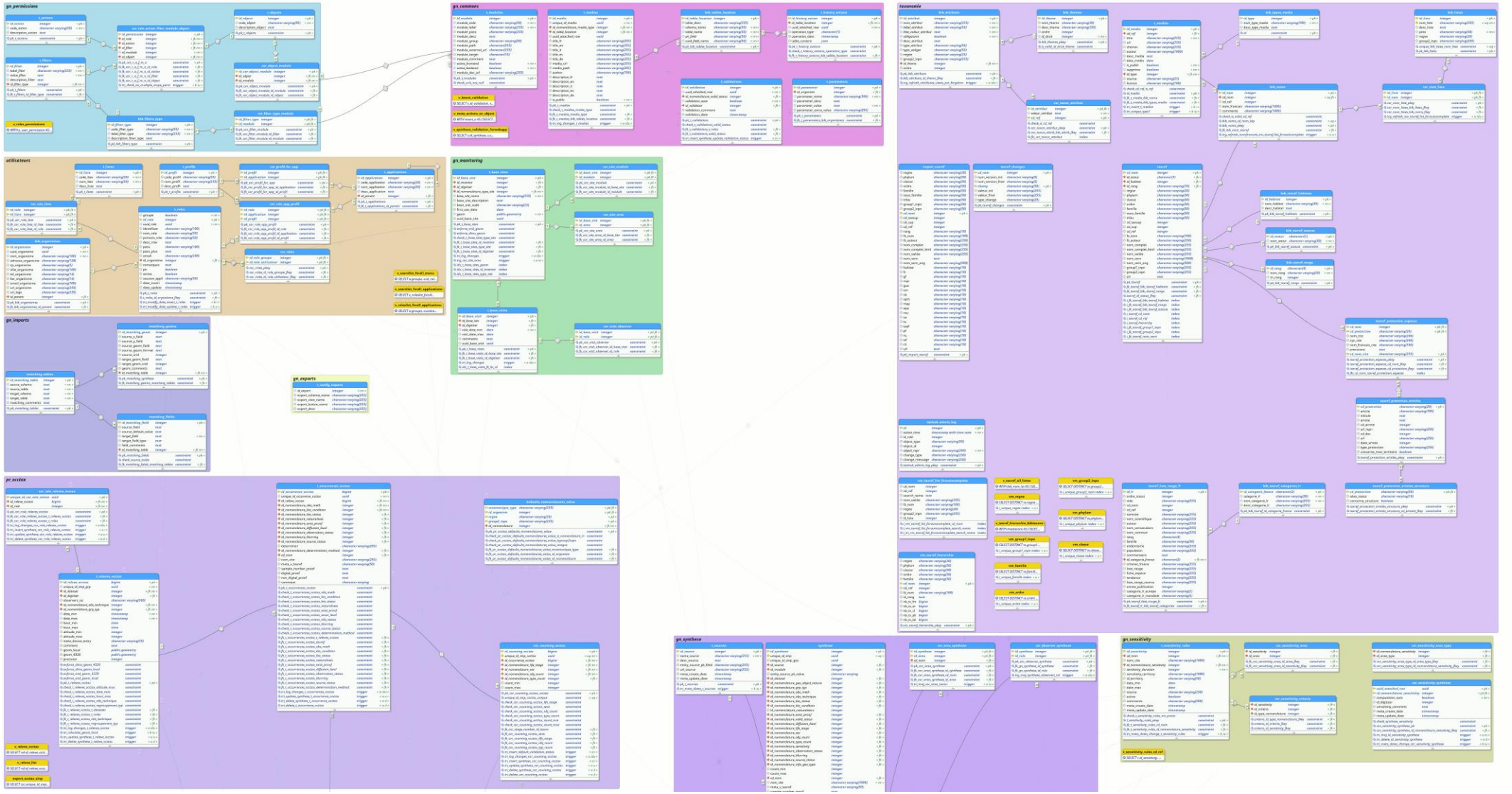
synthese.sql

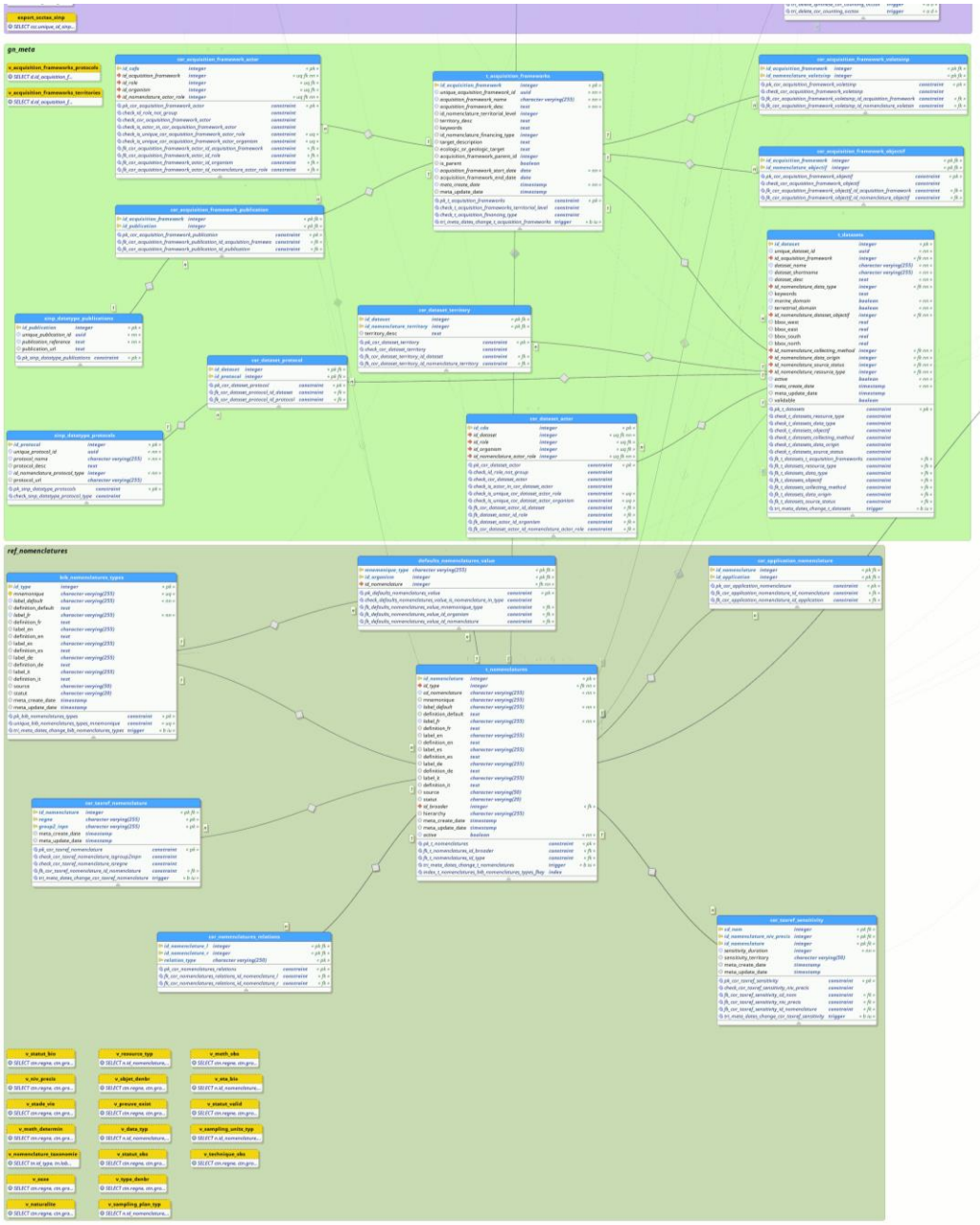
```
DROP VIEW IF EXISTS gn_monitoring.vs_comptage_rnmo;
CREATE VIEW gn_monitoring.vs_comptage_rnmo AS
WITH source AS (
    SELECT id_source
    FROM gn_synthese.t_sources
    WHERE name_source = 'MONITORING_COMPTAGE_RNMO' -- ici
'MONITORING_<module_path>.upper()'
    LIMIT 1
)
SELECT
        o.uuid_observation AS unique_id_sinp,
        v.uuid_base_visit AS unique_id_sinp_grp,
        (SELECT id_source FROM source) as id_source,
        o.id_observation AS entity_source_pk_value,
        v.id_dataset,
        v.id_nomenclature_geo_object_nature,
        v.id_nomenclature_grp_typ,
        v.id_nomenclature_obs_technique,
        ref_nomenclatures.get_id_nomenclature('IND', 'OBJ_DENBR') AS
id_nomenclature_obj_count,
        ref_nomenclatures.get_id_nomenclature('TYP_DENBR', 'Es') AS
id_nomenclature_type_count,
        ref_nomenclatures.get_id_nomenclature('STATUT_OBS', 'Pr') AS
id_nomenclature_observation_status,
        ref_nomenclatures.get_id_nomenclature('STATUT_SOURCE', 'Te') AS
id_nomenclature_source_status,
        ref_nomenclatures.get_id_nomenclature('TYP_INF_GEO', '1') AS
id_nomenclature_info_geo_type,

        (oc.data ->> 'nombre_individu'::text)::integer AS count_min,
        (oc.data ->> 'nombre_individu'::text)::integer AS count_max,
        o.id_observation,
        o.cd_nom::int AS cd_nom,
        t.nom_complet AS nom_cite,
        v.the_geom_4326,
        v.the_geom_point,
        v.geom_local as the_geom_local,
        v.date_min,
        v.date_max,
        observers,
        v.id_digitiser,
        v.id_module,
        v.comment_description,
        ids_observers,

        -- ## Colonnes complémentaires pouvant être utile
        v.id_base_site,
        v.id_base_visit
    FROM gn_monitoring.vs_visits v
    JOIN gn_commons.t_modules m ON m.id_module = v.id_module
    JOIN gn_monitoring.t_observations o ON o.id_base_visit =
v.id_base_visit
    JOIN gn_monitoring.t_observation_complements oc ON oc.id_observation =
o.id_observation
    JOIN taxonomie.taxref t ON t.cd_nom = o.cd_nom
    WHERE m.module_path = 'comptage_rnmo';
```

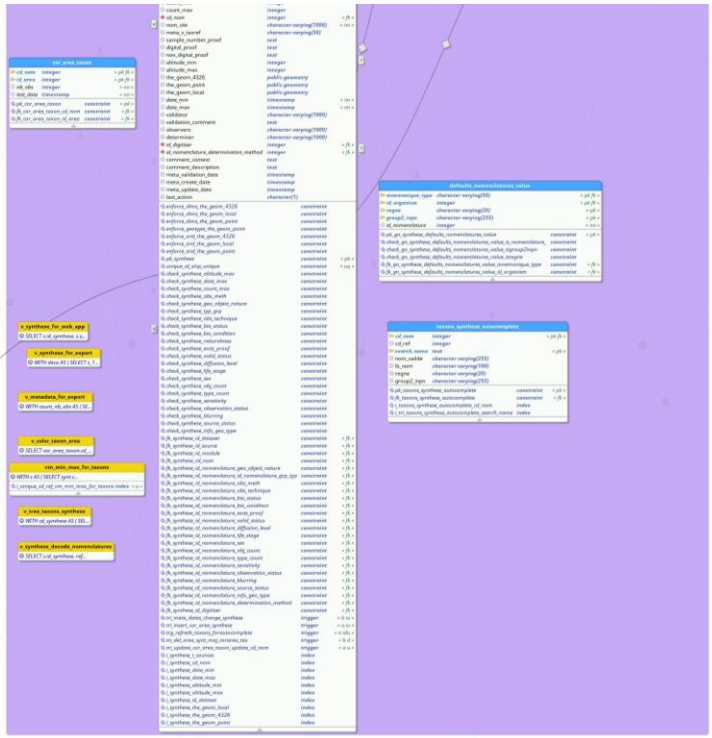
ANNEXE XIV : SCRIPT SQL – CREATION D'UNE VUE POUR LE MODULE « EXPORT »

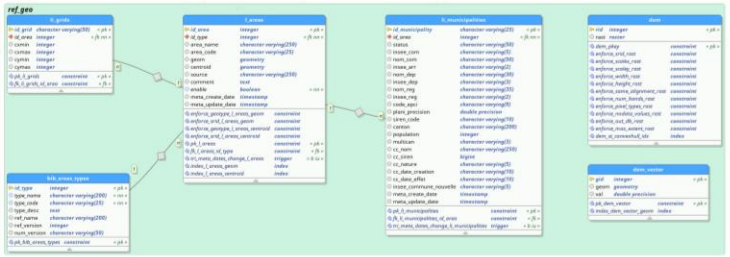
```
SELECT sit.id_base_site AS id_site,
    sit.uuid_base_site AS unique_id_site,
    COALESCE(string_agg(DISTINCT (r1.nom_role::text || ' '::text) ||
r1.prenom_role::text, ','::text)) AS id_digitiser,
    n1.label_default AS type_site,
    sit.base_site_name AS nom_site,
    sit.base_site_code AS code_site,
    sit.first_use_date AS date_enregistrement,
    sit.geom,
    st_x(st_centroid(sit.geom)) AS x_centroid,
    st_y(st_centroid(sit.geom)) AS y_centroid,
    visit.id_base_visit AS id_visite,
    visit.uuid_base_visit AS unique_id_visite,
    visit.visit_date_min,
    COALESCE(string_agg(DISTINCT (r2.nom_role::text || ' '::text) ||
r2.prenom_role::text, ', '::text)) AS observateurs,
    n2.label_default AS "precision",
    visit.comments AS com_visit,
    obs.id_observation,
    obs.uuid_observation AS unique_id_observation,
    (tax.nom_vern::text || ', '::text) || tax.nom_valide::text AS taxon,
    (obscmp.data ->> 'nombre_individu'::text)::integer AS nb_individu,
    obs.comments AS com_observation
   FROM gn_monitoring.t_base_sites sit
     JOIN gn_monitoring.t_base_visits visit ON visit.id_base_site = sit.id_base_site
     JOIN gn_monitoring.t_visit_complements visitcmp ON visitcmp.id_base_visit =
visit.id_base_visit
     JOIN gn_monitoring.t_observations obs ON obs.id_base_visit = visit.id_base_visit
     JOIN gn_monitoring.t_observation_complements obscmp ON obscmp.id_observation =
obs.id_observation
     JOIN gn_monitoring.cor_visit_observer obsv ON obsv.id_base_visit =
visit.id_base_visit
     LEFT JOIN utilisateurs.t_roles r1 ON r1.id_role = sit.id_digitiser
     LEFT JOIN utilisateurs.t_roles r2 ON r2.id_role = obsv.id_role
     LEFT JOIN ref_nomenclatures.t_nomenclatures n1 ON sit.id_nomenclature_type_site =
n1.id_nomenclature
     LEFT JOIN ref_nomenclatures.t_nomenclatures n2 ON ((visitcmp.data ->>
'id_nomenclature_precision_comptage'::text)::integer) = n2.id_nomenclature
     LEFT JOIN taxonomie.taxref tax ON tax.cd_nom = obs.cd_nom
  GROUP BY sit.id_base_site, n1.label_default, visit.id_base_visit, n2.label_default,
obs.id_observation, obscmp.data, tax.nom_valide, tax.nom_vern;
```

ANNEXE XV : MODELE CONCEPTUEL DE LA BASE DE DONNEES DE GEONATURE









ANNEXE XVI : SCRIPT SQL – INSERTION DES REFERENTIELS GEOGRAPHIQUES

```
-- Parcs Nationaux (Aire d'adhésion)

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 20, nom_site, id_mnhn, geom, 'Version: 12/2019', 'INPN', true
FROM ref_geo.inpn_pn
WHERE code_r_enp IN ('APAPN', 'AAPN');

-- Parcs Nationaux (Coeur)

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 1, nom_site, id_mnhn, geom, 'Version: 12/2019', 'INPN', true
FROM ref_geo.inpn_pn
WHERE code_r_enp = 'CPN';

-- Parcs Naturels Régionaux

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 15, nom_site, id_mnhn, geom, 'Version: 12/2019', 'INPN', true
FROM ref_geo.inpn_pnr;

-- Parcs Naturels Marins

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 14, nom_site, id_mnhn, geom, 'Version: 11/2016', 'INPN', true
FROM ref_geo.inpn_pnm;

-- ZICO

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 9, nom, id_diren, geom, 'Version: 1994', 'INPN', true
FROM ref_geo.inpn_zico;

-- ZPS

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 7, sitename, sitecode, geom, 'Version: 12/2019', 'INPN', true
FROM ref_geo.inpn_zps1912;

-- ZSC / SIC

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 8, sitename, sitecode, geom, 'Version: 12/2019', 'INPN', true
FROM ref_geo.inpn_sic1912;

-- ZNIEFF 1 (Terre/Mer)

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 3, nom, id_mnhn, geom, 'Version: 01/2020', 'INPN', true
FROM ref_geo.inpn_znieff1;
```

```
INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 3, nom, id_mnhn, geom, 'Version: 01/2020', 'INPN', true
FROM ref_geo.inpn_znieff1_mer;

-- ZNIEFF 2 (Terre/Mer)

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 2, nom, id_mnhn, geom, 'Version: 01/2020', 'INPN', true
FROM ref_geo.inpn_znieff2;
INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 2, nom, id_mnhn, geom, 'Version: 01/2020', 'INPN', true
FROM ref_geo.inpn_znieff2_mer;

-- Sites Acquis des Conservatoires d'espaces naturels

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 12, nom_site, id_mnhn, geom, 'Version: 12/2018', 'INPN', true
FROM ref_geo.inpn_cen;

-- Départements

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 26, nom_dep, id, geom, 'Version: 2020', 'IGN - Admin Express', true
FROM ref_geo.admexp_dpt;

-- LIEUX DIT S1

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 33, secteur, id_serena, geom, 'Couche: SECTEURBH', 'LPO', true
FROM ref_geo.rnbh_niveau1;

INSERT INTO ref_geo.l_areas (id_type, area_name, geom, comment, source, enable)
SELECT 33, nom, geom, 'Couche: sites parents', 'LPO', true
FROM ref_geo.rnmo_niveau1;

INSERT INTO ref_geo.l_areas (id_type, area_name, geom, comment, source, enable)
SELECT 33, numero, nom, geom, 'Couche: Niveau 1', 'RNSDP', true
FROM ref_geo.rnsdp_niv1_niveau_1;

INSERT INTO ref_geo.l_areas (id_type, area_name, geom, comment, source, enable)
SELECT 6, nom_racc, geom, 'Couche: STEPRO_Niv1_NEW', 'STEPRO', true
FROM ref_geo.stepro_niv1;

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 33, nom_racc, id, geom, 'Couche: STEPRO_Niv2_NEW', 'STEPRO', true
FROM ref_geo.stepro_niv2;

-- UNITE GESTION S2

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 34, toponymes, id_serena, geom, 'Couche: BH1', 'LPO', true
FROM ref_geo.rnbh_niveau2;
```

```
INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 34, nom, reference, geom, 'Couche: toponymie_OLERON_2012-polygon', 'LPO', true
FROM ref_geo.rnmo_niveau2_oleron;

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 34, toponymi_1, toponymie_, geom, 'Couche: toponymie_MOEZE-polygon', 'LPO',
true
FROM ref_geo.rnmo_niveau2_moeze;

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 34, nom, id, geom, 'Couche: Parcellaire', 'RNSDP', true
FROM ref_geo.rnsdp_niv2_parcellaire;

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 34, nom_racc, id, geom, 'Couche: STEPRO_Niv3_NEW', 'STEPRO', true
FROM ref_geo.stepro_niv3;

-- SITE INVENTAIRE S3

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 35, nom, CONCAT('RNMO_', nom), geom, 'Couche: MARES', 'LPO', true
FROM ref_geo.rnmo_niveau3;

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 35, nom, id, geom, 'Couche: Baisses et Mares', 'RNSDP', true
FROM ref_geo.rnsdp_niv3_baisses_et_mares;

INSERT INTO ref_geo.l_areas (id_type, area_name, area_code, geom, comment, source,
enable)
SELECT 35, nom, id, geom, 'Couche: Terrestre', 'RNSDP', true
FROM ref_geo.rnsdp_niv3_terrestre;
```

ANNEXE XVII : SCRIPT PYTHON – INSERTION DES DONNEES D'OBSERVATIONS ALEATOIRES POUR LA STEPRO

```
# -*- coding: utf-8 -*-
from PyQt5.QtSql import *

db = QSqlDatabase.addDatabase("QPSQL", "db1")
db.setDatabaseName("geonature2db")
db.setUserName("geonatureadmin")
db.setPassword("******")
db.setHostName("192.168.1.29")
db.setPort(5432)

# db = QSqlDatabase.addDatabase("QPSQL", "db1")
# db.setDatabaseName("geonature2db")
# db.setUserName("******")
# db.setPassword("*****")
# db.setHostName("********")
# db.setPort(5432)

# ========= Import de la table dans la base PostgreSQL =========
# [TABLES]
schema = "gn_imports"
table = "serena_obs_alea_stepro_1000"
table_origin = schema + "." + table
id_unique = "id_gn"
id_origin_duplique = "obse_id"  # ID dupliqué de la table d'origine

# ========= Execution des scripts de remplacement des choix de liste déroulantes =========

# ========= Définition des variables aléatoires =========
# [RELEVES]
id_dataset = "1"  # ID du Jeu de données
id_digitiser = "17"  # 17 = Maxime TOMA // Plugin 1 = Admin
date_debut = "obse_date_::timestamp"  # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp
date_fin = "obse_date_::timestamp"  # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp
hour_min = "NULL"
hour_max = "NULL"
commentaire_releve = "CONCAT('Commentaire: ',obse_comme,' | Précision: Site | Nom du site: ',site_nom)"
# 'blablabla' // ou // nom_colonne_commentaire // ou // CONCAT(...)
group_by_insert_releve = "GROUP BY " + id_origin_duplique + ", geom, obse_date_, obse_comme, site_nom"

# [OBSERVATEURS]
colonne_observateur_origin = "obsv_libel::integer"

# [OCCURENCES]
methode_observation = "41" # {NOT NULL}
etat_biologique = "157" # {NOT NULL}
statut_biologique = "29"
niveau_de_naturalite = "160"
preuve_existance = "81"
statut_observation = "88"
floutage_donnee = "175"
statut_source = "75"
determinateur = "NULL"
methode_determination = "447"
cd_nom_taxon = "taxo_mnhn_::integer"
detail_preuve_digital = "NULL"
detail_preuve_non_digital = "NULL"
commentaire_occurrences = "obse_nom"

# [COUNTING]
```

```
stade_vie = "1" # {NOT NULL}
sexe = "171" # {NOT NULL}
objet_denombrement = "146" # {NOT NULL}
type_denombrement = "94"
nb_min = "obse_nbrmi::integer"
nb_max = "obse_nbrma::integer"

# [VALIDATION]
statut_validation = "317" # 0(465) = En attente de validation / 1(317) = Certain, Très problable / 
2(318) = Probable / 3(319) = Douteux / 4(320) = Invalide / 5(321) = Non-réalisable / 6(322) = Inconnu
// ou // nom_colonne
id_validateur_origin = "58" # Paul TROTIGNON
commentaire_validation = "NULL" # 'blablabla' // ou // nom_colonne_commentaire // ou // CONCAT(...)
date_validation = "NOW()::timestamp " # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp


if (not db.open()):
    print("Erreur", u"Impossible de se connecter à la base de données : " + db.lastError().text())

else:
    # ========= Insertion des relevés =========

    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occtax.t_releves_occtax (id_dataset, id_digitiser,
id_nomenclature_obs_technique, id_nomenclature_grp_typ, date_min, date_max, hour_min, hour_max,
meta_device_entry, comment, geom_local, geom_4326, precision) "
    sql +="SELECT "+ id_dataset +", "+ id_digitiser +", 316, 132, "+ date_debut +", "+ date_fin +", "+
hour_min +", "+ hour_max +", 'script python', "+ commentaire_releve +", ST_Transform(geom, 2154) AS
geom2154, geom, 100 "
    sql +="FROM "+ table_origin +" "
    sql += group_by_insert_releve +" "
    sql +="ORDER BY "+ id_origin_duplique +" RETURNING id_releve_occtax;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[RELEVE]")

    query2 = QSqlQuery(db) # On selectionne juste les id en respectant les conditions précédentes et le
ORDER BY
    sql2 = "SELECT "+ id_origin_duplique +" FROM " + table_origin + " "
    sql2 += "GROUP BY "+ id_origin_duplique +" "
    sql2 += "ORDER BY "+ id_origin_duplique +";"
    query2.prepare(sql2)
    if not query2.exec_():
        print(query2.lastError().text())
        print("[RELEVE ID]")

    liste = [] # Définition de la correspondance id_origin / id_station et insertion dans une liste de
tuples
    while query.next() and query2.next():
        queryIdReleve = query.value(0)
        queryIdOrigin = query2.value(0)
        tuple = (queryIdOrigin,queryIdReleve)
        liste.append(tuple)

    # ========= Création d'une colonne "id_releve" si elle n'existe pas déjà =========
    # ========= Pour tous les éléments de la liste,  nous allons UPDATE la colonne id_station de la
table_origine et faire ainsi la correspondance =========

    for elt in liste:
        id_origin = elt[0]
        id_releve = elt[1]

        query = QSqlQuery(db)
        sql = "UPDATE "+ table_origin +" SET id_releve = ? "
```

```
        sql += "WHERE "+ id_origin_duplique +" = ?;"
        query.prepare(sql)
        query.addBindValue(id_releve)
        query.addBindValue(id_origin)
        if not query.exec_():
            print(query.lastError().text())
            print("[UPDATE ID_RELEVE]")

    # ========= Pour tous les éléments de la liste, nous allons faire la correspondance
observateur/relevé =========

    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occtax.cor_role_releves_occtax (id_releve_occtax, id_role)"
    sql += "SELECT id_releve, " + colonne_observateur_origin + " "
    sql += "FROM " + table_origin + ";"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans cor_role_releve]")

    # ========= Récupération de la version de TaxRef pour insertion =========

    query = QSqlQuery(db)
    sql = "SELECT parameter_value FROM gn_commons.t_parameters "
    sql += "WHERE parameter_name = 'taxref_version';"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Recupération de TaxRef]")

    taxref_version = None
    if query.next():
        taxref_version = query.value(0)

    # ========= Insertion des occurrences =========

    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occtax.t_occurrences_occtax (id_releve_occtax, id_nomenclature_obs_meth,
id_nomenclature_bio_condition, id_nomenclature_bio_status, id_nomenclature_naturalness,
id_nomenclature_exist_proof, "
    sql += "id_nomenclature_observation_status, id_nomenclature_blurring,
id_nomenclature_source_status, determiner, id_nomenclature_determination_method, cd_nom, nom_cite,
meta_v_taxref, digital_proof, non_digital_proof, comment) "
    sql += "SELECT "+table+".id_releve, "+ methode_observation +", "+ etat_biologique +", "+
statut_biologique +", "+ niveau_de_naturalite +", "+ preuve_existance +", "+ statut_observation +", "
    sql += floutage_donnee +", "+ statut_source +", "+ determinateur +", "+ methode_determination +", "
    sql += table+"."+ cd_nom_taxon +", CONCAT(taxref.nom_vern,' = <i> ', taxref.nom_complet, '</i> -
[', taxref.id_rang, ' - ', taxref.cd_nom ,']') AS nom_cite, '"+ taxref_version +"', "+
detail_preuve_digital +", "+ detail_preuve_non_digital +", "+ commentaire_occurrences +" "
    sql += "FROM "+ table_origin +" "
    sql += "LEFT JOIN pr_occtax.t_releves_occtax ON "+ table +".id_releve =
t_releves_occtax.id_releve_occtax "
    sql += "LEFT JOIN taxonomie.taxref ON "+ table +"."+ cd_nom_taxon +" = taxref.cd_nom "
    sql += "ORDER BY "+ table +"."+ id_unique +" RETURNING id_occurrence_occtax;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion Occurrences]")

    query2 = QSqlQuery(db) # On selectionne juste les id en respectant les conditions précédentes et le
ORDER BY
    sql2 = "SELECT "+ id_unique +" FROM " + table_origin + " "
    sql2 += "ORDER BY "+ id_unique +";"
    query2.prepare(sql2)
    if not query2.exec_():
        print(query2.lastError().text())
```

```
        print("[Récupération ID Occurrences]")

    liste2 = [] # Définition de la correspondance id_unique / id_occurrence_occtax et insertion dans
une liste de tuples
    while query.next() and query2.next():
        queryIdOccurrence = query.value(0)
        queryIdUnique = query2.value(0)
        tuple = (queryIdUnique,queryIdOccurrence)
        liste2.append(tuple)

    # ========= Création d'une colonne "id_occurrence_occtax" si elle n'existe pas déjà =========

    # ========= Pour tous les éléments de la liste, nous allons UPDATE la colonne id_occurence_occtax
de la table_origine et faire ainsi la correspondance =========

    for elt2 in liste2:
        id_origin = elt2[0]
        id_occurrence_occtax = elt2[1]

        query = QSqlQuery(db)
        sql = "UPDATE "+ table_origin +" SET id_occurrence_occtax = ? "
        sql += "WHERE "+ id_unique +" = ?;"
        query.prepare(sql)
        query.addBindValue(id_occurrence_occtax)
        query.addBindValue(id_origin)
        if not query.exec_():
            print(query.lastError().text())
            print("[Insertion de l'id_occurrence_occtax]")


    # ========= Insertion dans cor_couting_occtax =========
    # Insert auto dans la synthèse plus long (normal)

    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occtax.cor_counting_occtax (id_occurrence_occtax, id_nomenclature_life_stage,
id_nomenclature_sex, id_nomenclature_obj_count, id_nomenclature_type_count, count_min, count_max) "
    sql += "SELECT " + table + ".id_occurrence_occtax, " + stade_vie + ", " + sexe + ", " +
objet_denombrement + ", " + type_denombrement + ", " + nb_min + ", " + nb_max + " "
    sql += "FROM pr_occtax.t_occurrences_occtax "
    sql += "RIGHT JOIN " + table_origin + " ON " + table + ".id_occurrence_occtax =
t_occurrences_occtax.id_occurrence_occtax;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans cor_counting]")

    # ========= Validation =========

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_commons.t_validations (uuid_attached_row, id_nomenclature_valid_status,
id_validator, validation_auto, validation_date) "
    sql += "SELECT unique_id_sinp, "+ statut_validation +", "+ id_validateur_origin +", false, NOW() "
    sql += "FROM gn_synthese.synthese "
    sql += "INNER JOIN " + table_origin + " ON "+ table +".id_occurrence_occtax =
synthese.entity_source_pk_value::integer - 1;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans t_validations]")

    db.close()
    del db
    db = None
    QSqlDatabase.removeDatabase('db1')
```

ANNEXE XVIII : SCRIPT PYTHON – INSERTION DES DONNEES HABITATS POUR LA RNMO

```
# -*- coding: utf-8 -*-
from PyQt5.QtSql import *

db = QSqlDatabase.addDatabase("QPSQL", "db1")
db.setDatabaseName("geonature2db")
db.setUserName("postgres")
db.setPassword("pgAdmin")
db.setHostName("localhost")
db.setPort(5433)

if (not db.open()):
    print("Erreur", u"Impossible de se connecter à la base de données : " + db.lastError().text())

else:
    # ========= Import de la table dans la base PostgreSQL =========

    # ========= Création d'une colonne "id_station" si elle n'existe pas déjà =========

    # ========= Définition des variables aléatoires =========
    schema = "histo_imports"
    table = "hab_rnmo"
    table_origin = schema+"."+table

    id_dataset = "20" #ID du Jeu de données
    id_origin_duplique = "id" #ID dupliqué de la table d'origine
    date_debut = "TIMESTAMP '2012-01-01 00:00:00'" # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp
    date_fin = "TIMESTAMP '2012-01-01 00:00:00'"  # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp
    date_debut2 = "TIMESTAMP '2018-01-01 00:00:00'"  # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp
    date_fin2 = "TIMESTAMP '2018-01-01 00:00:00'"  # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp
    surface = "surfacem2" # ST_Area(geom) // ou // nom_colonne_surface + ATTENTION à voir si la colonne
est bien avec une surface en m²
    commentaire = "'blablabla'" # 'blablabla' // ou // nom_colonne_commentaire // ou // CONCAT(...)
    group_by_insert_station = "GROUP BY "+ id_origin_duplique +", surfacem2, geom"

    colonne_habitat_origin = "habi"
    code_typo_habitat = "7" # EUNIS = 7 // CORINE = 22 (à vérifier quand même systématiquement)

    # ========= Insertion des stations =========

    # Condition: Pas de Complexe d'habitat + Source = Jean TERRISSE
    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occhab.t_stations (id_dataset, date_min, date_max, is_habitat_complex, area,
id_nomenclature_area_surface_calculation, comment, geom_local, geom_4326,
id_nomenclature_geographic_object) "
    sql +="SELECT "+ id_dataset +", "+ date_debut +", "+ date_fin +", false, "+ surface +"::bigint,
482, "+ commentaire +", geom, ST_Transform(geom, 4326) AS geom4326, 174 "
    sql +="FROM "+ table_origin +" "
    sql +="WHERE observateu = 'Stéphane Guenneteau' "
    sql += group_by_insert_station +" "
    sql +="HAVING COUNT("+ id_origin_duplique +"::integer) = 1 "
    sql +="ORDER BY "+ id_origin_duplique +" RETURNING id_station;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())

    query2 = QSqlQuery(db) # ========= On selectionne juste les id en respectant les conditions
précédentes et le ORDER BY =========
```

```
    sql2 = "SELECT "+ id_origin_duplique +" FROM " + table_origin + " "
    sql2 += "WHERE observateu = 'Stéphane Guenneteau' "
    sql2 += "GROUP BY "+ id_origin_duplique +" "
    sql2 += "HAVING COUNT("+ id_origin_duplique +"::integer) = 1 "
    sql2 += "ORDER BY "+ id_origin_duplique +";"
    query2.prepare(sql2)
    if not query2.exec_():
        print(query2.lastError().text())

    liste = [] # ========= Définition de la correspondance id_origin / id_station et insertion dans une liste de tuples =========
    while query.next() and query2.next():
        queryIdStation = query.value(0)
        queryIdOrigin = query2.value(0)
        tuple = (queryIdOrigin,queryIdStation)
        liste.append(tuple)
    print(liste)

    # Condition: Complexe d'habitat + Source = Jean TERRISSE
    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occhab.t_stations (id_dataset, date_min, date_max, is_habitat_complex, area, id_nomenclature_area_surface_calculation, comment, geom_local, geom_4326, id_nomenclature_geographic_object) "
    sql += "SELECT " + id_dataset + ", "+ date_debut +", "+ date_fin +", true, "+ surface +"::bigint, 482, "+ commentaire +", geom, ST_Transform(geom, 4326) AS geom4326, 174 "
    sql += "FROM " + table_origin + " "
    sql += "WHERE observateu = 'Stéphane Guenneteau' "
    sql +=  group_by_insert_station +" "
    sql += "HAVING COUNT("+ id_origin_duplique +"::integer) > 1 "
    sql += "ORDER BY "+ id_origin_duplique +" RETURNING id_station;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())

    query2 = QSqlQuery(db)
    sql2 = "SELECT "+ id_origin_duplique +" FROM " + table_origin + " "
    sql2 += "WHERE observateu = 'Stéphane Guenneteau' "
    sql2 += "GROUP BY "+ id_origin_duplique +" "
    sql2 += "HAVING COUNT("+ id_origin_duplique +"::integer) > 1 "
    sql2 += "ORDER BY "+ id_origin_duplique +";"
    query2.prepare(sql2)
    if not query2.exec_():
        print(query2.lastError().text())

    while query.next() and query2.next():
        queryIdStation = query.value(0)
        queryIdOrigin = query2.value(0)
        tuple = (queryIdOrigin, queryIdStation)
        liste.append(tuple)
    print(liste)

    # Condition: Pas de Complexe d'habitat + Source = Jean TERRISSE
    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occhab.t_stations (id_dataset, date_min, date_max, is_habitat_complex, area, id_nomenclature_area_surface_calculation, comment, geom_local, geom_4326, id_nomenclature_geographic_object) "
    sql += "SELECT " + id_dataset + ", " + date_debut2 + ", " + date_fin2 + ", false, " + surface + "::bigint, 482, " + commentaire + ", geom, ST_Transform(geom, 4326) AS geom4326, 174 "
    sql += "FROM " + table_origin + " "
    sql += "WHERE observateu = 'Francis Lalanne' "
    sql += group_by_insert_station + " "
    sql += "HAVING COUNT(" + id_origin_duplique + "::integer) = 1 "
    sql += "ORDER BY " + id_origin_duplique + " RETURNING id_station;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
```

```
    query2 = QSqlQuery(db)
    sql2 = "SELECT " + id_origin_duplique + " FROM " + table_origin + " "
    sql2 += "WHERE observateu = 'Francis Lalanne' "
    sql2 += "GROUP BY " + id_origin_duplique + " "
    sql2 += "HAVING COUNT(" + id_origin_duplique + "::integer) = 1 "
    sql2 += "ORDER BY " + id_origin_duplique + ";"
    query2.prepare(sql2)
    if not query2.exec_():
        print(query2.lastError().text())

    while query.next() and query2.next():
        queryIdStation = query.value(0)
        queryIdOrigin = query2.value(0)
        tuple = (queryIdOrigin, queryIdStation)
        liste.append(tuple)
    print(liste)

    # Condition: Complexe d'habitat + Source = Jean TERRISSE
    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occhab.t_stations (id_dataset, date_min, date_max, is_habitat_complex, area,
id_nomenclature_area_surface_calculation, comment, geom_local, geom_4326,
id_nomenclature_geographic_object) "
    sql += "SELECT " + id_dataset + ", " + date_debut2 + ", " + date_fin2 + ", true, " + surface +
"::bigint, 482, " + commentaire + ", geom, ST_Transform(geom, 4326) AS geom4326, 174 "
    sql += "FROM " + table_origin + " "
    sql += "WHERE observateu = 'Francis Lalanne' "
    sql += group_by_insert_station + " "
    sql += "HAVING COUNT(" + id_origin_duplique + "::integer) > 1 "
    sql += "ORDER BY " + id_origin_duplique + " RETURNING id_station;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())

    query2 = QSqlQuery(db)
    sql2 = "SELECT " + id_origin_duplique + " FROM " + table_origin + " "
    sql2 += "WHERE observateu = 'Francis Lalanne' "
    sql2 += "GROUP BY " + id_origin_duplique + " "
    sql2 += "HAVING COUNT(" + id_origin_duplique + "::integer) > 1 "
    sql2 += "ORDER BY " + id_origin_duplique + ";"
    query2.prepare(sql2)
    if not query2.exec_():
        print(query2.lastError().text())

    while query.next() and query2.next():
        queryIdStation = query.value(0)
        queryIdOrigin = query2.value(0)
        tuple = (queryIdOrigin, queryIdStation)
        liste.append(tuple)
    print(liste)

    query2 = None
    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occhab.cor_station_observer (id_station, id_role) "
    sql += "SELECT id_station, 316 "
    sql += "FROM pr_occhab.t_stations "
    sql += "WHERE id_dataset = "+ id_dataset +" AND date_min BETWEEN '2012-01-01' AND '2012-12-31';"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())

    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occhab.cor_station_observer (id_station, id_role) "
    sql += "SELECT id_station, 336 "
    sql += "FROM pr_occhab.t_stations "
    sql += "WHERE id_dataset = "+ id_dataset +" AND date_min BETWEEN '2018-01-01' AND '2018-12-31';"
```

```
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())

    # ========= Pour tous les éléments de la liste,  nous allons UPDATE la colonne id_station de la
table_origine et faire ainsi la correspondance =========

    for elt in liste:
        id_origin = elt[0]
        id_station = elt[1]

        query = QSqlQuery(db)
        sql = "UPDATE "+ table_origin +" SET id_station = ? "
        sql += "WHERE "+ id_origin_duplique +" = ?;"
        query.prepare(sql)
        query.addBindValue(id_station)
        query.addBindValue(id_origin)
        if not query.exec_():
            print(query.lastError().text())

    # ========= Insertion des habitats =========

    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occhab.t_habitats (id_station, cd_hab, nom_cite,
id_nomenclature_determination_type, id_nomenclature_collection_technique) "
    sql += "SELECT t_stations.id_station, cd_hab, CONCAT(lb_code,' - ', lb_hab_fr) AS nom_cite, 486,
490 "
    sql += "FROM pr_occhab.t_stations, ref_habitats.habref, "+ table_origin +" "
    sql += "WHERE "+table+".id_station = t_stations.id_station AND is_habitat_complex = 'false' AND "+
colonne_habitat_origin +" = lb_code AND cd_typo = "+ code_typo_habitat +";"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())

    query = QSqlQuery(db)
    sql = "INSERT INTO pr_occhab.t_habitats (id_station, cd_hab, nom_cite,
id_nomenclature_determination_type, id_nomenclature_collection_technique) "
    sql += "SELECT t_stations.id_station, cd_hab, CONCAT(lb_code,' - ', lb_hab_fr) AS nom_cite, 486,
490 "
    sql += "FROM pr_occhab.t_stations, ref_habitats.habref, " + table_origin + " "
    sql += "WHERE " + table + ".id_station = t_stations.id_station AND is_habitat_complex = 'true' AND
"+ colonne_habitat_origin +" = lb_code AND cd_typo = "+ code_typo_habitat +";"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())

    db.close()
    del db
    db = None
    QSqlDatabase.removeDatabase('db1')
```

ANNEXE XIX : SCRIPT PYTHON – INSERTION DES DONNEES « COMPTAGE MENSUEL COMMUN » DE LA RNMO DANS LE MODULE MONITORING

```
# -*- coding: utf-8 -*-
from PyQt5.QtSql import *

db = QSqlDatabase.addDatabase("QPSQL", "db1")
db.setDatabaseName("geonature2db")
db.setUserName("geonatureadmin")
db.setPassword("*******")
db.setHostName("******")
db.setPort(5432)

# db = QSqlDatabase.addDatabase("QPSQL", "db1")
# db.setDatabaseName("geonature2db")
# db.setUserName("frederic")
# db.setPassword("*******")
# db.setHostName("*******")
# db.setPort(5432)

# ========= Import de la table dans la base PostgreSQL =========
# [TABLES]
schema = "gn_imports"
table = "rnmo_compt_mensu"
table_origin = schema + "." + table
id_unique = "id"
id_origin_duplique = "id"  # ID dupliqué de la table d'origine

# ========= Execution des scripts de remplacement des choix de liste déroulantes =========
# ========= Définition des variables aléatoires =========
# [SITES]
id_inventor_digitiser = "17"  # 17 = Maxime TOMA // Plugin 1 = Admin
type_site = "559"
nom_site = "obse_nom2"
group_by_insert_site = "GROUP BY ref.geom, "+ nom_site

# [OBSERVATEURS]
colonne_observateur_origin = "obse_obsv::integer"
colonne_observateur_origin2 = "obse_obsv2::integer"

# [VISITES]
id_dataset = "1"
id_module = "17"
date_debut = "obse_date::timestamp"  # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp
precision_comptage = "obse_preci::text"
commentaire_visits = "'Donnée historique intégrée le 24/08/2020'"

# [OBSERVATIONS]
cd_nom_taxon = "obse_mnhn::integer"
nombre = "obse_nombre::integer"

# [VALIDATION]
statut_validation = "317" # 0(465) = En attente de validation / 1(317) = Certain, Très problable /
2(318) = Probable / 3(319) = Douteux / 4(320) = Invalide / 5(321) = Non-réalisable / 6(322) = Inconnu
// ou // nom_colonne
id_validateur_origin = "10" # Frédéric ROBIN
commentaire_validation = "NULL" # 'blablabla' // ou // nom_colonne_commentaire // ou // CONCAT(...)
date_validation = "NOW()::timestamp " # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp
```

```
if (not db.open()):
    print("Erreur", u"Impossible de se connecter à la base de données : " + db.lastError().text())

else:
    # ========= Insertion des sites =========
    query = QSqlQuery(db)
    sql = "ALTER TABLE gn_monitoring.t_base_sites DISABLE TRIGGER tri_t_base_sites_calculate_alt; "
    if not query.exec_(sql):
        print(query.lastError().text())
        print("[Désactivation trigger]")

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.t_base_sites (id_inventor, id_digitiser,
id_nomenclature_type_site, base_site_name, geom, geom_local) "
    sql += "SELECT "+ id_inventor_digitiser +", "+ id_inventor_digitiser +", "+ type_site +", "+
nom_site +", (ST_DUMP(ST_Transform(ref.geom, 4326))).geom::geometry(Polygon,4326) AS geom4326,
(ST_DUMP(ref.geom)).geom::geometry(Polygon,2154) AS geom2154 "
    sql += "FROM " + table_origin + " "
    sql += "INNER JOIN ref_geo.l_areas ref ON ref.area_name = "+ nom_site +" "
    sql += group_by_insert_site +" "
    sql +="ORDER BY "+ nom_site +" RETURNING id_base_site;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[RELEVE]")
    print(sql)

    query2 = QSqlQuery(db) # On selectionne juste les id en respectant les conditions précédentes et le
ORDER BY
    sql2 = "SELECT "+ nom_site +" FROM " + table_origin + " "
    sql2 += "INNER JOIN ref_geo.l_areas ref ON ref.area_name = " + nom_site + " "
    sql2 += group_by_insert_site +" "
    sql2 += "ORDER BY "+ nom_site +";"
    query2.prepare(sql2)
    if not query2.exec_():
        print(query2.lastError().text())
        print("[RELEVE ID]")
    print(sql2)

    liste = [] # Définition de la correspondance id_origin / id_station et insertion dans une liste de
tuples
    while query.next() and query2.next():
        queryIdReleve = query.value(0)
        queryIdOrigin = query2.value(0)
        tuple = (queryIdOrigin,queryIdReleve)
        liste.append(tuple)

    query = QSqlQuery(db)
    sql = "ALTER TABLE gn_monitoring.t_base_sites ENABLE TRIGGER tri_t_base_sites_calculate_alt; "
    if not query.exec_(sql):
        print(query.lastError().text())
        print("[Réactiviation trigger]")

    # ========= Création d'une colonne "id_releve"  =========

    query = QSqlQuery(db)
    sql = "ALTER TABLE " + table_origin + " DROP COLUMN IF EXISTS id_origin_site; "
    sql += "ALTER TABLE "+ table_origin +" ADD COLUMN id_origin_site integer;"
    if not query.exec_(sql):
        print(query.lastError().text())

    # ========= Pour tous les éléments de la liste,  nous allons UPDATE la colonne id_station de la
table_origine et faire ainsi la correspondance =========

    for elt in liste:
        id_nom_site = elt[0]
```

```
        id_site = elt[1]

        query = QSqlQuery(db)
        sql = "UPDATE "+ table_origin +" SET id_origin_site = ? "
        sql += "WHERE "+ nom_site +" = ?;"
        query.prepare(sql)
        query.addBindValue(id_site)
        query.addBindValue(id_nom_site)
        if not query.exec_():
            print(query.lastError().text())
            print("[UPDATE ID_RELEVE]")


    # ========= Liaison dans la table cor_site_module et t_site_complements =========

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.cor_site_module (id_base_site, id_module)"
    sql += "SELECT id_origin_site, " + id_module + " "
    sql += "FROM " + table_origin + " GROUP BY id_origin_site;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans cor_site_module]")

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.t_site_complements (id_base_site, id_module)"
    sql += "SELECT id_origin_site, " + id_module + " "
    sql += "FROM " + table_origin + " GROUP BY id_origin_site;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans t_site_complements]")


    # ========= Insertion des visites (base) =========

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.t_base_visits (id_base_site, id_dataset, id_module, id_digitiser,
visit_date_min, visit_date_max, comments) "
    sql += "SELECT id_origin_site, "+ id_dataset +", "+ id_module +", "+ id_inventor_digitiser +", "+
date_debut +", "+ date_debut +", "+ commentaire_visits +" "
    sql += "FROM "+ table_origin +" "
    sql += "ORDER BY "+ id_unique +" RETURNING id_base_visit;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion Visites]")

    query2 = QSqlQuery(db) # On selectionne juste les id en respectant les conditions précédentes et le
ORDER BY
    sql2 = "SELECT "+ id_unique +" FROM " + table_origin + " "
    sql2 += "ORDER BY "+ id_unique +";"
    query2.prepare(sql2)
    if not query2.exec_():
        print(query2.lastError().text())
        print("[Récupération ID Visites]")

    liste2 = [] # Définition de la correspondance id_unique / id_occurrence_occtax et insertion dans
une liste de tuples
    while query.next() and query2.next():
        queryIdOccurrence = query.value(0)
        queryIdUnique = query2.value(0)
        tuple = (queryIdUnique,queryIdOccurrence)
        liste2.append(tuple)

    # ========= Création d'une colonne "id_occurrence_occtax" si elle n'existe pas déjà =========
```

```
    query = QSqlQuery(db)
    sql = "ALTER TABLE " + table_origin + " DROP COLUMN IF EXISTS id_origin_visit; "
    sql += "ALTER TABLE "+ table_origin +" ADD COLUMN id_origin_visit integer;"
    if not query.exec_(sql):
        print(query.lastError().text())

    # ========= Pour tous les éléments de la liste,  nous allons UPDATE la colonne id_occurence_occtax
de la table_origine et faire ainsi la correspondance =========

    for elt2 in liste2:
        id_origin = elt2[0]
        id_visit = elt2[1]

        query = QSqlQuery(db)
        sql = "UPDATE "+ table_origin +" SET id_origin_visit = ? "
        sql += "WHERE "+ id_unique +" = ?;"
        query.prepare(sql)
        query.addBindValue(id_visit)
        query.addBindValue(id_origin)
        if not query.exec_():
            print(query.lastError().text())
            print("[Insertion de l'id_origin_visit]")

    # ========= Pour tous les éléments de la liste, nous allons faire la correspondance
observateur/relevé =========

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.cor_visit_observer (id_base_visit, id_role)"
    sql += "SELECT id_origin_visit, " + colonne_observateur_origin + " "
    sql += "FROM " + table_origin + ";"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans cor_visit_observer]")

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.cor_visit_observer (id_base_visit, id_role)"
    sql += "SELECT id_origin_visit, " + colonne_observateur_origin2 + " "
    sql += "FROM " + table_origin + " "
    sql += "WHERE " + colonne_observateur_origin2 + " IS NOT NULL;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans cor_visit_observer]")

    # ========= Insertion dans t_visit_complements =========

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.t_visit_complements (id_base_visit, data) "
    sql += "SELECT id_origin_visit, CONCAT('{\"id_nomenclature_precision_comptage\":', "+
precision_comptage +",'}')::jsonb "
    sql += "FROM "+table_origin+" "
    sql += "WHERE "+ precision_comptage +" IS NOT NULL;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans cor_counting]")

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.t_visit_complements (id_base_visit, data) "
    sql += "SELECT id_origin_visit, CONCAT('{\"id_nomenclature_precision_comptage\":',
'null','}')::jsonb "
    sql += "FROM " + table_origin + " "
    sql += "WHERE " + precision_comptage + " IS NULL;"
    query.prepare(sql)
```

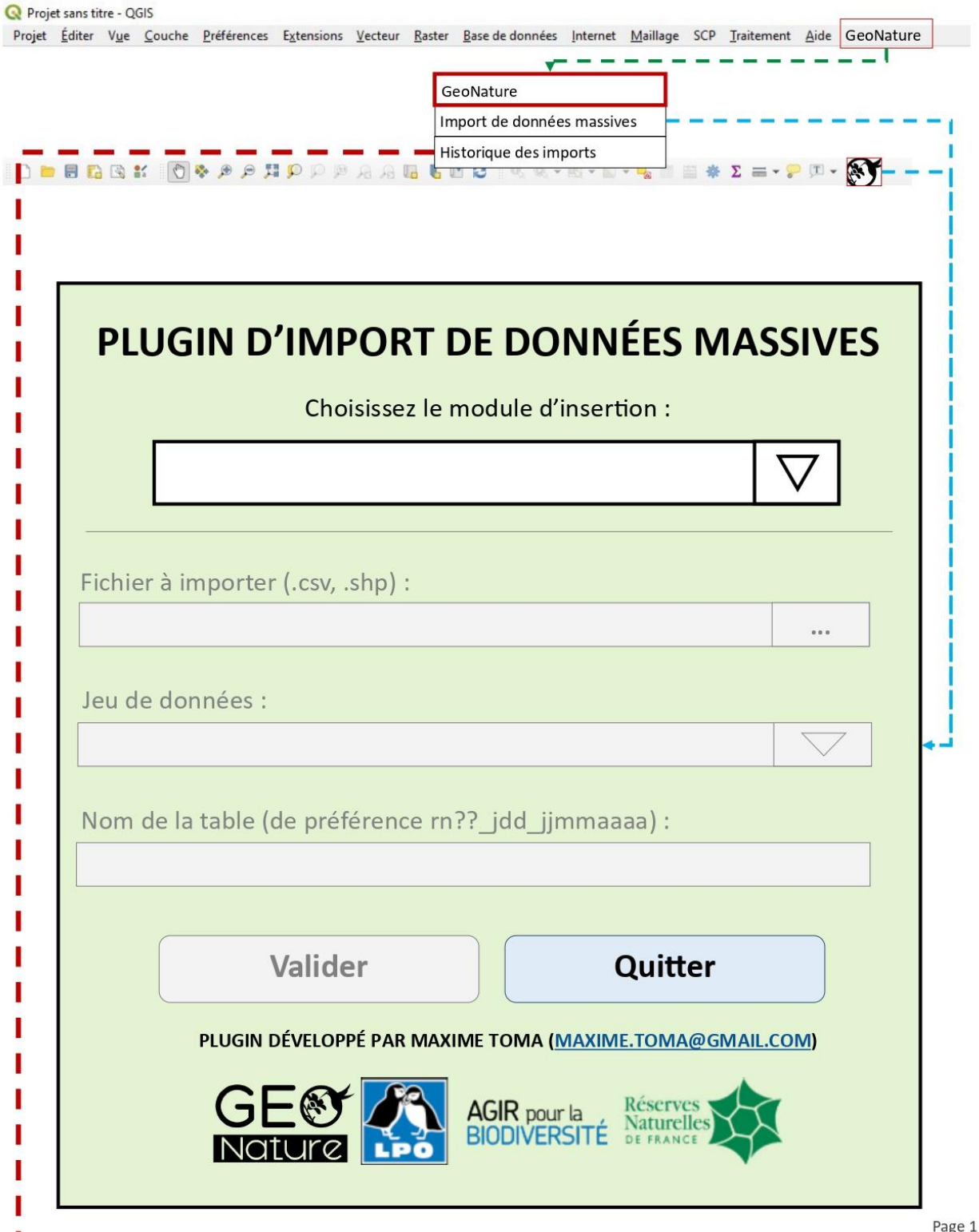
```
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans cor_counting]")

    # ========= Insertion des observations (base) =========

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.t_observations (id_base_visit, cd_nom) "
    sql += "SELECT id_origin_visit, taxref.cd_nom "
    sql += "FROM "+ table_origin +" "
    sql += "LEFT JOIN taxonomie.taxref ON "+ table +"."+ cd_nom_taxon +" = taxref.cd_nom "
    sql += "ORDER BY "+ id_unique +" RETURNING id_observation;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion Visites]")

    query2 = QSqlQuery(db) # On selectionne juste les id en respectant les conditions précédentes et le
ORDER BY
    sql2 = "SELECT "+ id_unique +" FROM " + table_origin + " "
    sql2 += "ORDER BY "+ id_unique +";"
    query2.prepare(sql2)
    if not query2.exec_():
        print(query2.lastError().text())
        print("[Récupération ID Visites]")

    liste3 = [] # Définition de la correspondance id_unique / id_occurrence_occtax et insertion dans
une liste de tuples
    while query.next() and query2.next():
        queryIdObserv = query.value(0)
        queryIdUnique = query2.value(0)
        tuple = (queryIdUnique,queryIdObserv)
        liste3.append(tuple)

    # ========= Création d'une colonne "id_occurrence_occtax" si elle n'existe pas déjà =========

    query = QSqlQuery(db)
    sql = "ALTER TABLE " + table_origin + " DROP COLUMN IF EXISTS id_origin_observ; "
    sql += "ALTER TABLE "+ table_origin +" ADD COLUMN id_origin_observ integer;"
    if not query.exec_(sql):
        print(query.lastError().text())

    # ========= Pour tous les éléments de la liste,  nous allons UPDATE la colonne id_occurence_occtax
de la table_origine et faire ainsi la correspondance =========

    for elt3 in liste3:
        id_origin = elt3[0]
        id_observ = elt3[1]

        query = QSqlQuery(db)
        sql = "UPDATE "+ table_origin +" SET id_origin_observ = ? "
        sql += "WHERE "+ id_unique +" = ?;"
        query.prepare(sql)
        query.addBindValue(id_observ)
        query.addBindValue(id_origin)
        if not query.exec_():
            print(query.lastError().text())
            print("[Insertion de l'id_origin_observ]")

    # ========= Insertion dans t_observation_complements =========

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.t_observation_complements (id_observation, data) "
    sql += "SELECT id_origin_observ, CONCAT('{\"nombre_individu\":', "+ nombre +",'}')::jsonb "
    sql += "FROM "+table_origin+" "
    sql += "WHERE "+ nombre +" IS NOT NULL;"
```

```
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans t_observation_complements]")

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_monitoring.t_visit_complements (id_base_visit, data) "
    sql += "SELECT id_origin_observ, CONCAT('{\"nombre_individu\":', 'null','}')::jsonb "
    sql += "FROM " + table_origin + " "
    sql += "WHERE " + nombre + " IS NULL;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans t_observation_complements]")

    # ========= Insertion dans la synthèse =========
    # ATTENTION 1000 données MAX
    # Voir le fichier SQL

    query = QSqlQuery(db)
    sql = "SELECT gn_synthese.import_row_from_table('id_source', '7',
'gn_monitoring.vs_comptage_rnmo')"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans synthese]")

    # ========= Validation =========

    query = QSqlQuery(db)
    sql = "INSERT INTO gn_commons.t_validations (uuid_attached_row, id_nomenclature_valid_status,
id_validator, validation_auto, validation_date) "
    sql += "SELECT uuid_observation, "+ statut_validation +", "+ id_validateur_origin +", false, NOW()
"
    sql += "FROM gn_monitoring.t_observations "
    sql += "INNER JOIN " + table_origin + " ON "+ table +".id_origin_observ =
t_observations.id_observation;"
    query.prepare(sql)
    if not query.exec_():
        print(query.lastError().text())
        print("[Insertion dans t_validations]")

    db.close()
    del db
    db = None
    QSqlDatabase.removeDatabase('db1')
```

ANNEXE XX : MAQUETTE DE L'INTERFACE HOMME-MACHINE DU PLUGIN « GEOPIM »

Schéma de l'Interface Homme-Machine (IHM) du futur plugin QGIS d'import massif de données

1. MENU ET FENETRE PRINCIPALE



Page 1

PLUGIN D'IMPORT DE DONNÉES MASSIVES

Choisissez le module d'insertion :

OccTax ▽

Fichier à importer (.csv, .shp) :

[ ] [...]

Jeu de données :

[ ] ▽

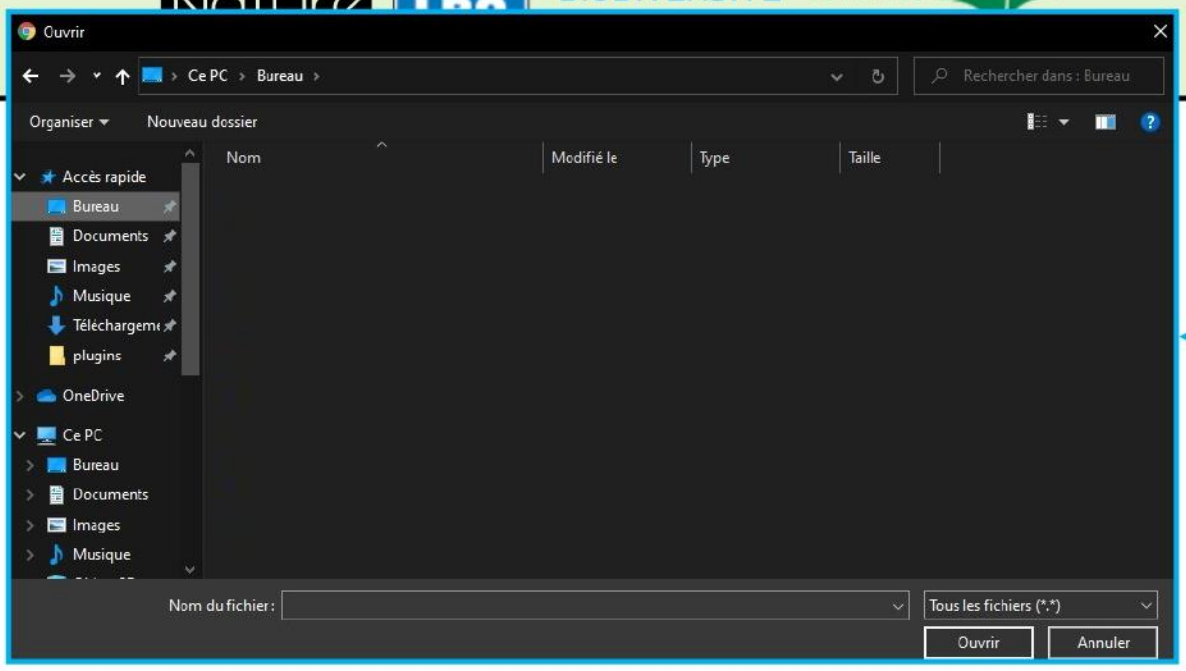
Nom de la table (de préférence rn??_jdd_jjmmaaaa) :

[ ]

[Valider] [Quitter]

PLUGIN DÉVELOPPÉ PAR MAXIME TOMA (MAXIME.TOMA@GMAIL.COM)

GEO Nature — LPO — AGIR pour la BIODIVERSITÉ — Réserves Naturelles DE FRANCE



PLUGIN D'IMPORT DE DONNÉES MASSIVES

Choisissez le module d'insertion :

OccTax ▽

Fichier à importer (.csv, .shp) :

T:\Sep\Espaces Prtoégés\ETUDES SEP PAT NAT\2020 GEONATURE\4_PLUGIN_QGIS\IMPORT\truc.csv ...

CHAMPS NON-CONFORMES: EXPLICATION

Jeu de données :

▽

Nom de la table (de préférence rn??_jdd_jjmmaaaa) :

Valider | Quitter

PLUGIN DÉVELOPPÉ PAR MAXIME TOMA (MAXIME.TOMA@GMAIL.COM)

GEO Nature | LPO | AGIR pour la BIODIVERSITÉ | Réserves Naturelles DE FRANCE

PLUGIN D'IMPORT DE DONNÉES MASSIVES

Choisissez le module d'insertion :

OccTax ▽

Fichier à importer (.csv, .shp) :

T:\Sep\Espaces Prtoégés\ETUDES SEP PAT NAT\2020 GEONATURE\4_PLUGIN_QGIS\IMPORT\truc.csv ...

XXXX LIGNES DÉTECTÉES / CHAMPS EN CONFORMITÉ

Jeu de données :

▽

Nom de la table (de préférence rn??_jdd_jjmmaaaa) :

Valider | Quitter

PLUGIN DÉVELOPPÉ PAR MAXIME TOMA (MAXIME.TOMA@GMAIL.COM)

GEO Nature | LPO | AGIR pour la BIODIVERSITÉ | Réserves Naturelles DE FRANCE

PLUGIN D'IMPORT DE DONNÉES MASSIVES

Choisissez le module d'insertion :

OccTax ▽

Fichier à importer (.csv, .shp) :

T:\Sep\Espaces Prtoégés\ETUDES SEP PAT NAT\2020 GEONATURE\4_PLUGIN_QGIS\IMPORT\truc.csv ...

XXXX LIGNES DÉTECTÉES / CHAMPS EN CONFORMITÉ

Jeu de données :

Observation aléatoire RNBH ▽

Nom de la table (de préférence rn??_jdd_jjmmaaaa) :

rnbh obs@lea/08072020

ESPACE OU CARACTÈRE SPÉCIAL DÉTECTÉ

Valider | Quitter

PLUGIN DÉVELOPPÉ PAR MAXIME TOMA (MAXIME.TOMA@GMAIL.COM)

GEO Nature | LPO | AGIR pour la BIODIVERSITÉ | Réserves Naturelles DE FRANCE

PLUGIN D'IMPORT DE DONNÉES MASSIVES

Choisissez le module d'insertion :

OccTax ▽

Fichier à importer (.csv, .shp) :

T:\Sep\Espaces Prtoégés\ETUDES SEP PAT NAT\2020 GEONATURE\4_PLUGIN_QGIS\IMPORT\truc.csv ...

XXXX LIGNES DÉTECTÉES / CHAMPS EN CONFORMITÉ

Jeu de données :

Observation aléatoire RNBH ▽

Nom de la table (de préférence rn??_jdd_jjmmaaaa) :

rnbh_obs_alea_08072020

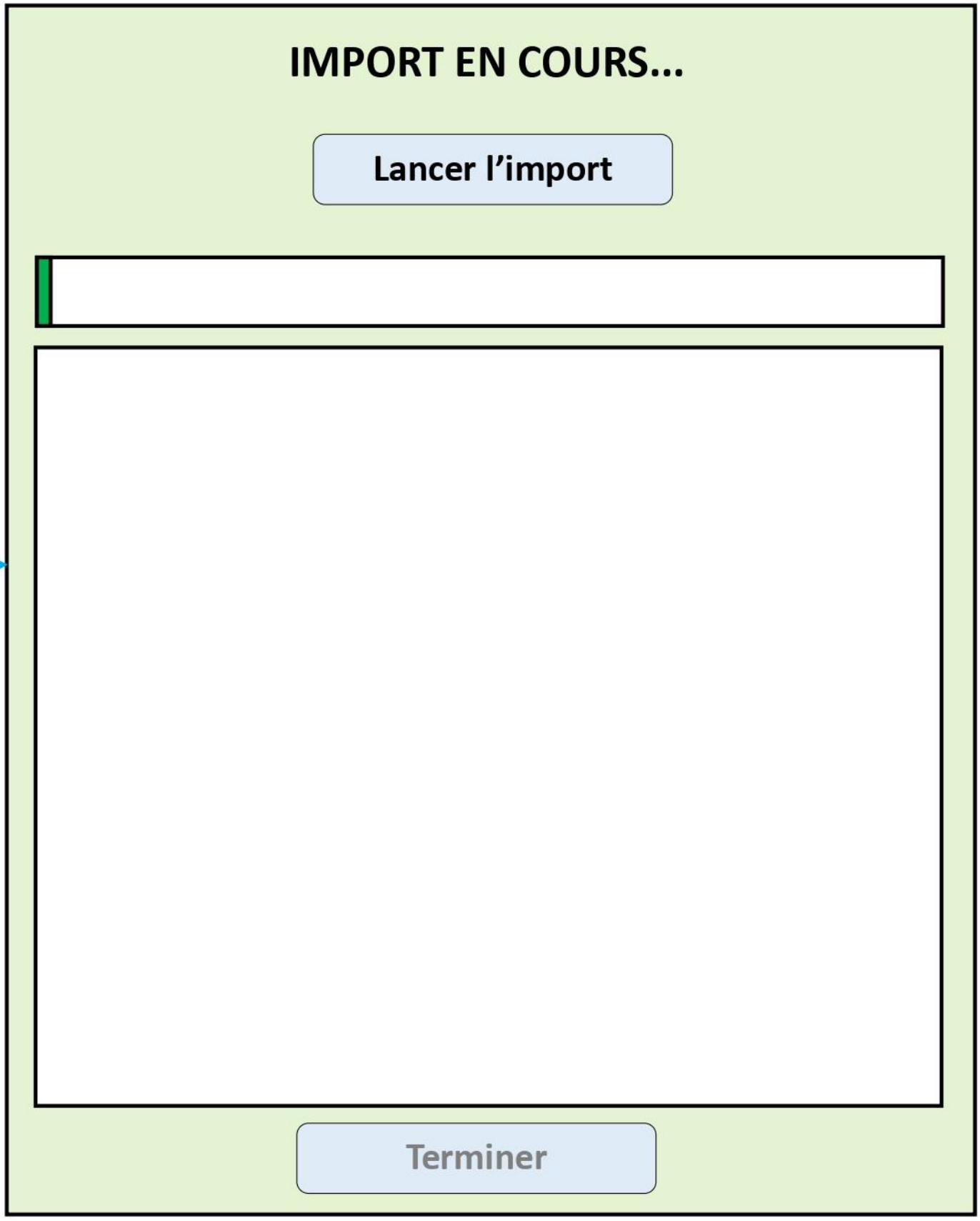
Valider **Quitter**

PLUGIN DÉVELOPPÉ PAR MAXIME TOMA (MAXIME.TOMA@GMAIL.COM)

GEO Nature LPO AGIR pour la BIODIVERSITÉ Réserves Naturelles DE FRANCE

2. IMPORT DANS OCCTAX & MONITORING

On clique sur « Lancer l'import » qui a pour conséquence de lancer le script d'intégration



IMPORT EN COURS...

Lancer l'import

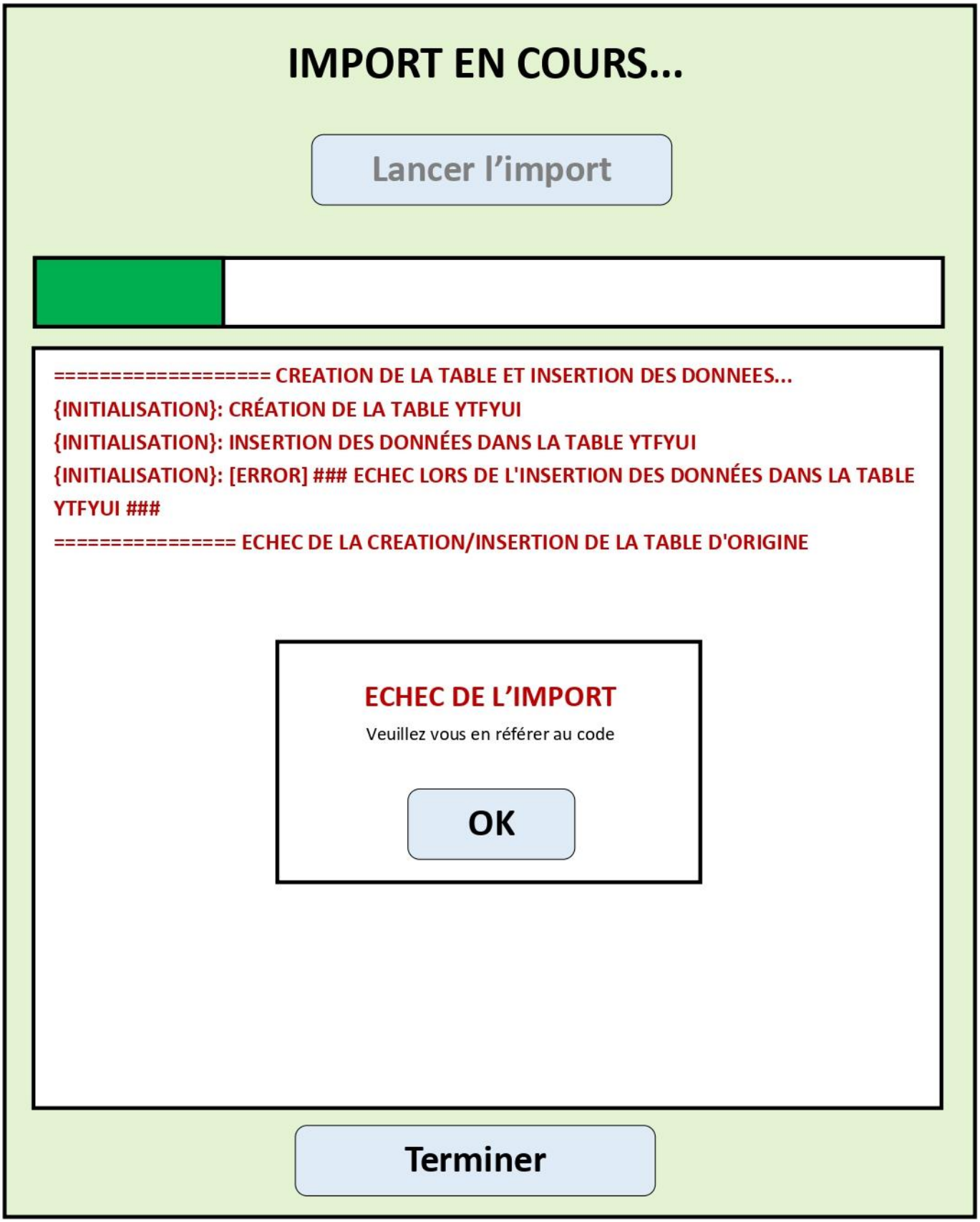
========== CREATION DE LA TABLE ET INSERTION DES DONNEES...
{INITIALISATION}: CRÉATION DE LA TABLE XDGH
{INITIALISATION}: INSERTION DES DONNÉES DANS LA TABLE XDGH
{INITIALISATION}: DÉFINITION DE LA GÉOMÉTRIE À PARTIR DU X/Y DE LA TABLE XDGH
========== INSERTION EN COURS...
{INITIALISATION}: REM PAR LEUR ID
{INITIALISATION}: MA
{RELEVÉ}: INSERTION D
{RELEVÉ}: DÉFINITION _OCCTAX
{OBSERVATEURS}: INS RS/RELEVÉ
{OCCURRENCES}: RÉCU
{OCCURRENCES}: INSE TAXON - DE 255 CARA-TÈRES)
{OCCURRENCES}: DÉFINITION DE LA CORRESPONDANCE XDGH / T_OCCURENCES_OCCTAX
{OCCURRENCES}: INSERTION DES OCCURRENCES DE RELEVÉ (NOM_TAXON + DE 255 CARA-TÈRES)
{OCCURRENCES}: DÉFINITION DE LA CORRESPONDANCE XDGH / T_OCCURENCES_OCCTAX
{COUNTING}: INSERTION DU DÉNOMBREMENT DES OCCURRENCES DE RELEVÉ
{VALIDATION}: APPLICATION DE LA VALIDATION POUR LES OCCURRENCES CONCERNÉES
{HISTORIQUE}: HISTORISATION DE L'IMPORT
========== INSERTION TERMINÉE

SUCCES DE L'IMPORT

Les données ont été importés avec succès

OK

Terminer



Le code en rouge donne alors le/les endroits où le script a eu une erreur.

3. HISTORIQUE DES IMPORTS

Vers Page 1

HISTORIQUE DES IMPORTS

Nom de la table	Module	Jeu de données	Date d'import
rnbh_obs_alea_08070220	OccTax	Observation Aléatoire RNBH	2020-07-08 14:50:00

= **Supprime la table d'import uniquement (pour alléger la BDD)** mais il sera impossible après de supprimer les données insérées dans le module en question

= Supprime la table d'import ET les données insérées dans le module en question

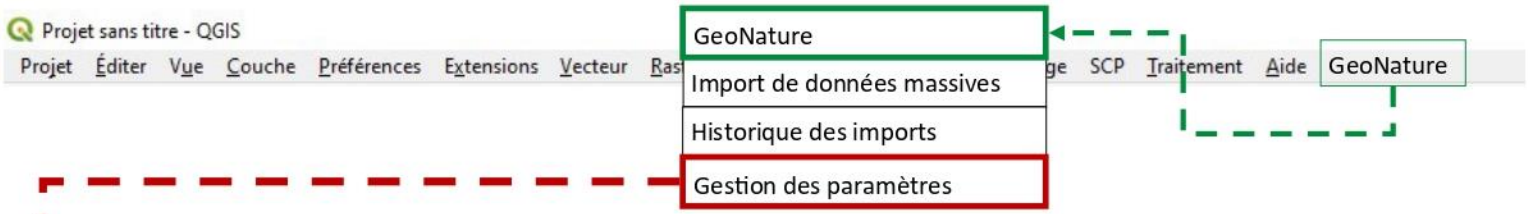
L'import et les données associées ont bien été supprimés

OK

Confirmez-vous la suppression de cet import ?

Oui **Non**

4. GESTION DES PARAMETRES



GESTION DES PARAMETRES

Connexion à la base de données :

Hôte :

000.000.0.00

Nom de la base de données :

geonature2db

Utilisateur :

geonatureadmin

Mot de passe :

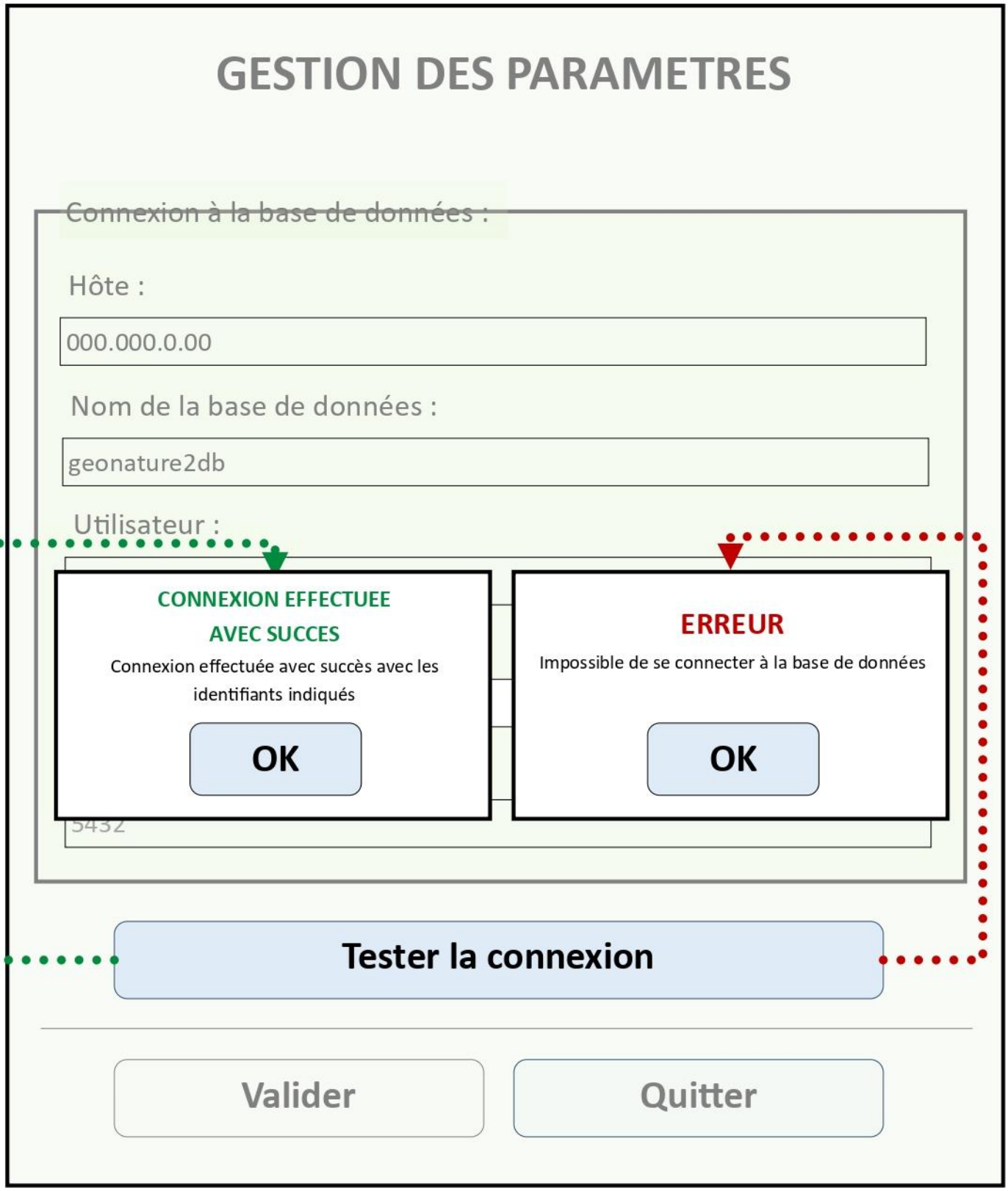
****

Port :

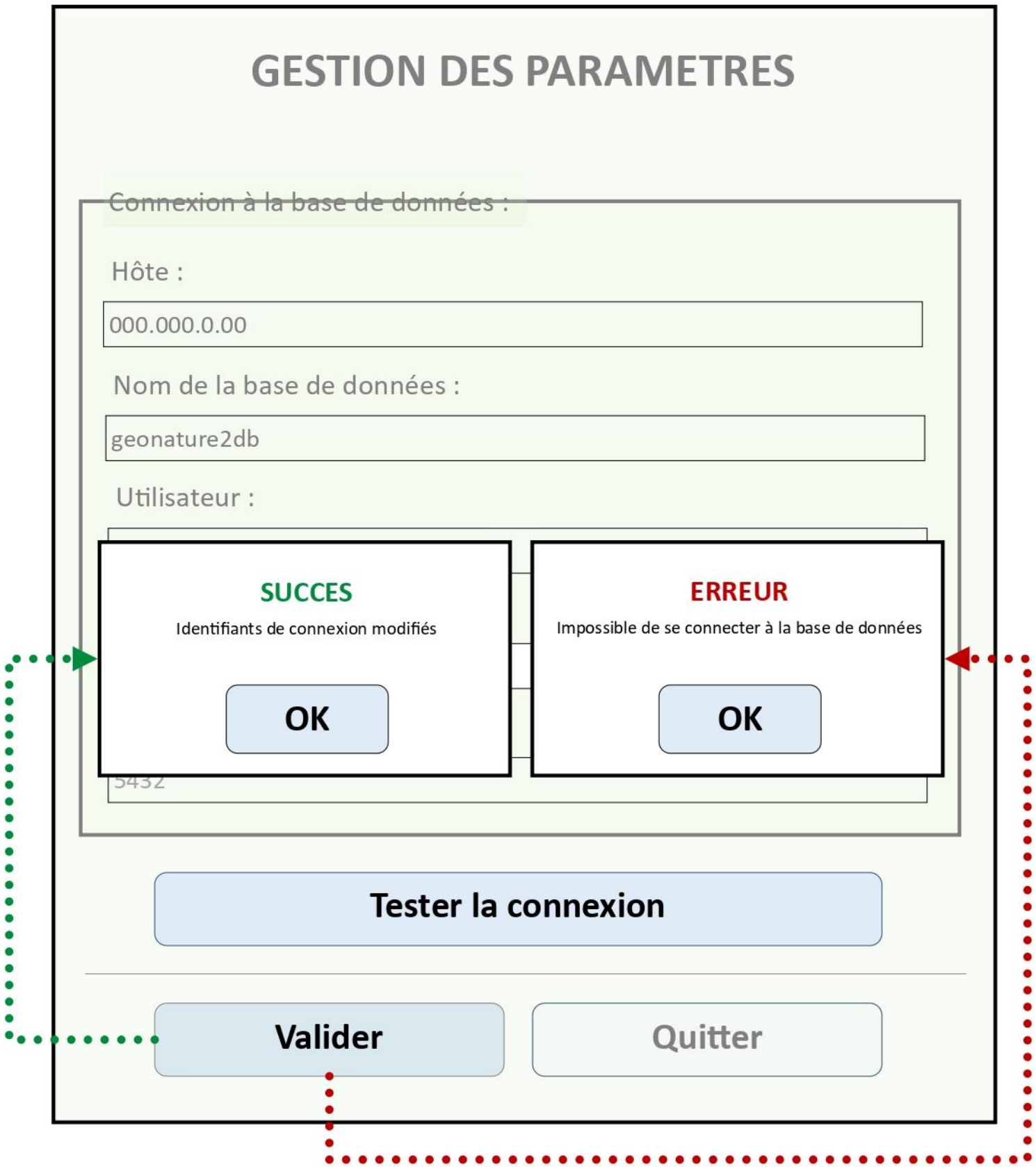
5432

Tester la connexion

Valider **Quitter**



Avant de valider la modification, on doit cliquer sur le bouton « Tester la connexion ».
Si elle est valide, le bouton « Valider » devient cliquable.



Une valeur peut être modifié entre le moment de la vérification et d'appuyer sur le bouton « Valider » Ainsi, on revérifie si la connexion peut se faire.
Si rien n'a été touché, seul le message en vert s'affiche. Dans le cas contraire, on réaffichera les messages affichés par le test de connexion avant d'afficher le message en vert.

ANNEXE XXI : SCRIPT PYTHON PRINCIPAL DU PLUGIN

```
# -*- coding: utf-8 -*-
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtSql import *
from qgis.core import *
import os

from .import_dialog import *
from .historique_dialog import *
from .gestion_param_dialog import *
from .param_bdd import *

class GeoNatureLPOImportPlugin:
    def __init__(self, iface):
        self.interface = iface

    def initGui(self):
        # ================================================/ Définition des QActions
/================================================
        iconImport = QIcon(os.path.dirname(__file__) + "/icons/import_donnees2.png")
        self.actionImport = QAction(iconImport, u"Import de données", self.interface.mainWindow())
        self.actionImport.triggered[bool].connect(self.gereImport)

        iconHistorique = QIcon(os.path.dirname(__file__) + "/icons/historique_imports.png")
        self.actionHistorique = QAction(iconHistorique, u"Historique des imports",
self.interface.mainWindow())
        self.actionHistorique.triggered[bool].connect(self.gereHistorique)

        iconParam = QIcon(os.path.dirname(__file__) + "/icons/gestion_param.png")
        self.actionParam = QAction(iconParam, u"Gestion des paramètres", self.interface.mainWindow())
        self.actionParam.triggered[bool].connect(self.gereParam)

        # ================================================/ Définition du QMenu
/================================================
        self.menuGeoPIM = QMenu(u"Import GeoNature")
        self.menuGeoPIM.addAction(self.actionImport)
        self.menuGeoPIM.addAction(self.actionHistorique)
        self.menuGeoPIM.addAction(self.actionParam)

        # Emplacement menu

self.interface.mainWindow().menuBar().insertMenu(self.interface.firstRightStandardMenu().menuAction(),s
elf.menuGeoPIM)

        # ================================================/ Définition de la ToolBar
/================================================
        self.toolbarGeoPIM = self.interface.addToolBar(u"Import GeoNature");
        self.toolbarGeoPIM.setObjectName("barreOutilImportGeoNature")
        self.toolbarGeoPIM.addAction(self.actionImport)
        self.toolbarGeoPIM.addSeparator()
        self.toolbarGeoPIM.addAction(self.actionHistorique)


    def unload(self):
        self.interface.mainWindow().menuBar().removeAction(self.menuGeoPIM.menuAction())
        self.interface.mainWindow().removeToolBar(self.toolbarGeoPIM)

    def gereImport(self):
        # ================================================/ Ouverture de l'action "Import de données"
/================================================
        dbType = "postgres"
        db = QSqlDatabase.addDatabase("QPSQL", "db1")
        db.setDatabaseName(db_nom)
```

```
        db.setUserName(db_user)
        db.setPassword(db_password)
        db.setHostName(db_hote)
        db.setPort(db_port)

        dbSchema = "lpo_imports"

        if (not db.open()):
            msg = QMessageBox()
            msg.setIcon(QMessageBox.Critical)
            msg.setText("Erreur")
            msg.setInformativeText(u"Impossible de se connecter à la base de données ...")
            msg.setWindowTitle("Erreur")
            msg.exec_()
        else:
            dlg = ImportDialog(db, dbType, dbSchema)
            dlg.show()  # ligne à mettre en commentaire pour avoir une fenêtre modale
            result = dlg.exec_()
            if result:
                pass
            if db.isOpen():
                db.close()

    def gereHistorique(self):
        # ================================================/ Ouverture de l'action "Historique des
imports" /================================================
        dbType = "postgres"
        db = QSqlDatabase.addDatabase("QPSQL", "db1")
        db.setDatabaseName(db_nom)
        db.setUserName(db_user)
        db.setPassword(db_password)
        db.setHostName(db_hote)
        db.setPort(db_port)

        dbSchema = "lpo_imports"

        if (not db.open()):
            msg = QMessageBox()
            msg.setIcon(QMessageBox.Critical)
            msg.setText("Erreur")
            msg.setInformativeText(u"Impossible de se connecter à la base de données ...")
            msg.setWindowTitle("Erreur")
            msg.exec_()
        else:
            dlg = HistoriqueDialog(db, dbType, dbSchema)
            dlg.show()  # ligne à mettre en commentaire pour avoir une fenêtre modale
            result = dlg.exec_()
            if result:
                pass
            if db.isOpen():
                db.close()

    def gereParam(self):
        # ================================================/ Ouverture de l'action "Gestion des
paramètres" /================================================
        dlg = GestionParamDialog()
        dlg.show()  # ligne à mettre en commentaire pour avoir une fenêtre modale
        result = dlg.exec_()
        if result:
            pass
```

ANNEXE XXII : SCRIPT PYTHON D'INSERTION DE DONNEES DANS OCCTAX

```
# -*- coding: utf-8 -*-
import sys, os

from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtSql import *

from qgis.core import *

sys.path.append(os.path.dirname(os.path.abspath(__file__)) + "/forms")
from .forms.import_ajout_form import *

class ImportOcctaxDialog(QDialog, Ui_import_ajout_form):
    def __init__(self, db, dbType, dbSchema, moduleid, modulenom, jddid, jddnom, nomtable,
cheminfichier):
        QWidget.__init__(self)
        self.db = db
        self.dbType = dbType
        self.dbSchema = dbSchema
        self.moduleid = moduleid
        self.modulenom = modulenom
        self.jddid = jddid
        self.jddnom = jddnom
        self.nomtable = nomtable
        self.fichier = cheminfichier
        self.setupUi(self)

        self.progressBar_code.setValue(0)
        self.ifErrorOnInsertIntoGeoNature = -1 # Testeur si une erreur intervient dans l'import dans le
module

        # ================================================/ Signaux & Slots
/================================================
        self.pb_lancer.clicked.connect(self.creationTable)
        self.pb_terminer.clicked.connect(self.close)

    def suppressionApresEchec(self):
        if self.ifErrorOnInsertIntoGeoNature != -1: # Seulement si il y a eu des insertions à partir de
la table t_releves_occtax
            self.titre.setText("Echec de l'import")
            self.textEdit_code.append("======================== ECHEC DE L'INSERTION
========================")
            self.textEdit_code.setStyleSheet("background-color: white; color: red;")

            # Suppression cascade des relevés
            query = QSqlQuery(self.db)
            sql = "DELETE FROM pr_occtax.t_releves_occtax WHERE id_releve_occtax IN (SELECT
id_releve_occtax FROM lpo_imports." + self.nomtable + ");"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Echec de l'import}: [ERROR] ### Erreur lors de la
suppression des données ###")
                self.textEdit_code.append(query.lastError().text())

        # Suppression table dans lpo_imports
        query = QSqlQuery(self.db)
        sql = "DROP TABLE lpo_imports." + self.nomtable + ";"
        if not query.exec_(sql):
            self.textEdit_code.append("{Echec de l'import}: [ERROR] ### Erreur lors de la suppression
de la table dans lpo_imports ###")
```

```
            self.textEdit_code.append(query.lastError().text())

        msg = QMessageBox()
        msg.setIcon(QMessageBox.Warning)
        msg.setText("Echec de l'insertion des données")
        msg.setInformativeText("Veuillez vous référer au déroulement du code.")
        msg.setWindowTitle("Echec")
        msg.exec_()

        self.pb_terminer.setEnabled(True)

    def creationTable(self):
        self.titre.setText("Création table et insertion des données...")
        self.pb_lancer.setEnabled(False)
        self.textEdit_code.setText("================== CREATION DE LA TABLE ET INSERTION DES
DONNEES... ==================")

        openInsert = -1 # Testeur si une erreur intervient dans la création des tables avant l'import

        # Create puis insert dans la base
        self.textEdit_code.append("{Initialisation}: Création de la table "+self.nomtable+"")
        query = QSqlQuery(self.db)
        sql = "CREATE TABLE IF NOT EXISTS lpo_imports.%s (id serial PRIMARY KEY, id_releve integer,
geom geometry(POINT), date_debut timestamp, date_fin timestamp, heure_debut time, heure_fin time,
x_wgs84 double precision, y_wgs84 double precision, site character varying(255), " % (self.nomtable)
        sql += "site_auto character varying(255), precipoint character varying(255), observateu
character varying(255), obs2 character varying(255), obs3 character varying(255), obs4 character
varying(255), commentrel text, nom_taxon text, nom_vernaculaire text, code_taxon integer, methobs
character varying(255), etatbio character varying(255), "
        sql += "statutbio character varying(255), nivnatural character varying(255), preuvexist
character varying(255), statutobs character varying(255), statutsrce character varying(255), determinat
character varying(255), "
        sql += "methdeterm character varying(255), preuvdig character varying(255), preuvndig character
varying(255), commentocc text, nb_min integer, nb_max integer, sexe character varying(255), stade_vie
character varying(255), "
        sql += "objdenombr character varying(255), typdenombr character varying(255), statutvalid
character varying(255), validateur character varying(255), datevalid timestamp, commentval text,
id_releve_occtax integer, id_occurrence_occtax integer);"
        if not query.exec_(sql):
            openInsert += 1
            self.titre.setText("Echec de la création de la table")
            self.textEdit_code.append("================ ECHEC DE LA CREATION/INSERTION DE LA TABLE
D'ORIGINE ================")
            self.textEdit_code.setStyleSheet("background-color: white; color: red;")
            self.pb_terminer.setEnabled(True)

        if openInsert == -1:
            self.textEdit_code.append("{Initialisation}: Insertion des données dans la table " +
self.nomtable + "")
            batch = "set PGPASSWORD=cncmax09&& D:\Applications\PostgreSQL\\11\\bin\psql.exe -h
192.168.1.29 -d geonature2db -U geonatureadmin -c "
            batch += "\"\copy lpo_imports.%s" % (self.nomtable)
            batch += "(id, id_releve, date_debut, date_fin, heure_debut, heure_fin, x_wgs84, y_wgs84,
site, site_auto, "
            batch += "precipoint, observateu, obs2, obs3, obs4, commentrel, nom_taxon,
nom_vernaculaire, code_taxon, "
            batch += "methobs, etatbio, statutbio, nivnatural, preuvexist, statutobs, statutsrce, "
            batch += "determinat, methdeterm, preuvdig, preuvndig, commentocc, nb_min, nb_max, sexe,
stade_vie, "
            batch += "objdenombr, typdenombr, statutvalid, validateur, datevalid, commentval) "
            batch += "FROM '%s' DELIMITER ';' CSV HEADER ENCODING 'UTF8';" % (self.fichier)

            if os.system('cmd /c "'+batch+'"') != 0:
                openInsert += 1
                self.textEdit_code.append("{Initialisation}: [ERROR] ### Echec lors de l'insertion des
données dans la table "+self.nomtable+" ###")
```

```
                self.titre.setText("Echec de l'insertion des données")
                self.textEdit_code.append("================ ECHEC DE LA CREATION/INSERTION DE LA TABLE
D'ORIGINE ================")
                self.textEdit_code.setStyleSheet("background-color: white; color: red;")

                self.suppressionApresEchec()

        if openInsert == -1:
            # Création de la geométrie
            self.textEdit_code.append("{Initialisation}: Définition de la géométrie à partir du x/y de
la table "+self.nomtable+"")
            query = QSqlQuery(self.db)
            sql = "UPDATE lpo_imports.%s SET geom = ST_SetSRID(ST_Point(x_wgs84, y_wgs84), 4326)" %
(self.nomtable)
            query.prepare(sql)
            if not query.exec_():
                openInsert += 1
                self.textEdit_code.append("{Initialisation}: [ERROR] ### Echec lors de la création de
la géométrie à partir du X/Y de la table "+self.nomtable+" ###")
                self.textEdit_code.append(query.lastError().text())
                self.titre.setText("Echec de conversion de la geom")
                self.textEdit_code.append("================ ECHEC DE LA CREATION/INSERTION DE LA TABLE
D'ORIGINE ================")
                self.textEdit_code.setStyleSheet("background-color: white; color: red;")

                self.suppressionApresEchec()

        if openInsert == -1:
            self.insertOcctax()



    def insertOcctax(self):
        self.titre.setText("Import en cours...")
        # self.pb_lancer.setEnabled(False)
        self.textEdit_code.append("======================== INSERTION EN COURS...
========================")
        self.ifErrorOnInsertIntoGeoNature = -1 # Comptage du nombre d'erreurs

        # ================================================/ Remplacement des choix de liste
déroulantes en ID /================================================
        self.textEdit_code.append("{Initialisation}: Remplacement du texte des listes de choix par leur
id")
        self.progressBar_code.setValue(10)

        # [TABLES]
        schema = self.dbSchema
        table = self.nomtable
        table_origin = schema + "." + table
        id_unique = "id"
        id_origin_duplique = "id_releve"  # ID dupliqué de la table d'origine

        query = QSqlQuery(self.db)

        # Observateurs

        sql = "UPDATE "+ table_origin +" SET observateu = id_role "
        sql += "FROM utilisateurs.t_roles "
        sql += "WHERE observateu = CONCAT(UPPER(nom_role), ' ', prenom_role) AND id_role IS NOT NULL; "

        sql += "UPDATE "+ table_origin +" SET obs2 = id_role "
        sql += "FROM utilisateurs.t_roles "
        sql += "WHERE obs2 = CONCAT(UPPER(nom_role), ' ', prenom_role) AND id_role IS NOT NULL; "

        sql += "UPDATE "+ table_origin +" SET obs3 = id_role "
        sql += "FROM utilisateurs.t_roles "
```

```
sql += "WHERE obs3 = CONCAT(UPPER(nom_role), ' ', prenom_role) AND id_role IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET obs4 = id_role "
sql += "FROM utilisateurs.t_roles "
sql += "WHERE obs4 = CONCAT(UPPER(nom_role), ' ', prenom_role) AND id_role IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET validateur = id_role "
sql += "FROM utilisateurs.t_roles "
sql += "WHERE validateur = CONCAT(UPPER(nom_role), ' ', prenom_role) AND id_role IS NOT NULL; "

# Nommenclatures

sql += "UPDATE "+ table_origin +" SET methObs = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE methObs = label_default AND id_type = 14 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET etatBio = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE etatBio = label_default AND id_type = 7 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET statutBio = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE statutBio = label_default AND id_type = 13 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET nivNatural = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE nivNatural = label_default AND id_type = 8 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET preuvExist = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE preuvExist = label_default AND id_type = 15 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET statutObs = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE statutObs = label_default AND id_type = 18 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET statutSrce = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE statutSrce = label_default AND id_type = 19 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET methDeterm = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE methDeterm = label_default AND id_type = 106 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET sexe = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE sexe = label_default AND id_type = 9 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET stade_vie = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE stade_vie = label_default AND id_type = 10 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET objDenombr = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE objDenombr = label_default AND id_type = 6 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET typDenombr = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE typDenombr = label_default AND id_type = 21 AND id_nomenclature IS NOT NULL; "

sql += "UPDATE "+ table_origin +" SET statutValid = id_nomenclature "
sql += "FROM ref_nomenclatures.t_nomenclatures "
sql += "WHERE statutValid = label_default AND id_type = 101 AND id_nomenclature IS NOT NULL; "

query.prepare(sql)
if not query.exec_():
```

```
            self.textEdit_code.append("{Initialisation}: [ERROR] ### Erreur lors du remplacement des
noms par les ids ###")
            self.textEdit_code.append(query.lastError().text())
            self.suppressionApresEchec()

        if self.ifErrorOnInsertIntoGeoNature == -1: # Test si pas d'erreurs
            self.textEdit_code.append("{Initialisation}: Mapping des champs")
            self.progressBar_code.setValue(20)

        # ================================================/ Définition des variables aléatoires
/================================================

        # [RELEVES]
        id_dataset = "%s" % (self.jddid)  # ID du Jeu de données
        id_digitiser = "1"  # 17 = Maxime TOMA // Plugin 1 = Admin
        date_debut = "date_debut::timestamp"  # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp
        date_fin = "date_fin::timestamp"  # TIMESTAMP 'aaaa-mm-jj 00:00:00' // ou //
nom_colonne_date::timestamp
        hour_min = "heure_debut::time"
        hour_max = "heure_fin::time"
        commentaire_releve = "CONCAT('Commentaire: ',commentrel,' | Précision du point: ',precipoint)"
# 'blablabla' // ou // nom_colonne_commentaire // ou // CONCAT(...)

        group_by_insert_releve = "GROUP BY id_releve, geom, date_debut, date_fin, heure_debut,
heure_fin, commentrel, precipoint"

        # [OBSERVATEURS]
        colonne_observateur_1 = "observateu::integer"
        colonne_observateur_2 = "obs2::integer"
        colonne_observateur_3 = "obs3::integer"
        colonne_observateur_4 = "obs4::integer"

        # [OCCURENCES]
        methode_observation = "methobs::integer"  # {NOT NULL}
        etat_biologique = "etatbio::integer"  # {NOT NULL}
        statut_biologique = "statutbio::integer"
        niveau_de_naturalite = "nivnatural::integer"
        preuve_existance = "preuvexist::integer"
        statut_observation = "statutobs::integer"
        floutage_donnee = "175"
        statut_source = "statutsrce::integer"
        determinateur = "determinat"
        methode_determination = "methdeterm::integer"
        cd_nom_taxon = "code_taxon::integer"
        detail_preuve_digital = "preuvdig"
        detail_preuve_non_digital = "preuvndig"
        commentaire_occurrences = "commentocc"

        # [COUNTING]
        stade_vie = "stade_vie::integer"  # {NOT NULL}
        sexe = "sexe::integer"  # {NOT NULL}
        objet_denombrement = "objdenombr::integer"  # {NOT NULL}
        type_denombrement = "typdenombr::integer"
        nb_min = "nb_min"
        nb_max = "nb_max"

        # [VALIDATION]
        statut_validation = "statutvalid::integer"  # 0(465) = En attente de validation / 1(317) =
Certain, Très problable / 2(318) = Probable / 3(319) = Douteux / 4(320) = Invalide / 5(321) = Non-
réalisable / 6(322) = Inconnu // ou // nom_colonne
        id_validateur_origin = "validateur::integer"  # Paul TROTIGNON
        commentaire_validation = "commentval"  # 'blablabla' // ou // nom_colonne_commentaire // ou //
CONCAT(...)

        if self.ifErrorOnInsertIntoGeoNature == -1:
```

```
            # ================================================/ Insertion des relevés
/================================================
            self.textEdit_code.append("{Relevé}: Insertion des relevés")
            self.progressBar_code.setValue(30)

            query = QSqlQuery(self.db)
            sql = "INSERT INTO pr_occtax.t_releves_occtax (id_dataset, id_digitiser,
id_nomenclature_obs_technique, id_nomenclature_grp_typ, date_min, date_max, hour_min, hour_max,
meta_device_entry, comment, geom_local, geom_4326, precision) "
            sql += "SELECT " + id_dataset + ", " + id_digitiser + ", 316, 132, " + date_debut + ", " +
date_fin + ", " + hour_min + ", " + hour_max + ", 'script python', " + commentaire_releve + ",
ST_Transform(geom, 2154) AS geom2154, geom, 100 "
            sql += "FROM " + table_origin + " "
            sql += group_by_insert_releve + " "
            sql += "ORDER BY " + id_origin_duplique + " RETURNING id_releve_occtax;"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Relevé}: [ERROR] ### Erreur lors de l'insertion des relevés
###")
                self.textEdit_code.append(query.lastError().text())



            query2 = QSqlQuery(self.db)  # On selectionne juste les id en respectant les conditions
précédentes et le ORDER BY
            sql2 = "SELECT " + id_origin_duplique + " FROM " + table_origin + " "
            sql2 += "GROUP BY " + id_origin_duplique + " "
            sql2 += "ORDER BY " + id_origin_duplique + ";"
            query2.prepare(sql2)
            if not query2.exec_():
                self.textEdit_code.append("{Relevé}: [ERROR] ### Erreur lors de la sélection de
l'id_releve ###")
                self.textEdit_code.append(query2.lastError().text())



            # Définition de la correspondance id_origin / id_station et insertion dans une liste de
tuples
            self.textEdit_code.append("{Relevé}: Définition de la correspondance "+table+" /
t_releves_occtax")
            self.progressBar_code.setValue(40)

            liste = []
            while query.next() and query2.next():
                queryIdReleve = query.value(0)
                queryIdOrigin = query2.value(0)
                tuple = (queryIdOrigin, queryIdReleve)
                liste.append(tuple)

            # Pour tous les éléments de la liste,  nous allons UPDATE la colonne id_station de la
table_origine et faire ainsi la correspondance

            for elt in liste:
                id_origin = elt[0]
                id_releve_occtax = elt[1]

                query = QSqlQuery(self.db)
                sql = "UPDATE " + table_origin + " SET id_releve_occtax = ? "
                sql += "WHERE " + id_origin_duplique + " = ?;"
                query.prepare(sql)
                query.addBindValue(id_releve_occtax)
                query.addBindValue(id_origin)
                if not query.exec_():
                    self.textEdit_code.append("{Relevé}: [ERROR] ### Erreur lors de l'insertion des
id_releve_occtax dans "+table+" ###")
                    self.textEdit_code.append(query.lastError().text())
```

```
                    self.ifErrorOnInsertIntoGeoNature += 1
                    self.suppressionApresEchec()

        if self.ifErrorOnInsertIntoGeoNature == -1:
            # ================================================/ Correspondance Observateur/Relevé
/=================================================
            self.textEdit_code.append("{Observateurs}: Insertion et correspondance
observateurs/relevé")
            self.progressBar_code.setValue(50)

            query = QSqlQuery(self.db)
            sql = "INSERT INTO pr_occtax.cor_role_releves_occtax (id_releve_occtax, id_role) "
            sql += "SELECT DISTINCT id_releve_occtax, "+colonne_observateur_1+" "
            sql += "FROM "+table_origin+" "
            sql += "WHERE "+colonne_observateur_1+" IS NOT NULL;"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Observateurs}: [ERROR] ### Erreur lors de l'insertion
d'observateurs (column 1) dans cor_role_releves_occtax ###")
                self.textEdit_code.append(query.lastError().text())
                self.ifErrorOnInsertIntoGeoNature += 1



            query = QSqlQuery(self.db)
            sql = "INSERT INTO pr_occtax.cor_role_releves_occtax (id_releve_occtax, id_role) "
            sql += "SELECT DISTINCT id_releve_occtax, " + colonne_observateur_2 + " "
            sql += "FROM " + table_origin + " "
            sql += "WHERE " + colonne_observateur_2 + " IS NOT NULL;"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Observateurs}: [ERROR] ### Erreur lors de l'insertion
d'observateurs (column 2) dans cor_role_releves_occtax ###")
                self.textEdit_code.append(query.lastError().text())
                self.ifErrorOnInsertIntoGeoNature += 1



            query = QSqlQuery(self.db)
            sql = "INSERT INTO pr_occtax.cor_role_releves_occtax (id_releve_occtax, id_role) "
            sql += "SELECT DISTINCT id_releve_occtax, " + colonne_observateur_3 + " "
            sql += "FROM " + table_origin + " "
            sql += "WHERE " + colonne_observateur_3 + " IS NOT NULL;"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Observateurs}: [ERROR] ### Erreur lors de l'insertion
d'observateurs (column 3) dans cor_role_releves_occtax ###")
                self.textEdit_code.append(query.lastError().text())
                self.ifErrorOnInsertIntoGeoNature += 1



            query = QSqlQuery(self.db)
            sql = "INSERT INTO pr_occtax.cor_role_releves_occtax (id_releve_occtax, id_role) "
            sql += "SELECT DISTINCT id_releve_occtax, " + colonne_observateur_4 + " "
            sql += "FROM " + table_origin + " "
            sql += "WHERE " + colonne_observateur_4 + " IS NOT NULL;"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Observateurs}: [ERROR] ### Erreur lors de l'insertion
d'observateurs (column 4) dans cor_role_releves_occtax ###")
                self.textEdit_code.append(query.lastError().text())
                self.ifErrorOnInsertIntoGeoNature += 1
```

```
        if self.ifErrorOnInsertIntoGeoNature != -1:
            self.suppressionApresEchec()
        elif self.ifErrorOnInsertIntoGeoNature == -1:
            # ================================================/ Récupération de la version de TaxRef
pour insertion /================================================
            self.textEdit_code.append("{Occurrences}: Récupération de la version de TaxRef")
            self.progressBar_code.setValue(60)

            query = QSqlQuery(self.db)
            sql = "SELECT parameter_value FROM gn_commons.t_parameters "
            sql += "WHERE parameter_name = 'taxref_version';"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Occurrences}: [ERROR] ### Erreur lors de la récupération de
la version de TaxRef ###")
                self.textEdit_code.append(query.lastError().text())
                self.ifErrorOnInsertIntoGeoNature += 1
                self.suppressionApresEchec()


            taxref_version = None
            if query.next():
                taxref_version = query.value(0)

        if self.ifErrorOnInsertIntoGeoNature == -1:
            # ================================================/ Insertion des occurences
/================================================
            self.textEdit_code.append("{Occurrences}: Insertion des occurrences de relevé (nom_taxon -
de 255 caratères)")
            self.progressBar_code.setValue(70)

            query = QSqlQuery(self.db)
            sql = "INSERT INTO pr_occtax.t_occurrences_occtax (id_releve_occtax,
id_nomenclature_obs_meth, id_nomenclature_bio_condition, id_nomenclature_bio_status,
id_nomenclature_naturalness, id_nomenclature_exist_proof, "
            sql += "id_nomenclature_observation_status, id_nomenclature_blurring,
id_nomenclature_source_status, determiner, id_nomenclature_determination_method, cd_nom, nom_cite,
meta_v_taxref, digital_proof, non_digital_proof, comment) "
            sql += "SELECT " + table + ".id_releve_occtax, " + methode_observation + ", " +
etat_biologique + ", " + statut_biologique + ", " + niveau_de_naturalite + ", " + preuve_existance + ",
" + statut_observation + ", "
            sql += floutage_donnee + ", " + statut_source + ", " + determinateur + ", " +
methode_determination + ", "
            sql += table + "." + cd_nom_taxon + ", CONCAT(taxref.nom_vern,' = <i> ',
taxref.nom_complet, '</i> - [', taxref.id_rang, ' - ', taxref.cd_nom ,']') AS nom_cite, '" +
taxref_version + "', " + detail_preuve_digital + ", " + detail_preuve_non_digital + ", " +
commentaire_occurrences + " "
            sql += "FROM " + table_origin + " "
            sql += "LEFT JOIN pr_occtax.t_releves_occtax ON " + table + ".id_releve =
t_releves_occtax.id_releve_occtax "
            sql += "LEFT JOIN taxonomie.taxref ON "+ table +"."+ cd_nom_taxon +" = taxref.cd_nom "
            sql += "GROUP BY " + table + "." + id_unique + ", " + table + ".id_releve_occtax, " +
methode_observation + ", " + etat_biologique + ", " + statut_biologique + ", " + niveau_de_naturalite +
", " + preuve_existance + ", " + statut_observation + ", "
            sql += statut_source + ", " + determinateur + ", " + methode_determination + ", " + table +
"." + cd_nom_taxon + ", taxref.nom_vern, taxref.nom_complet, taxref.id_rang, taxref.cd_nom, " +
detail_preuve_digital + ", " + detail_preuve_non_digital + ", " + commentaire_occurrences + " "
            sql += "HAVING LENGTH(CONCAT(taxref.nom_vern,' = <i> ', taxref.nom_complet, '</i> - [',
taxref.id_rang, ' - ', taxref.cd_nom ,']')) <= 255 "
            sql += "ORDER BY " + table + "." + id_unique + " RETURNING id_occurrence_occtax;"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Occurrences}: [ERROR] ### Erreur lors de l'insertion des
occurrences de relevé ###")
                self.textEdit_code.append(sql)
                self.textEdit_code.append(query.lastError().text())
```

```
                self.ifErrorOnInsertIntoGeoNature += 1


            query2 = QSqlQuery(self.db)  # On selectionne juste les id en respectant les conditions
précédentes et le ORDER BY
            sql2 = "SELECT " + table + "." + id_unique + " FROM " + table_origin + ", taxonomie.taxref
"
            sql2 += "WHERE " + cd_nom_taxon + " = taxref.cd_nom "
            sql2 += "GROUP BY " + table + "." + id_unique + ", taxref.nom_vern, taxref.nom_complet,
taxref.id_rang, taxref.cd_nom "
            sql2 += "HAVING LENGTH(CONCAT(taxref.nom_vern,' = <i> ', taxref.nom_complet, '</i> - [',
taxref.id_rang, ' - ', taxref.cd_nom ,']')) <= 255 "
            sql2 += "ORDER BY " + table + "." + id_unique + ";"
            query2.prepare(sql2)

               if not query2.exec_():
                self.textEdit_code.append("{Occurrences}: [ERROR] ### Erreur lors de la sélection de
l'id unique de "+table+" ###")
                self.textEdit_code.append(query2.lastError().text())
                self.ifErrorOnInsertIntoGeoNature += 1

            self.textEdit_code.append("{Occurrences}: Définition de la correspondance " + table + " /
t_occurences_occtax")
            self.progressBar_code.setValue(80)



            liste2 = []  # Définition de la correspondance id_unique / id_occurrence_occtax et
insertion dans une liste de tuples
            while query.next() and query2.next():
                queryIdOccurrence = query.value(0)
                queryIdUnique = query2.value(0)
                tuple = (queryIdUnique, queryIdOccurrence)
                liste2.append(tuple)

            # Pour tous les éléments de la liste,  nous allons UPDATE la colonne id_occurence_occtax de
la table_origine et faire ainsi la correspondance

            for elt2 in liste2:
                id_origin = elt2[0]
                id_occurrence_occtax = elt2[1]

                query = QSqlQuery(self.db)
                sql = "UPDATE " + table_origin + " SET id_occurrence_occtax = ? "
                sql += "WHERE " + id_unique + " = ?;"
                query.prepare(sql)
                query.addBindValue(id_occurrence_occtax)
                query.addBindValue(id_origin)
                if not query.exec_():
                    self.textEdit_code.append("{Occurrences}: [ERROR] ### Erreur lors de l'insertion
des id_occurrence_occtax dans "+table+" ###")
                    self.textEdit_code.append(query.lastError().text())
                    self.ifErrorOnInsertIntoGeoNature += 1
                    self.suppressionApresEchec()

        if self.ifErrorOnInsertIntoGeoNature == -1:
            # ================================================/ Insertion des occurences (cas où le
nom_taxon dépasse 255 caractères) /================================================
            self.textEdit_code.append("{Occurrences}: Insertion des occurrences de relevé (nom_taxon +
de 255 caratères)")
            self.progressBar_code.setValue(70)

            query = QSqlQuery(self.db)
            sql = "INSERT INTO pr_occtax.t_occurrences_occtax (id_releve_occtax,
id_nomenclature_obs_meth, id_nomenclature_bio_condition, id_nomenclature_bio_status,
```

```
id_nomenclature_naturalness, id_nomenclature_exist_proof, "
            sql += "id_nomenclature_observation_status, id_nomenclature_blurring,
id_nomenclature_source_status, determiner, id_nomenclature_determination_method, cd_nom, nom_cite,
meta_v_taxref, digital_proof, non_digital_proof, comment) "
            sql += "SELECT " + table + ".id_releve_occtax, " + methode_observation + ", " +
etat_biologique + ", " + statut_biologique + ", " + niveau_de_naturalite + ", " + preuve_existance + ",
" + statut_observation + ", "
            sql += floutage_donnee + ", " + statut_source + ", " + determinateur + ", " +
methode_determination + ", "
            sql += table + "." + cd_nom_taxon + ", CONCAT(taxref.lb_nom,' = <i> ', taxref.nom_complet,
'</i> - [', taxref.id_rang, ' - ', taxref.cd_nom ,']') AS nom_cite, '" + taxref_version + "', " +
detail_preuve_digital + ", " + detail_preuve_non_digital + ", " + commentaire_occurrences + " "
            sql += "FROM " + table_origin + " "
            sql += "LEFT JOIN pr_occtax.t_releves_occtax ON " + table + ".id_releve =
t_releves_occtax.id_releve_occtax "
            sql += "LEFT JOIN taxonomie.taxref ON " + table + "." + cd_nom_taxon + " = taxref.cd_nom "
            sql += "WHERE " + table + ".id_occurrence_occtax IS NULL "
            sql += "ORDER BY " + table + "." + id_unique + " RETURNING id_occurrence_occtax;"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Occurrences}: [ERROR] ### Erreur lors de l'insertion des
occurrences de relevé ###")
                self.textEdit_code.append(sql)
                self.textEdit_code.append(query.lastError().text())
                self.ifErrorOnInsertIntoGeoNature += 1



            query2 = QSqlQuery(self.db)  # On selectionne juste les id en respectant les conditions
précédentes et le ORDER BY
            sql2 = "SELECT " + id_unique + " FROM " + table_origin + " "
            sql2 += "WHERE " + table + ".id_occurrence_occtax IS NULL "
            sql2 += "ORDER BY " + id_unique + ";"
            query2.prepare(sql2)
            if not query2.exec_():
                self.textEdit_code.append("{Occurrences}: [ERROR] ### Erreur lors de la sélection de
l'id unique de "+table+" ###")
                self.textEdit_code.append(query2.lastError().text())
                self.ifErrorOnInsertIntoGeoNature += 1

            self.textEdit_code.append("{Occurrences}: Définition de la correspondance " + table + " /
t_occurences_occtax")
            self.progressBar_code.setValue(80)



            liste2 = []  # Définition de la correspondance id_unique / id_occurrence_occtax et
insertion dans une liste de tuples
            while query.next() and query2.next():
                queryIdOccurrence = query.value(0)
                queryIdUnique = query2.value(0)
                tuple = (queryIdUnique, queryIdOccurrence)
                liste2.append(tuple)

            # Pour tous les éléments de la liste,  nous allons UPDATE la colonne id_occurence_occtax de
la table_origine et faire ainsi la correspondance

            for elt2 in liste2:
                id_origin = elt2[0]
                id_occurrence_occtax = elt2[1]

                query = QSqlQuery(self.db)
                sql = "UPDATE " + table_origin + " SET id_occurrence_occtax = ? "
                sql += "WHERE " + id_unique + " = ?;"
                query.prepare(sql)
                query.addBindValue(id_occurrence_occtax)
```

```
                query.addBindValue(id_origin)
                if not query.exec_():
                    self.textEdit_code.append("{Occurrences}: [ERROR] ### Erreur lors de l'insertion
des id_occurrence_occtax dans "+table+" ###")
                    self.textEdit_code.append(query.lastError().text())
                    self.ifErrorOnInsertIntoGeoNature += 1
                    self.suppressionApresEchec()

        if self.ifErrorOnInsertIntoGeoNature == -1:
            # ================================================/ Insertion dans cor_couting_occtax
/================================================
            self.textEdit_code.append("{Counting}: Insertion du dénombrement des occurrences de
relevé")
            self.progressBar_code.setValue(90)

            query = QSqlQuery(self.db)
            sql = "INSERT INTO pr_occtax.cor_counting_occtax (id_occurrence_occtax,
id_nomenclature_life_stage, id_nomenclature_sex, id_nomenclature_obj_count, id_nomenclature_type_count,
count_min, count_max) "
            sql += "SELECT " + table + ".id_occurrence_occtax, " + stade_vie + ", " + sexe + ", " +
objet_denombrement + ", " + type_denombrement + ", " + nb_min + ", " + nb_max + " "
            sql += "FROM pr_occtax.t_occurrences_occtax "
            sql += "RIGHT JOIN " + table_origin + " ON " + table + ".id_occurrence_occtax =
t_occurrences_occtax.id_occurrence_occtax;"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Counting}: [ERROR] ### Erreur lors de l'insertion dans
cor_counting_occtax ###")
                self.textEdit_code.append(query.lastError().text())
                self.ifErrorOnInsertIntoGeoNature += 1
                self.suppressionApresEchec()



        if self.ifErrorOnInsertIntoGeoNature == -1:
            # ================================================/ Validation
/================================================
            self.textEdit_code.append("{Validation}: Application de la validation pour les occurrences
concernées")
            self.progressBar_code.setValue(95)

            query = QSqlQuery(self.db)
            sql = "INSERT INTO gn_commons.t_validations (uuid_attached_row,
id_nomenclature_valid_status, id_validator, validation_auto, validation_comment, validation_date) "
            sql += "SELECT unique_id_sinp_occtax, " + statut_validation + ", " + id_validateur_origin +
", false, "+ commentaire_validation +", NOW() "
            sql += "FROM pr_occtax.cor_counting_occtax "
            sql += "INNER JOIN " + table_origin + " ON "+ table +".id_occurrence_occtax =
cor_counting_occtax.id_occurrence_occtax;"
            query.prepare(sql)
            if not query.exec_():
                self.textEdit_code.append("{Validation}: [ERROR] ### Erreur lors de l'insertion des
occurrences validées depuis " + table + " ###")
                self.textEdit_code.append(query.lastError().text())
                self.ifErrorOnInsertIntoGeoNature += 1
                self.suppressionApresEchec()


        if self.ifErrorOnInsertIntoGeoNature == -1:
            self.progressBar_code.setValue(100)
            self.textEdit_code.append("{Historique}: Historisation de l'import")
            self.insertHistoriqueTable()
            self.titre.setText("Import terminé")
            self.textEdit_code.append("======================== INSERTION TERMINÉE
========================")
            self.textEdit_code.setStyleSheet("background-color: white; color: green;")
```

```
            msg = QMessageBox()
            msg.setIcon(QMessageBox.Information)
            msg.setText("Les données ont été intégrés avec succès.")
            msg.setWindowTitle("Tâche complète")
            msg.exec_()

            self.pb_terminer.setEnabled(True)

    def insertHistoriqueTable(self):
        query = QSqlQuery(self.db)
        wrelation = self.dbSchema + "." + "import_historique"
        sql = "INSERT INTO "+ wrelation +" (imphisto_nom_table, imphisto_nom_module, imphisto_nom_jdd,
imphisto_date_import, imphisto_suppr_histo, imphisto_suppr_histo_with_module_data) "
        sql += "VALUES ('%s', '%s', '%s', NOW(), false, false);" % (self.nomtable, self.modulenom,
self.jddnom)
        query.prepare(sql)
        if not query.exec_():
            msg = QMessageBox()
            msg.setIcon(QMessageBox.Critical)
            msg.setText("Erreur - création import_historique")
            msg.setInformativeText(query.lastError().text())
            msg.setWindowTitle("Erreur")
            msg.exec_()

    def close(self):
        QDialog.close(self)
```

ANNEXE XXIII : GUIDE UTILISATEUR DU MODULE MONITORING

Utilisation du module Monitoring de GeoNature pour la saisie du Comptage Mensuel Commun

Version du 10 août 2020

Maxime TOMA - 2020

Présentation de l'architecture du module

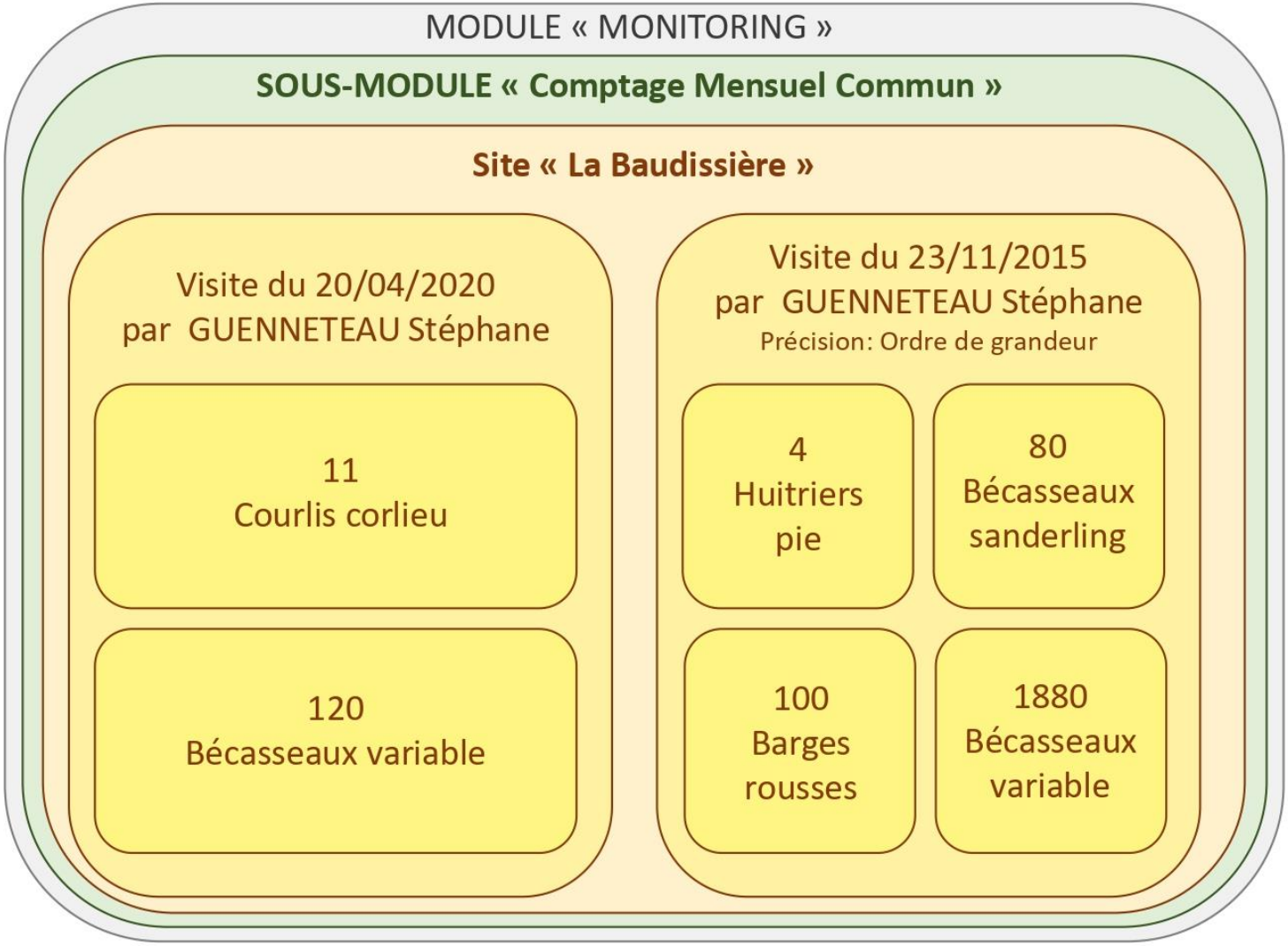
Le module « Monitoring » (ou Module Générique de Suivis) présente un fonctionnement un peu différent que ses congénères.

En effet ce module abrite des **sous-modules** représentant chacun un **protocole de suivis** (ou jeu de données).

Un sous-module se compose de trois parties imbriquées:

- Sites
- Visites du site
- Observations pour chaque visite

Voici un exemple ci-contre avec le sous-module « Comptage Mensuel Commun ».



Présentation de l'interface

Chaque boite correspond à un protocole de suivis (ou Jeu de données).
Cliquez sur la boite correspondant au protocole de « Comptage Mensuel Commun » de votre RN.

Les sites

Chaque protocole a ses propres attributs à chaque niveau (site, visite, observation).

Les sites sont définis qu'une seule fois. Ensuite, chaque site accueille un certains nombre de visites.

Pour ce protocole, un site a comme attributs:
- Nom du site
- Code du site
- Date de description (date à laquelle on enregistre le polygone)

Champs calculés automatiquement (non-modifiables):
- Date de la dernière visite
- Nombre de visites

<u>NB:</u> ***Seul les sites peuvent accueillir une géométrie à l'heure actuelle****. La possibilité de le mettre pour les visites et les observations est en cours de réalisation par l'équipe de développement de GeoNature.*

4/11

Les visites

Les visites contiennent une ou plusieurs observations.

Pour ce protocole, une visite a comme attributs:
- Date de la visite
- Observateur(s)
- Précision (champ personnalisé)
- Commentaire

Champs calculés automatiquement (non-modifiables):
- Nombre d'observations

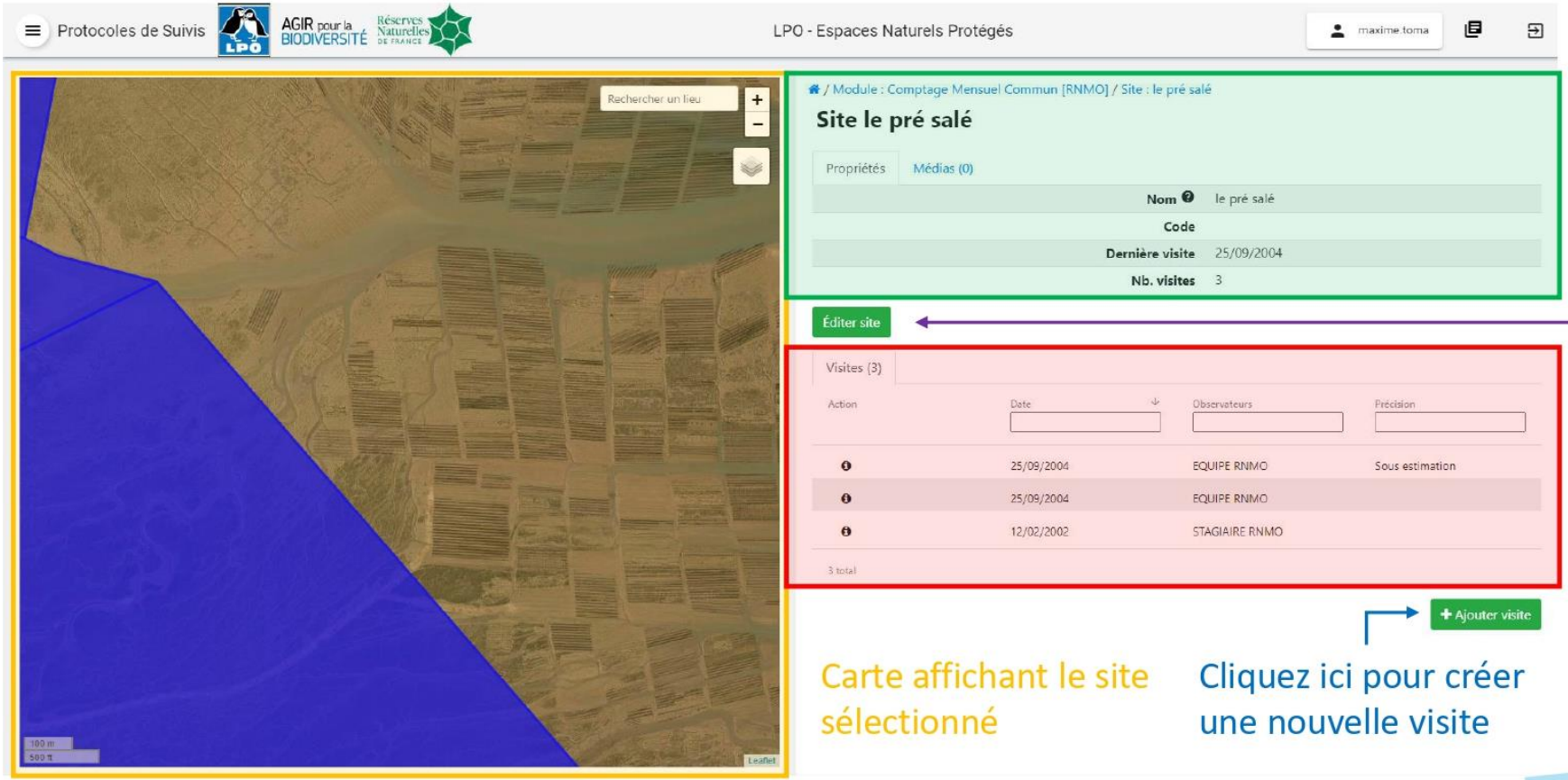
Les observations

Pour ce protocole, une observation a comme attributs:

- Taxon
- Dénombrement (champ personnalisé)
- Commentaire

Ajouter une visite

⚠ Pour l'ajout/ou modification de la géométrie d'un site, s'adresser aux administrateurs.



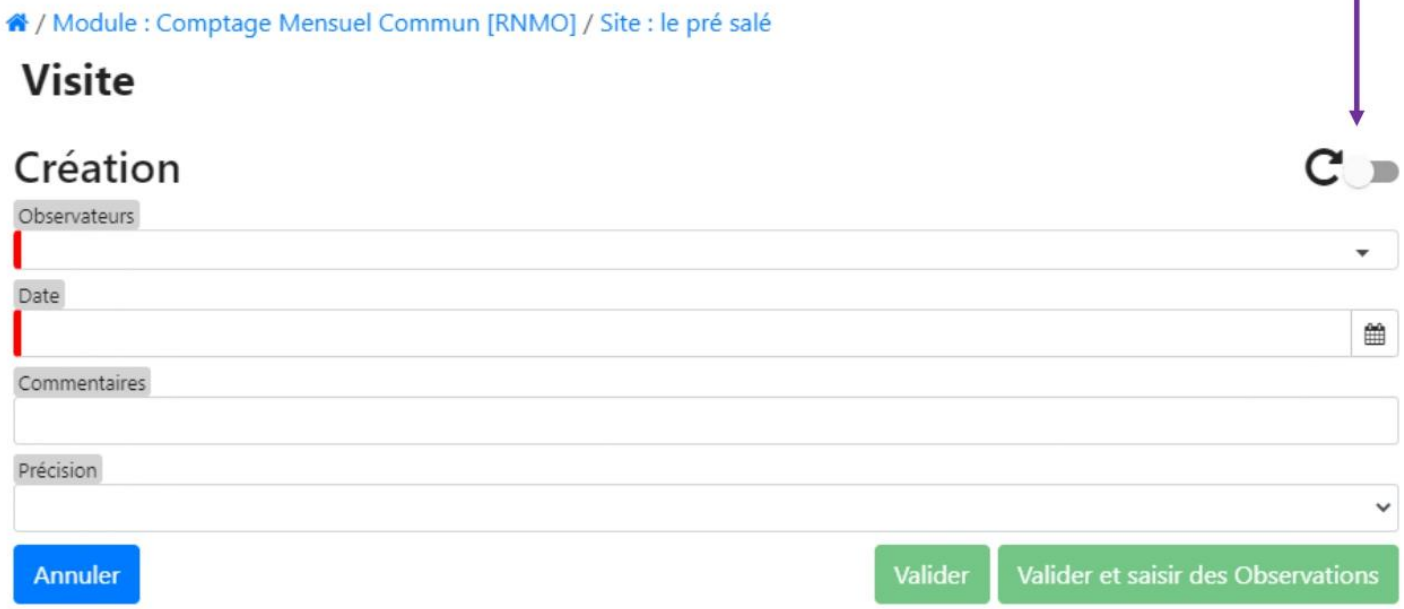
Informations (et médias) liées au site.

Cliquez ici pour modifier les informations liées au site.

Informations liées aux visites (avec possibilité de filtrer/trier les visites). Le « i » permet d'afficher le détail de la visite.

7/11

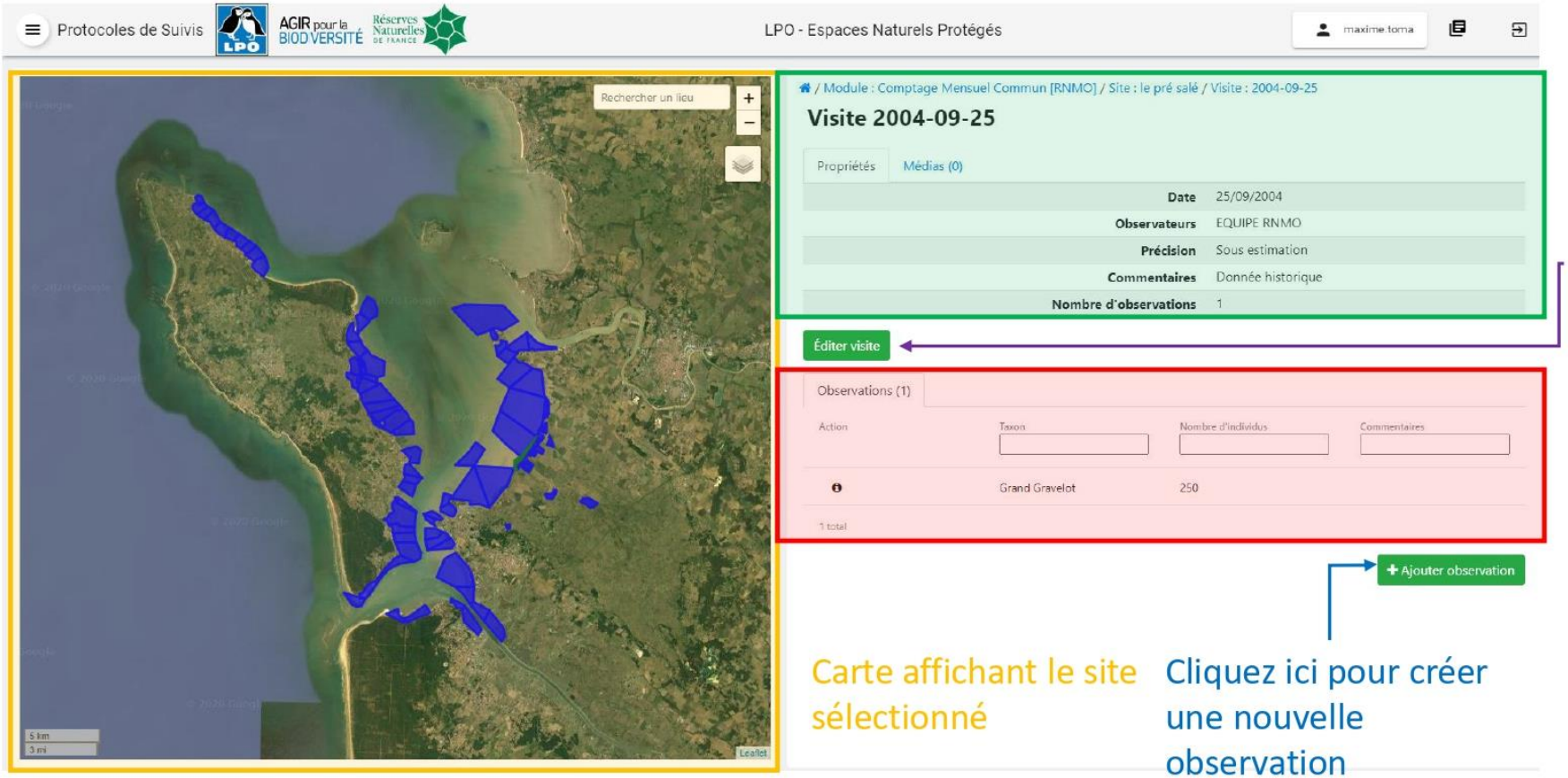
Ajouter une visite



Si activé, permet l'enchainement de création de visite pour ce site

1. Saisissez **un ou plusieurs observateurs** (les champs rouges sont obligatoires)
2. Saisissez **la date**
3. Ecrivez (ou non) **un commentaire**
4. Donnez (ou non) **une précision** pour les observations de la visite
5. Validez ou Validez en saisissant directement une observation dans cette visite

Ajouter une observation



Informations (et médias) liées à la visite.

Cliquez ici pour modifier les informations liées à la visite.

Informations liées aux observations (avec possibilité de filtrer/trier les observations). Le « i » permet d'afficher le détail de l'observation.

9/11

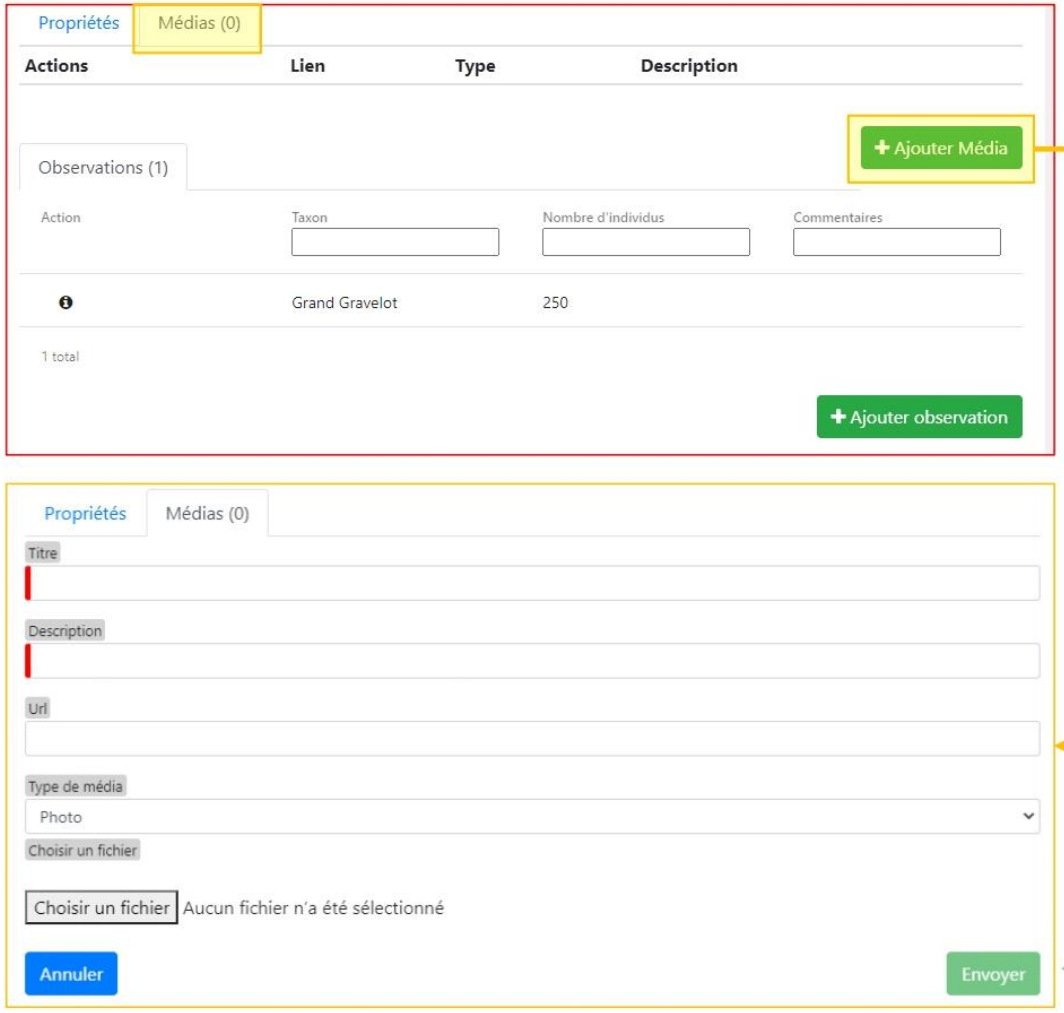
Ajouter une observation

Si activé, permet l'enchainement de création d'observations pour cette visite

1. Saisissez **un taxon** (les champs rouges sont obligatoires)
2. Ecrivez (ou non) **un commentaire**
3. Indiquez **le nombre d'individus**
4. Validez l'observation

10/11

Ajouter un média



Un (ou plusieurs) média(s) peut être affilié à un site, une visite ou une observation et sera enregistré dans la base de données.

1. Cliquez sur l'onglet « Médias »
2. Cliquez sur « Ajouter Média »
3. Indiquez **les différentes informations** ainsi que choisissez le média à importer
4. Validez

11/11

ANNEXE XXIV : GUIDE UTILISATEUR DU PLUGIN « GEOPIM »

GEO Nature

GEOPIM

Guide d'utilisation

Version 1.1.0

Version du 27 août 2020

LPO

AGIR pour la BIODIVERSITÉ

SOMMAIRE

DECOUVERTE DES ELEMENTS DU PLUGIN 3

LA GESTION DES PARAMETRES 4

L'INSERTION DES DONNEES DANS GEONATURE 6

FENETRE PRINCIPALE 6

FENETRE DE L'IMPORT 10

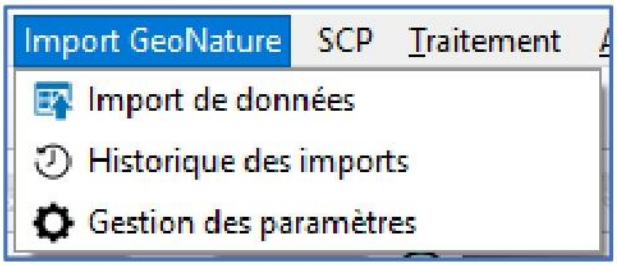
L'HISTORIQUE DES IMPORTS 12

DECOUVERTE DES ELEMENTS DU PLUGIN

Une fois le plugin installé, il se compose de la façon suivante :

- **Menu déroulant :**

Le menu déroulant est composé de plusieurs boutons d'action (cliquables). Ces boutons d'action ouvrent les différents formulaires qui leurs sont associés.



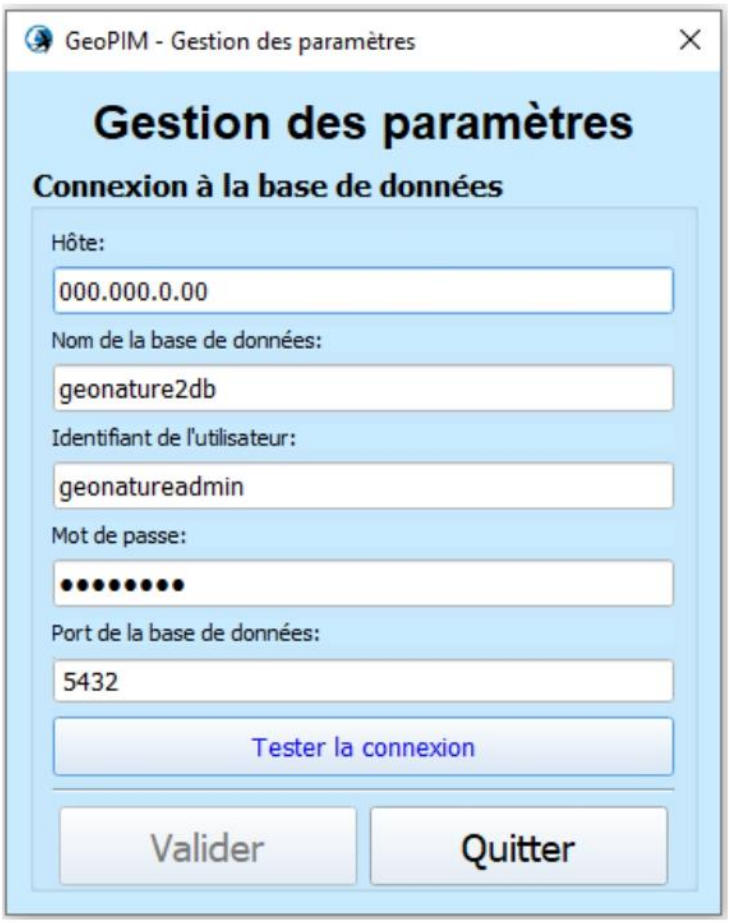
Il existe 3 actions différentes : **Import de données**, **Historique des imports** et **Gestion des Paramètres**. Elles seront détaillées tout à l'heure.

- **Toolbar :**

La toolbar, située en permanence sur la fenêtre principale de QGIS, sont des raccourcis permettant l'ouverture des formulaires principaux. Seuls les actions **Import de données** et **Historique des imports** sont présentes sur la toolbar.

LA GESTION DES PARAMETRES

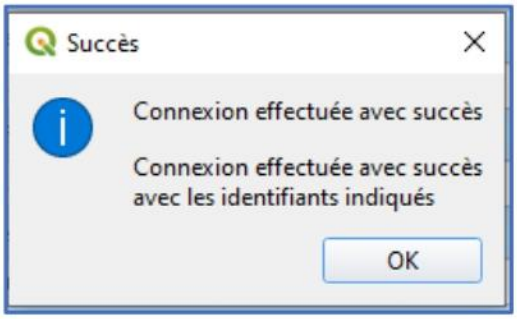
Il existe donc une action nommée **Gestion des paramètres**. Elle ouvre une fenêtre demandant de renseigner ses identifiants de connexion à votre base de données GeoNature.

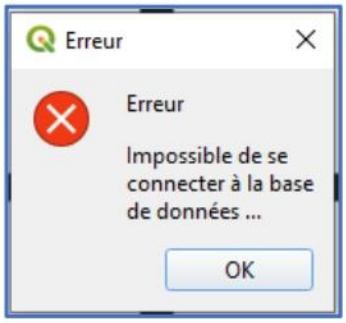


Elle garde également en mémoire, et en clair, les identifiants renseignés dans le fichier *param_bdd.py*.

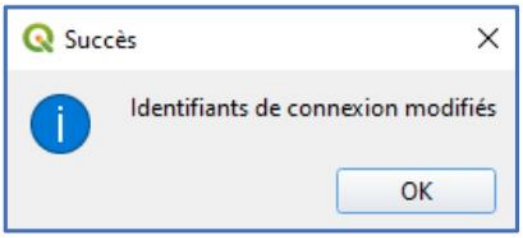
Ainsi, vous pouvez changer vos identifiants si vous avez une autre base GeoNature, par exemple, ou tout simplement les renseigner pour votre première utilisation.

1. En premier, vous allez modifier vos identifiants, puis vous cliquerez sur « Tester la connexion ». Une petite fenêtre s'affichera et vous indiquera si les identifiants sont bons, ou non, et que la connexion a pu être faite, ou non, avec la base.

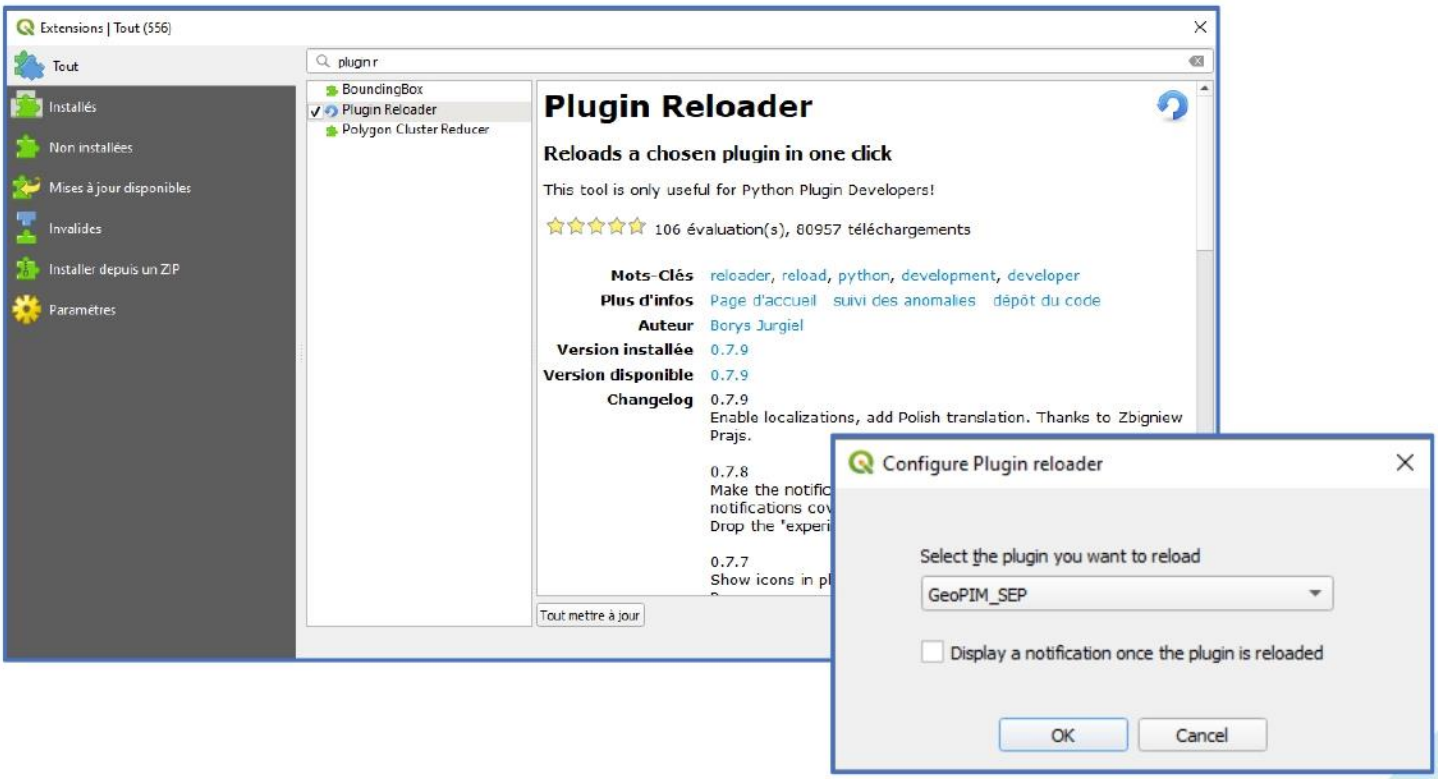




2. Le bouton « Valider » devient alors cliquable. Si vous n'avez rien modifié, le message de validation de la modification des identifiants s'affiche. Dans le cas contraire, une revérification de la connexion se fera.



3. **ATTENTION** : La modification n'est effective que si vous rafraichissez le plugin. Pour ce faire, soit vous fermez QGIS et vous le relancez, soit vous installez le Plugin Reloader (https://plugins.qgis.org/plugins/plugin_reloader/).



L'INSERTION DES DONNEES DANS GEONATURE

L'import des données dans GeoNature est géré en deux fenêtres :

Fenêtre principale

Tout d'abord, **la fenêtre principale**. Elle se compose de 4 parties :

- **<u>Le choix du module</u>** :

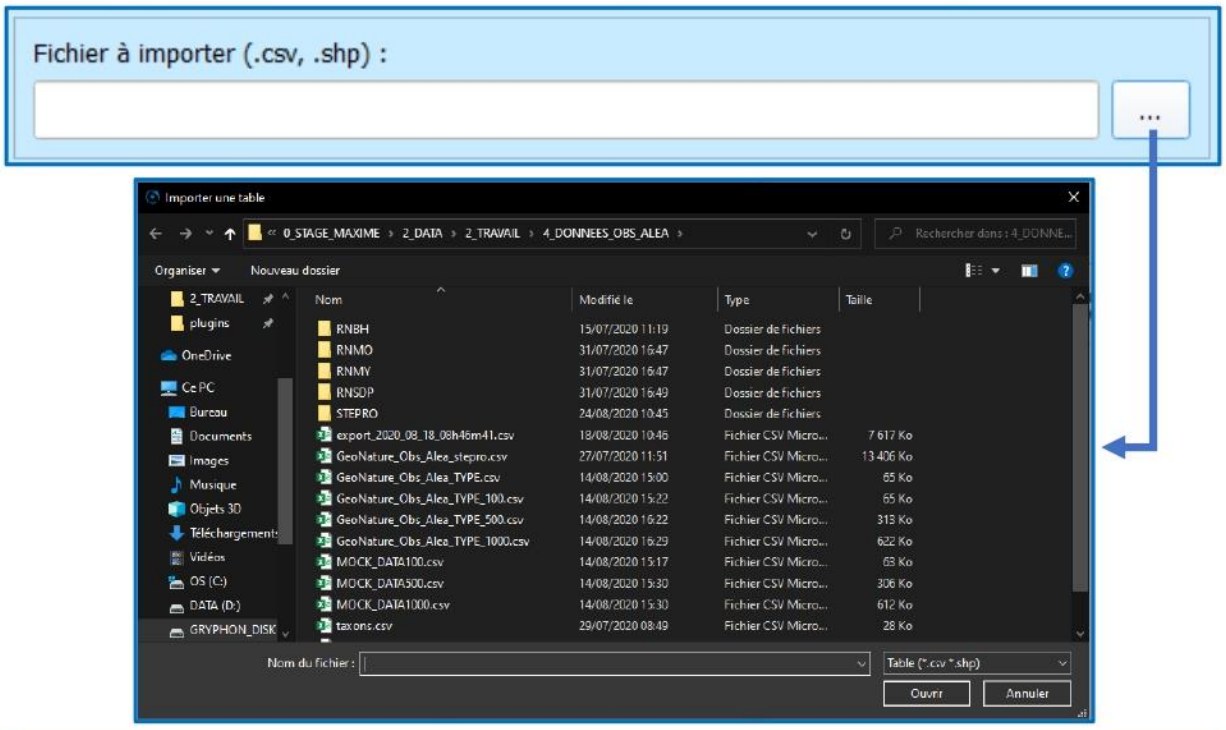
En premier, nous choisissons le module de GeoNature dans lequel on souhaite insérer nos données.

Actuellement, seuls les imports vers **OccTax** et le sous-module **Comptage Mensuel Commun** du module Monitoring sont disponibles.

Après avoir sélectionné le module, la partie du choix du fichier s'active.

- **<u>Le choix du fichier</u>** :

Nous choisissons ensuite le fichier (en .csv ou en .shp) que l'on veut importer. Actuellement, les modules supportés n'acceptent uniquement que les imports au format .csv.

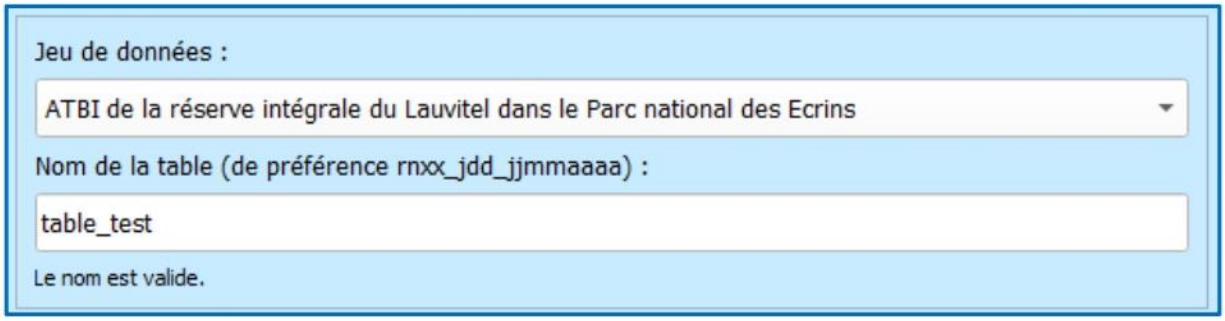


Attention, l'import se fait uniquement si l'ordre des colonnes du fichier type est respecté. Ce fichier type est mis à disposition des utilisateurs sur le serveur de la LPO (T:\Sep\Espaces Protégés\ETUDES SEP PAT NAT\2020 GEONATURE\FICHIERS_TYPE).

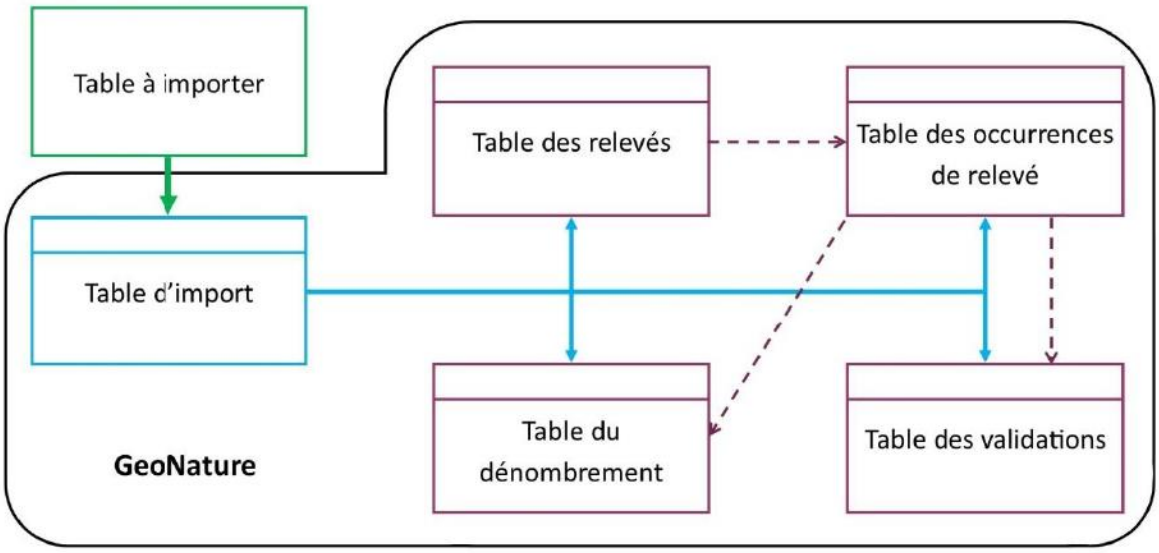
Après avoir sélectionné le fichier, la partie du choix du JDD (Jeu de données) et du nom de la table d'import s'active.

- **Le choix du JDD et du nom de la table** :

Nous pouvons, après avoir choisi le module et le fichier à importer, le jeu de données associé à ces données. Si un jeu de données n'est pas présent dans la liste c'est que soit il n'est pas lié au module, soit il n'a pas encore été créé. Veuillez alors vous tourner vers un administrateur de votre GeoNature.



Enfin, nous pouvons choisir un nom pour la table d'import. La table d'import est une table provisoire que va accueillir la base de GeoNature afin d'intégrer ensuite les données dans les différentes tables que compose le module.

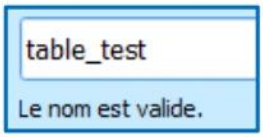


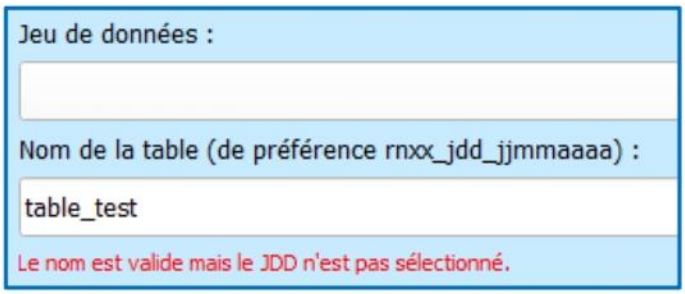
En vert : le CSV à importer.

En bleu : le CSV importé désormais sous la forme d'une table PostgreSQL.

En violet : les tables composantes du module OccTax.

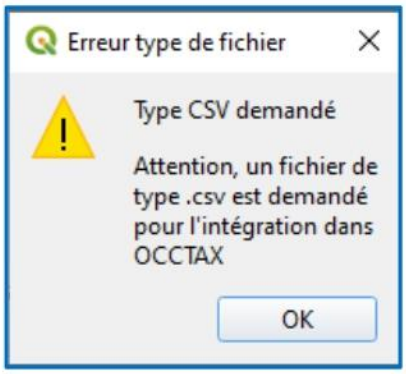
Le nom doit au moins contenir trois lettres, aucun espace ou caractère spécial, et ne doit pas être utilisé par une autre table. Veillez à ce que le nom soit assez significatif afin de facilement le retrouver dans l'**Historique des imports**. Ainsi, une vérification est faite en temps réel et vous indique en dessous si votre nom de table est validé. Dans ce cas, le bouton « Valider » s'active.





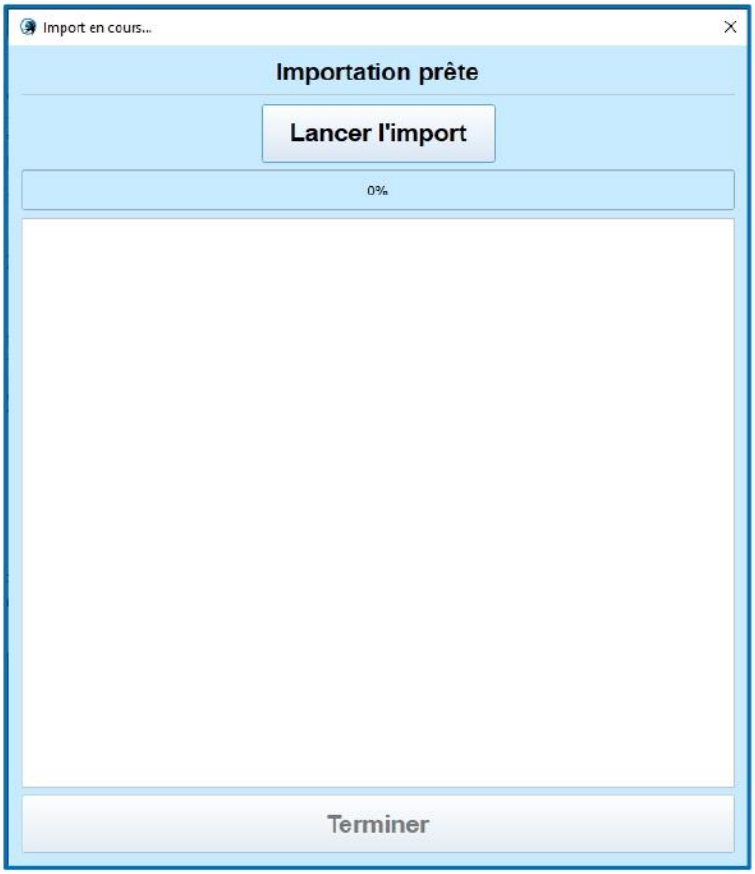
- **<u>La validation</u>** :

Ainsi, quand tout est bon, le bouton « Valider » s'active. On peut alors cliquer dessus pour passer à l'import des données. Cependant, avant d'afficher cette fenêtre, on vérifie si le format de fichier est bon pour le module choisi.



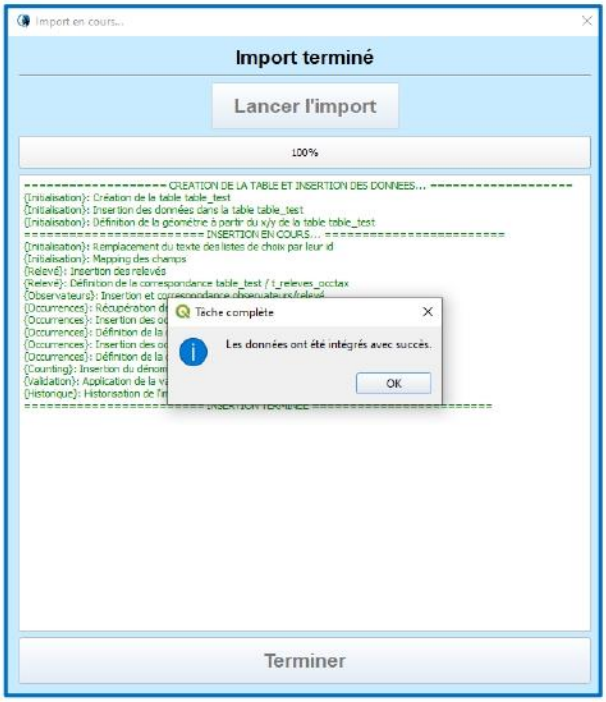
Fenêtre de l'import

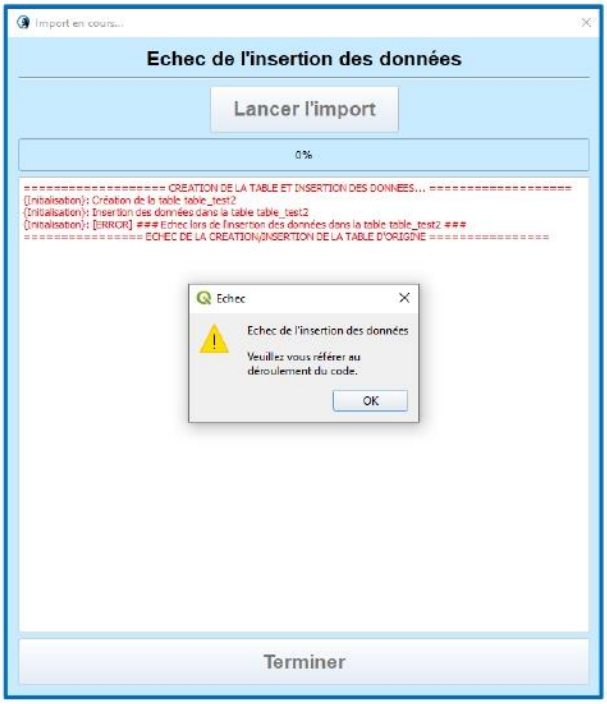
Nous arrivons enfin sur une nouvelle fenêtre. Cette dernière va prendre tous les éléments précédemment indiqués et va lancer un script d'intégration dans le module en question. Ainsi, on cliquera sur « Lancer l'import » pour commencer à intégrer les données.



Une barre indique la progression de l'opération tandis que des lignes apparaissent décrivant les différentes étapes de cette opération. Il se peut que la fenêtre ne réponde pas pendant un petit moment, c'est normal. L'intégration continue toujours.

Le log (rapport) des étapes de l'intégration est alors indiqué en vert, accompagné d'une petite fenêtre, indiquant le succès de l'opération. Dans le cas contraire, le texte sera affiché en rouge, les données ne seront pas intégrés et la table d'import aura été supprimée.





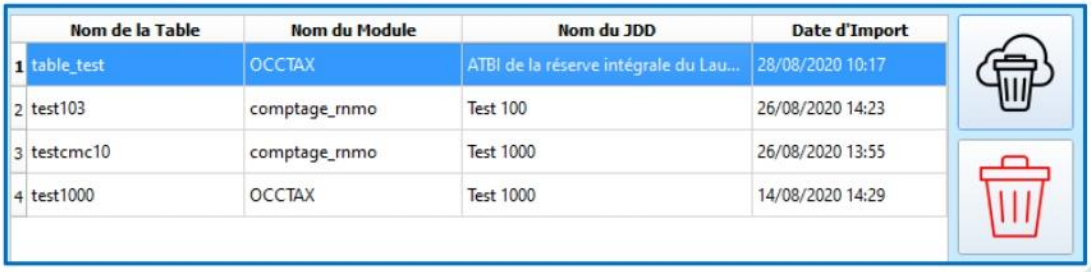
L'HISTORIQUE DES IMPORTS

Une dernière fenêtre décrit l'historique des imports effectués via le plugin sur la base GeoNature.

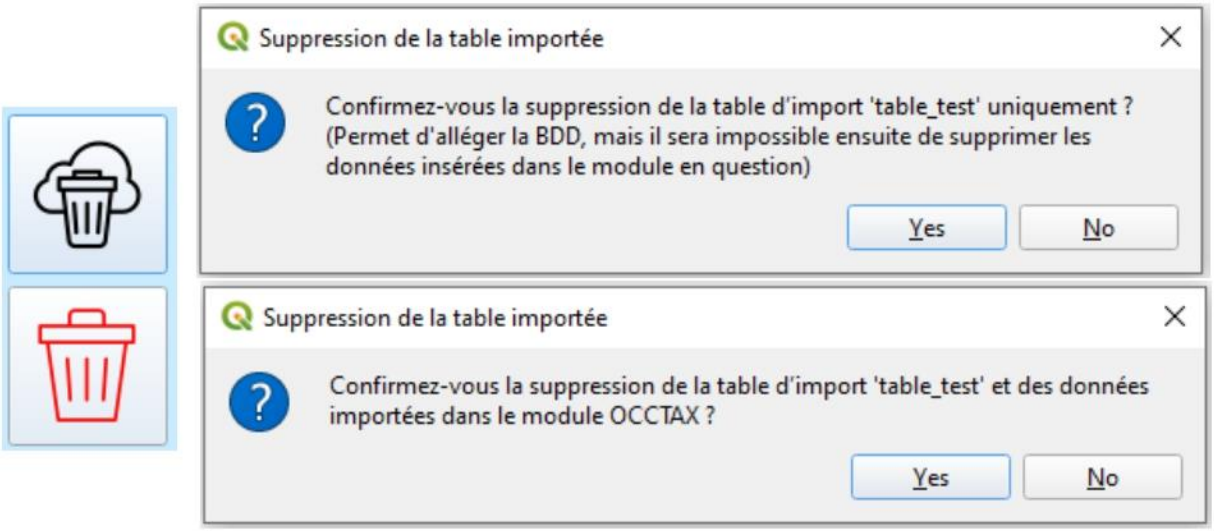
Il est alors indiqué le nom de la table d'import choisi par l'utilisateur (ex : *table_test* de tout à l'heure), le nom du module dans lequel les données ont été intégrés, le nom du JDD auquel ils ont été associés et enfin la date et heure de l'import.

Cette fenêtre fait office de suivi des imports mais on peut également les supprimer. Une fois une ligne sélectionnée, on a deux boutons :

- Le premier (**noir**) supprimera la table d'import uniquement. Les données seront conservé dans le module de destination mais ne pourront plus être supprimés de façon massive (en fonction de la table d'import donc).
- Le second (**rouge**), lui, supprimera à la fois la table d'import ainsi que les données associées dans le module de GeoNature.



On demande alors confirmation à l'utilisateur pour la suppression des objets.



Tant que la fenêtre n'a pas été fermée, la ligne supprimée restera vide.

	Nom de la Table	Nom du Module	Nom du JDD	Date d'Import
!				
2	test103	comptage_rnmo	Test 100	26/08/2020 14:23
3	testcmc10	comptage_rnmo	Test 1000	26/08/2020 13:55
4	test1000	OCCTAX	Test 1000	14/08/2020 14:29

ANNEXE XXV : FICHES RECAPITULATIVES DE TRAVAIL

Fiche récapitulative de travail sur les modules – GeoNature

Module concerné : **Interface GeoNature / Validation / Synthèse / Dashboard**

Période de travail : 25/05/2020 – 12/08/2020

Historique des actions réalisées :

Mai	
25/05/2020	• Documentation, acquisition des connaissances
26/05/2020	• Documentation, acquisition des connaissances • Manipulation de GeoNature / UsersHub / TaxHub • Création de Jeux de Données (JDD) et de Cadres d'Acquisition (CA) • Navigation dans la base de données : découverte des tables et du MCD
27/05/2020	• Documentation, acquisition des connaissances • Manipulation des droits (CRUVED)
29/05/2020	• Import de niveau 1 dans la Synthèse
Juin	
03/06/2020	• Suppression des données entrées dans la Synthèse • Insertion et suppression de 100 nouvelles données test ayant le même X/Y ou le même polygone
04/06/2020	• Intégration des limites des RNs dans le Référentiel Géographique (ref_geo)
05/06/2020	• Insertion des taxons issus de SERENA via TaxHub
11/06/2020	• Insertion des utilisateurs issus de SERENA via UsersHub (début)
12/06/2020	• Insertion des utilisateurs issus de SERENA via UsersHub (suite et fin)
16/06/2020	• Création user et organisme LIENSs et ajout avec Jean TERRISSE
18/06/2020	• Travail Tableau des permissions • Ajout Éric BRUGEL dans UsersHub
19/06/2020	• Insertion de référentiels géographiques (PN, PNR, Natura 2000 etc…)
23/06/2020	• Premiers essais sur la personnalisation graphique de l'interface de GeoNature (Logos, image accueil, bannière d'info, nom) • Installation et configuration du module Dashboard • Modification du canevas de la carte
24/06/2020	• Modification de la vue d'export SINP de la Synthèse (observateurs mis automatiquement)
25/06/2020	• Ajout des logos des réserves dans la bannière d'info + modification CSS • Test d'installation du module Exports (échec)
26/06/2020	• Modification des icones des modules • Test d'installation du module Monitoring (échec – incompatibilité avec la version actuellement installée)
30/06/2020	• Activation de l'accès à l'interface « Mon compte » dans GeoNature • Configuration des filtres par réserve dans Dashboard et Synthèse

Fiche récapitulative de travail sur les modules – GeoNature

Juillet	
02/07/2020	• Intégration d'espèces invertébrés dans TaxHub
06/07/2020	• Création de 3 niveaux de sites spécifiques dans le Référentiel Géographique (Lieu-dit, Unité de Gestion, Site d'Inventaire) • Intégration des sites spécifiques de la RNBH • Création LPO, LPO85, ONCFS, Ecogardes RNMO/RNBH, Stagiaires RNMO/RNBH Equipe RNMO/RNBH etc… dans UsersHub
15/07/2020	• Intégration des sites spécifiques de la RNMO
23/07/2020	• Création Ecogardes, Stagiaires, Equipe des réserves restantes dans UsersHub • Intégration des sites spécifiques de la RNSDP et de la STEPRO
28/07/2020	• Installation sous machine virtuelle de la dernière version de GeoNature
31/07/2020	• Découverte et essais d'installation de divers modules dont Atlas, Citizen, Import, Export et Mobile
Août	
10/08/2020	• Rédaction du guide utilisateur du module Synthèse, Validation
11/08/2020	• Rédaction du guide administrateur (début)
12/08/2020	• Rédaction du guide administrateur (fin)

Fiche récapitulative de travail sur les modules – GeoNature

Module concerné : **OccTax**

Période de travail : 03/06/2020 – 14/08/2020

Historique des actions réalisées :

	Juin	
03/06/2020	•	Commencement Tableau de correspondance des champs
04/06/2020	•	Rédaction d'un guide rapide d'utilisation de OccTax pour le personnel des réserves
19/06/2020	•	Continuation du Tableau de correspondance des champs
	Juillet	
01/07/2020	•	Création script python d'intégration dans OccTax
02/07/2020	•	Travail sur le script python d'intégration avec donnée test
03/07/2020	•	Travail sur le script python d'intégration avec donnée test
06/07/2020	•	Travail sur le script python d'intégration avec donnée test (fin)
07/07/2020	•	Intégration des données de la RNBH
08/07/2020	•	Modification d'une partie du script d'intégration
17/07/2020	•	Intégration des données de la RNMO (début)
20/07/2020	•	Intégration des données de la RNMO (fin)
21/07/2020	•	Création du CSV type pour les observations aléatoires
27/07/2020	•	Intégration des données de la STEPRO (crash de la base – code validation à revoir)
	Août	
14/08/2020	•	Modification du code python d'intégration (STEPRO)

Fiche récapitulative de travail sur les modules – GeoNature

Module concerné : **OccHab**

Période de travail : 15/06/2020 – 02/07/2020

Historique des actions réalisées :

Juin	
15/06/2020	• Recherches sur les fonctions SQL • Tableau de correspondance Lilleau des Niges (SHP/GEONATURE)
16/06/2020	• Rédaction du script d'intégration Lilleau des Niges
17/06/2020	• Travail sur la donnée de Moëze-Oléron
18/06/2020	• Travail sur la donnée de Lilleau des Niges • Modification script SQL (ajout de la condition cd_typo = 7) • Test d'intégration de la donnée Lilleau des Niges dans la BD test - OK
22/06/2020	• Création script SQL pour les données de la RNSDP
23/06/2020	• Modification du script SQL pour les données de la RNLDN
24/06/2020	• Intégration des données Hermelles 2020 de la RNMO
26/06/2020	• Intégration des données Hermelles 2013-2015 de la RNMO et Zostères 2014 de la RNMY
29/06/2020	• Travail sur la donnée intégrée (RNMO/RNSDP/RNMY)
30/06/2020	• Intégration des données habitats de la RNMO • Travail sur le script python
Juillet	
01/07/2020	• Travail sur le script python • Intégration de la carto habitat 2011 et 2019 de la RNMO
02/07/2020	• Intégration du reste de la carto de la RNMO 2011 (format TAB) – Reprojection car de base LAMBEtendu

Fiche récapitulative de travail sur les modules – GeoNature

Module concerné : **Monitoring (Module Générique de Suivis)**

Période de travail : 29/07/2020 – 08/09/2020

Historique des actions réalisées :

Juillet	
29/07/2020	• Installation et configuration générale du module • Création et configuration du sous-module Comptage Mensuel Commun RNMO (CMC RNMO) • Intégration des sites dans le sous-module CMC RNMO
30/07/2020	• Fin d'intégration des sites dans le sous-module CMC RNMO • Intégration des visites et des observations
Août	
10/08/2020	• Rédaction du guide utilisateur du module
24/08/2020	• Rédaction du CSV type • Création du fichier synthese.sql • Actualisation de la synthèse pour l'import massif de données • Refonte du fichier python d'intégration
Septembre	
07/09/2020	• Création des sous-modules « Comptage Mensuel Commun » pour toutes les RN restantes
08/09/2020	• Intégration des sites de la STEPRO de la RNSDP

Fiche récapitulative de travail sur les modules – GeoNature

Module concerné : **Exports**

Période de travail : 31/07/2020 – 11/09/2020

Historique des actions réalisées :

Juillet	
31/07/2020	• Réussite d'installation (config MAIL ok pour maxime.toma@lpo.fr)
Août	
10/08/2020	• Rédaction du guide utilisateur du module • Création d'un export pour le Comptage Mensuel Commun de la RNMO
Septembre	
11/09/2020	• Modification du mail pour celui de Frédéric ROBIN

Fiche récapitulative de travail sur les modules – GeoNature

Module concerné : Aucun **(Plugin)**

Période de travail : 08/07/2020 – 28/08/2020

Historique des actions réalisées :

Juillet	
08/07/2020	• Création des IHM • Création des premiers fichiers du plugin • Validation de l'IHM
09/07/2020	• Création des fichiers .ui • Début de création des fichiers de dialog
13/07/2020	• Rédaction des fichiers de dialog (principal + import_dialog)
15/07/2020	• Correctif symbologie plugin • Rédaction des fichiers de dialog (import_occtax_dialog) + gestion des msg textEdit
16/07/2020	• Rédaction des fichiers de dialog (import)
17/07/2020	• Travail sur le plugin : Intégration de données test
20/07/2020	• Rédaction des fichiers de dialog (historique, import, import occtax)
21/07/2020	• Rédaction des fichiers de dialog (historique, import, import occtax) • Création vues/tables pour le fonctionnement du plugin
22/07/2020	• Rédaction des fichiers de dialog (historique, import, import occtax)
24/07/2020	• Rédaction des fichiers de dialog (historique, import, import occtax)
27/07/2020	• Rédaction des fichiers de dialog (historique, import, import occtax)
28/07/2020	• Fin de la rédaction des fichiers de dialog pour OccTax
Août	
14/08/2020	• Modification du code import_occtax • Test d'import avec 500, 1000 et 5000 données
25/08/2020	• Création des IHM partie 2 • Validation de l'IHM partie 2 • Création des fichiers .ui partie 2
26/08/2020	• Développement du code pour l'import dans Monitoring + Connexion à la base de données
27/08/2020	• Gestion d'une erreur sur VM (affichage de l'heure) • Rédaction du guide utilisateur
28/08/2020	• Rédaction du guide utilisateur • Rédaction du guide administrateur