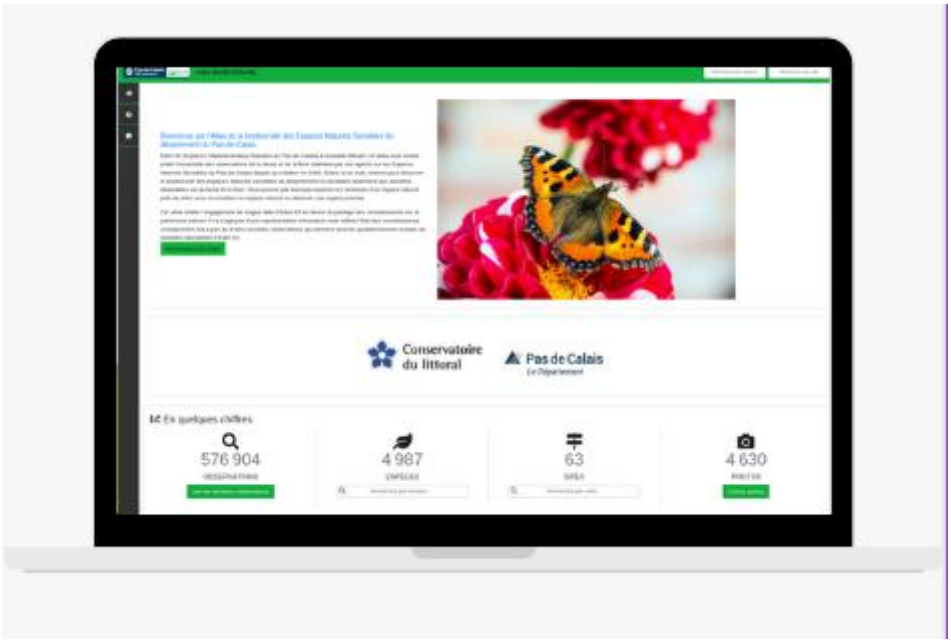
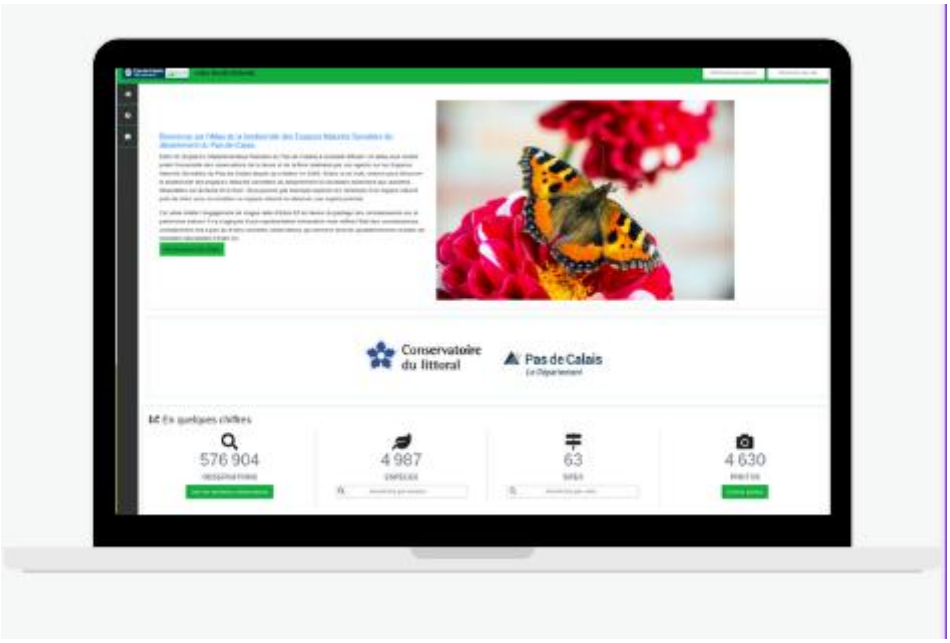
Rapport de stage : Création d'un Atlas de faune et flore



Du 07/04/2025 au 27/06/2025

Présenté par SITA-MOKOKO Geneviève Marlyse
Etudiante en BUT2 de Science des Données parcours VCOD
Maitre de Stage : Lydie DELAYEN Responsable SIG
Tuteur pédagogique : Pascal Di Scala

Rapport de stage : Création d’un Atlas de faune et flore



Remerciements

Tout d’abord je tiens à remercier chaleureusement ma maître de stage : Lydie DELAYEN au sein de Eden62, pour sa confiance, son encadrement et son soutien tout au long de ce projet. Grâce à elle, j’ai pu évoluer dans un environnement de travail enrichissant et dynamique. Je tiens aussi à remercier grandement Romain GARENAUX pour ses précieux conseils, son attention.

Je remercie aussi toute l’équipe d'Eden 62 pour l’accueil chaleureux et la bonne ambiance de travail je me suis facilement intégrée grâce à vous tous !! Merci également à mon tuteur de stage Pascal Di Scala, pour sa présence et son suivi.

Merci à vous tous pour votre bienveillance et votre soutien, ça m'a beaucoup plu !

Sommaire

Table des illustrations

Lexique

API (Application Programming Interface)
Interface de programmation qui permet à différents logiciels ou applications de communiquer entre eux. Dans le cadre d'un atlas web, une API peut servir à interroger la base de données pour afficher dynamiquement des cartes ou des fiches espèces.

AppGens v2.0
Application métier développée par Eden 62 pour gérer les données naturalistes collectées sur les sites. Elle s'appuie sur une base de données centralisée (GENS) et intègre des fonctionnalités de consultation, de saisie et d'analyse des observations.

Atlas de la biodiversité
Application cartographique permettant de visualiser et consulter des données naturalistes. Elle est destinée à valoriser les observations d'espèces sur un territoire donné auprès du grand public ou des partenaires.

Base de données
Système structuré de stockage de données permettant leur consultation, leur mise à jour et leur traitement. Dans le cadre du projet, les données naturalistes sont stockées dans une base PostgreSQL alimentée par AppGens.

Donnée naturaliste
Information issue de l'observation d'une espèce animale ou végétale dans un lieu et à un instant donné. Elle peut inclure des précisions sur l'espèce, la localisation, la date, le milieu, le comportement, etc.

Eden 62
Syndicat mixte départemental créé en 1993 par le Département du Pas-de-Calais pour gérer les Espaces Naturels Sensibles (ENS) du territoire.

ENS (Espaces Naturels Sensibles)
Sites identifiés par les départements comme présentant un intérêt écologique, paysager ou patrimonial. Ils sont protégés, aménagés et souvent ouverts au public.

GeoNature
Suite d'outils open source développée par des structures gestionnaires d'espaces naturels (notamment les parcs nationaux) pour la gestion, l'analyse et la diffusion des données naturalistes.

GeoNature-atlas
Module web de GeoNature permettant de publier en ligne les données d'observations naturalistes via des interfaces cartographiques et interactives.

Leaflet
Bibliothèque JavaScript open source permettant de créer des cartes interactives sur le web. Souvent utilisée dans les applications SIG web comme GeoNature-atlas.

Open source
Logiciel dont le code source est accessible, modifiable et réutilisable librement. Cela favorise la collaboration, la transparence et la pérennité des outils.

Phénologie
Étude des cycles saisonniers des espèces (floraison, migration, reproduction, etc.). Les données phénologiques permettent de suivre les effets du climat sur la biodiversité.

PostgreSQL / PostGIS
PostgreSQL est un système de gestion de base de données relationnelle. PostGIS est son extension spatiale, qui permet de stocker et d'interroger des données géographiques.

SIG (Système d'Information Géographique)
Ensemble d'outils et de méthodes permettant de gérer, analyser et représenter des données spatialisées. Indispensable en écologie pour cartographier des habitats, espèces ou pressions.

Taxonomie
Science de la classification des êtres vivants. En informatique naturaliste, elle structure les données selon un référentiel hiérarchique (règne, embranchement, famille, genre, espèce, etc.) pour permettre des recherches cohérentes.

Territoire d'action
Zone géographique dans laquelle une structure (comme Eden 62) exerce sa mission de gestion, de surveillance et de valorisation de la nature.

Visualisation cartographique
Représentation graphique de données sur une carte, souvent interactive, permettant une compréhension spatiale rapide (ex : répartition des espèces, sites d'observations, etc.).

Taxon : un taxon est une unité quelconque (genre, famille, espèce, sous-espèce, etc.) issu de la hiérarchie taxonomique. Généralement le terme est employé pour désigner l'espèce ou la sous-espèce.

Base de données spatiale : base de données permettant de stocker des informations géolocalisées (points, lignes, polygones) grâce à un type de données nommé Geometry.

Vue-matérialisée : à la différence des « vues » traditionnelles qui sont des représentations virtuelles d'une requête sur une table existante, les « vues-matérialisées » contiennent physiquement les données issues d'une requête.

FDW (Foreign Data Wrapper) : mécanisme de base de données consistant à créer une base fille, vide de toute données, à partir d'une table « mère » existante. La base fille est un simple miroir de la base mère sur laquelle on peut régler les droits d'admission à la base mère.

Introduction

Eden 62, où je réalise mon stage, est un syndicat mixte départemental chargé de la gestion des Espaces Naturels Sensibles (ENS) dans le Pas-de-Calais. En France, les ENS ont été créés par la loi du 18 juillet 1985 dans le but de préserver des milieux naturels remarquables ou menacés. Le Département du Pas-de-Calais a confié la gestion de ces sites à Eden 62 dès sa création en 1993.

L'organisme assure aujourd'hui la gestion de plus de 50 sites naturels répartis sur l'ensemble du territoire départemental, représentant une grande diversité d'habitats : dunes, marais, coteaux calcaires, terrils, boisements humides… Pour remplir ses missions de préservation, de gestion et de sensibilisation, Eden 62 collecte depuis de nombreuses années un grand volume de données naturalistes, portant sur la faune, la flore et les habitats.

Historiquement, ces données étaient essentiellement utilisées en interne, notamment dans le cadre des plans de gestion. Toutefois, à l'heure de la transition numérique et de l'ouverture croissante des données environnementales, Eden 62 s'inscrit aujourd'hui dans une dynamique de modernisation de ses outils et de diffusion de l'information.

Afin d'accompagner cette évolution, l'établissement s'est doté d'un système d'information interne structuré autour de l'application métier **AppGens v2.0**, développée en interne. Cette application centralise les données métiers au sein de la base **GENS** (Gestion des Espaces Naturels Sensibles), qui constitue aujourd'hui le socle de l'information écologique de la structure.

Dans le cadre de la valorisation de ces données, Eden 62 a engagé un projet de développement d'un **atlas en ligne de la biodiversité**, s'appuyant sur le moteur **GeoNature-atlas**, une application web libre développée initialement par le Parc national des Écrins. Cet outil permet de consulter de manière interactive les observations naturalistes, par taxon, par site, ou par période, dans une logique à la fois scientifique et pédagogique.

C'est sur ce projet que porte mon stage de 5 mois au sein du **département Informatique/SIG** d'Eden 62. Mon rôle consiste à mettre en place cet atlas web en connectant l'outil GeoNature-atlas aux données contenues dans la base GENS, tout en assurant leur structuration, leur cohérence et leur valorisation.

Dans ce rapport de stage, je présenterai dans un premier temps l'organisation d'Eden 62, le département Informatique/SIG et l'architecture de l'application AppGens. Dans un second temps, j'exposerai les enjeux et le fonctionnement de GeoNature-atlas ainsi que le processus de connexion avec la base GENS. Enfin, je détaillerai l'architecture de l'atlas, les développements réalisés ainsi que les perspectives d'évolution pour cet outil de diffusion.

1 Présentation de la structure : Eden 62

Eden 62 est un syndicat mixte départemental créé en 1993 par le Conseil départemental du Pas-de-Calais. Sa mission principale est la gestion des **Espaces Naturels Sensibles (ENS)** sur le territoire départemental. Ces espaces, désignés en raison de leur intérêt écologique, paysager ou patrimonial, font l'objet d'actions spécifiques de préservation, de restauration, d'aménagement et de sensibilisation.

L'organisme assure la **gestion de plus de 50 ENS**, représentant une diversité de milieux naturels : dunes littorales, marais, terrils, boisements humides, coteaux calcaires, etc. Ces sites sont répartis sur l'ensemble du département, ce qui confère à Eden 62 une connaissance fine et territorialisée de la biodiversité locale.

Pour assurer ses missions, Eden 62 repose sur une équipe pluridisciplinaire regroupant des techniciens de terrain, des naturalistes, des éducateurs à l'environnement, des agents administratifs, ainsi qu'un **département Informatique/SIG** chargé de la structuration, de l'exploitation et de la valorisation des données.

Outre la gestion écologique des sites, Eden 62 mène des actions de **sensibilisation** auprès des scolaires et du grand public, via des animations nature, des événements et des outils pédagogiques. L'organisme travaille également en étroite collaboration avec les collectivités locales, les services de l'État, les associations naturalistes et les citoyens.

1.1 Missions du département Informatique/SIG

Le département **Informatique/SIG** d'Eden 62 joue un rôle stratégique dans la centralisation, le traitement et la valorisation des données produites par les équipes de terrain et les partenaires. Il intervient à la croisée des enjeux **écologiques, techniques et numériques**, dans le cadre d'une gestion de plus en plus data-centrée.

Ses missions principales sont les suivantes :

- **Développement et maintenance des outils métiers** : le département est responsable du développement de l'application interne **AppGens v2.0**, utilisée pour saisir, structurer et exploiter les données métiers collectés sur les ENS.

- **Gestion des bases de données** : il administre la base **GENS**, qui constitue le socle informationnel de l'ensemble des observations faune/flore et qui centralise les informations relatives sa la gestion (planification des plans de gestion et suivi des opérations).

- **Production cartographique** : à l'aide de logiciels SIG (notamment QGIS), le service réalise des cartes thématiques pour les plans de gestion, les suivis écologiques ou la communication interne/externe.

- **Structuration des flux de données** : il assure l'interopérabilité entre les différents systèmes (bases internes, outils open source comme GeoNature) et gère l'intégrité, la qualité et la sécurité des données.

- **Développement web SIG** : il conçoit ou intègre des applications web permettant de visualiser et de partager des données géographiques, à destination des partenaires ou du grand public (ex : projet d'atlas en ligne).

- **Appui technique aux autres services** : le département accompagne les équipes naturalistes, techniques ou administratives dans l'utilisation des outils numériques, la saisie des données, ou encore l'analyse spatiale.

Dans un contexte d'ouverture des données et de valorisation numérique de la biodiversité, ce département est un acteur clé de la modernisation des pratiques de gestion environnementale au sein d'Eden 62.

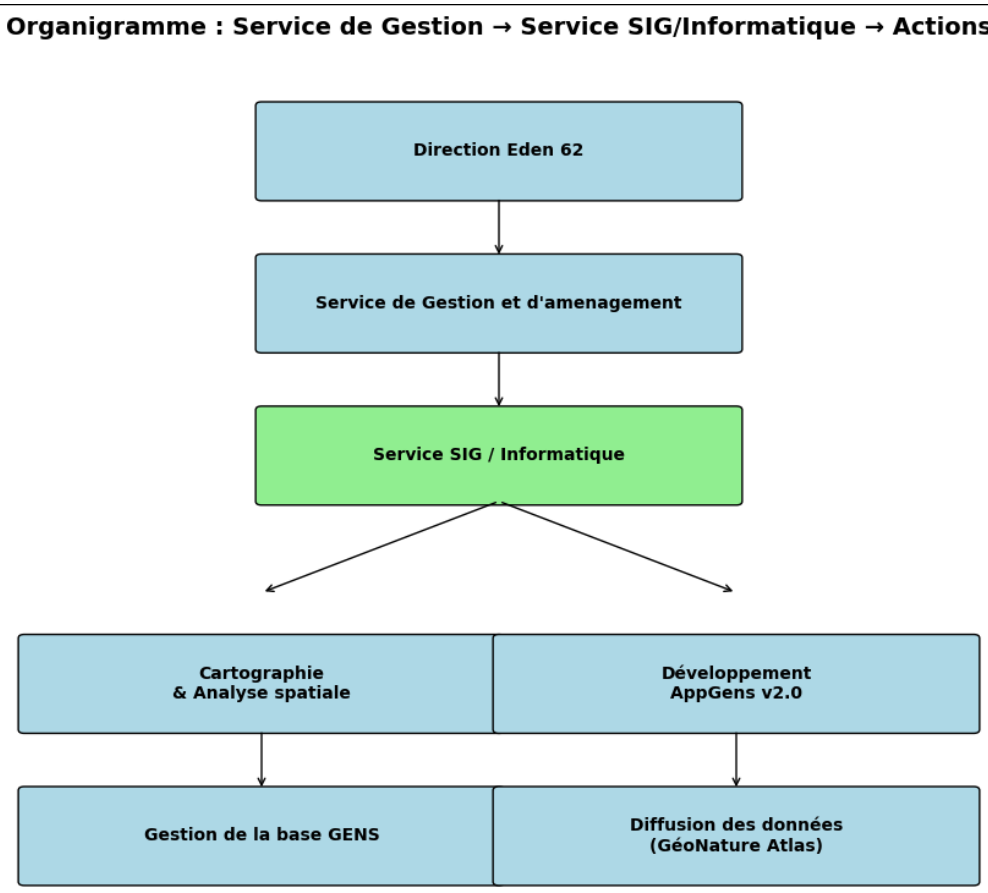


Figure 1-1-Organigramme du Service SIG/Informatique

2 Présentation de GENS

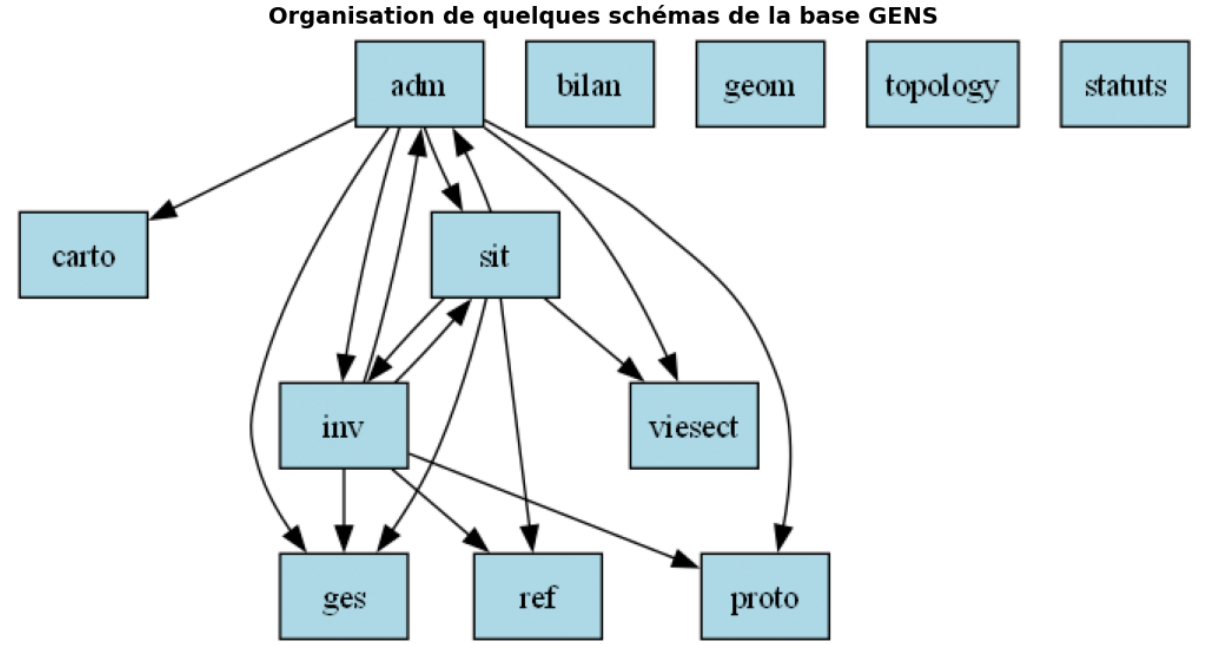


Figure 2-1-Présentation des différents schémas présents dans GENS

En interne, Eden 62 a développé une application métier nommée **AppGENS v2.0**. Cette application permet de **centraliser l'ensemble des données naturalistes** collectées sur les différents sites naturels gérés par l'organisme. Elle permet aussi d'intégrer la planification des actions de gestion et d'en suivre la réalisation. Chaque garde-nature ou agent peut y saisir des informations relatives aux espèces observées durant les suivis écologiques réalisés sur le terrain, gérer des campagnes d'inventaire et exporter les données saisies.

Toutes ces informations sont stockées dans une base de données centrale appelée **GENS** (*Gestion des Espaces Naturels Sensibles*). Il s'agit d'une **infrastructure relationnelle complexe**, structurée en **19 schémas** contenant plus d'une centaine de tables. Ces schémas sont interconnectés grâce à des clés étrangères permettant de relier les différents types de données entre eux.

Parmi ces schémas, **quatre jouent un rôle clé** dans l'intégration des données dans GeoNature Atlas :

- **Inv** : contient les informations sur les inventaires et les campagnes d'observations. C'est ici que sont enregistrées les occurrences naturalistes (date, lieu, espèce, observateur, etc.).
- **Sit** : regroupe les données relatives aux sites d'étude, notamment leurs **noms**, **types** et **géométries** (polygones décrivant les périmètres des sites gérés).
- **Adm** : recense les agents d'Eden 62, avec leurs identifiants et rôles. Il permet de faire le lien entre les observations et les personnes qui les ont réalisées.

- **Ref** : contient le **référentiel taxonomique** utilisé dans GENS, incluant les noms latins et vernaculaires, les statuts de protection, les groupes faune/flore, etc.

Cette structuration permet à Eden 62 d'assurer une gestion fine, collaborative et historisée de ses données métiers. Elle facilite aussi leur réutilisation, notamment dans le cadre d'outils de valorisation comme **GeoNature Atlas**, en s'appuyant sur des scripts d'extraction et de transformation adaptés.

2.1 Présentation de l'application métier AppGens v2.0

Figure 2-2-Interface de AppGens v2.0

AppGens v2.0 est l'application métier développée en interne par le service SIG/Informatique d'Eden 62. Elle constitue l'interface principale permettant la **gestion, la consultation et l'enrichissement de la base de données GENS**, utilisée pour centraliser les observations faunistiques et floristiques réalisées sur les Espaces Naturels Sensibles (ENS) du département du Pas-de-Calais.

Accessible via une interface web interne AppGens v2.0 répond à plusieurs objectifs :

- **Saisie structurée** des observations de terrain par les agents ;
- **Consultation rapide** des données par espèce, par site ou par période ;
- **Suivi spatio-temporel** des relevés naturalistes ;

- **Exportation des données** pour traitement cartographique ou analytique (ex : via QGIS ou GeoNature) ;

- **Gestion fine des droits utilisateurs**, en fonction du profil (agent, expert, administrateur…).

L'application repose sur une architecture technique robuste :

- Une base de données PostgreSQL/PostGIS(GENS) ;

- Un développement web personnalisé basé sur des technologies modernes (PHP, JavaScript, HTML/CSS) ;

- Une intégration partielle avec des outils complémentaires comme QGIS ou GeoNature.

AppGens v2.0 joue un rôle central dans l'écosystème numérique d'Eden 62, en tant que **point d'entrée et de structuration des données métiers** avant leur diffusion éventuelle sur des plateformes ouvertes comme **GeoNature Atlas**

3 Présentation de GeoNature et GeoNature Atlas

3.1 GeoNature

« Nos outils sont pensés, dès le départ, pour pouvoir être déployés par d'autres structures dans des contextes différents. Pour cela, l'accent est mis sur des développements génériques et sur la publication de l'outil sous licence libre pour en faciliter l'utilisation par d'autres ».

- Camille Monchicourt, responsable du système d'informations au Parc national des Ecrins

GeoNature est une application open-source sous licence libre développée initialement par le Parc national des Écrins, aujourd'hui maintenue et enrichie par une communauté d'utilisateurs regroupant des parcs naturels, conservatoires, départements, syndicats mixtes et bureaux d'études. Elle permet de **centraliser, gérer, structurer et valoriser des données naturalistes** (faune, flore), issues d'observations de terrain, d'inventaires ou de suivis écologiques.

L'application est conçue pour répondre aux besoins des structures publiques en matière de :

- Saisie et consultation d'observations naturalistes ;
- Suivi d'espèces et de protocoles scientifiques ;
- Gestion des référentiels taxonomiques, des utilisateurs et des droits ;
- Structuration des données pour la diffusion vers des plateformes partenaires comme l'INPN ou l'Observatoire de la biodiversité.

3.2 GeoNature Atlas

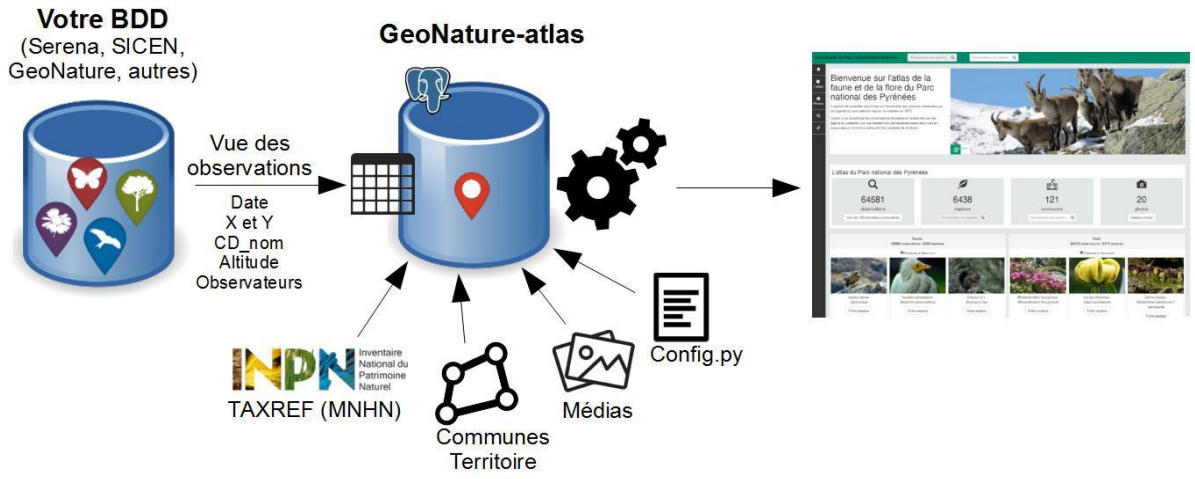


Figure 3-1-Description du fonctionnement global de geonature-atlas

GeoNature Atlas est un module complémentaire de la suite GeoNature. Il s'agit d'une application web open source dédiée à la **diffusion des données naturalistes** vers le grand public ou des partenaires, sous la forme d'un **atlas dynamique et interactif**.

Développé par le **Parc national des Écrins**, cet outil permet notamment :

- La **visualisation cartographique** des espèces observées sur un territoire donné ;
- La **consultation de fiches espèces** (répartition, période de présence, statuts biologiques ou réglementaires, etc.) ;
- L'accès à des **filtres par groupes taxonomiques**, périodes, statuts ou zones géographiques ;
- Une **mise à disposition pédagogique et transparente** des données de biodiversité.

Pour **Eden 62**, GeoNature Atlas est l'outil retenu pour **valoriser et rendre accessibles les données issues de la base GENS**, à travers une interface claire et attractive, destinée aux **citoyens, élus, partenaires techniques ou scientifiques**. Il s'inscrit dans une démarche de **transparence et de valorisation** des actions de **gestion et de suivi des espaces naturels sensibles** du département.

Dans le cadre de ce projet, l'objectif était de **déployer GeoNature Atlas en l'adaptant à un contexte local** sans instance complète de GeoNature. L'outil s'appuie pour cela sur un **modèle de données simplifié**, optimisé pour la publication d'observations, de taxons et de données géographiques. Le modèle conceptuel de données (MCD) suivant présente les entités principales qui structurent les données utilisées par Atlas.

3.2.1 Structure des données dans GeoNature Atlas

L'application GeoNature Atlas repose sur un modèle de données simplifié, orienté vers la publication. Contrairement à l'application GeoNature complète, plus complexe et conçue pour la saisie et la gestion de données, Atlas se concentre uniquement sur les données utiles à la diffusion : taxons, observations, médias, attributs descriptifs et entités géographiques associées.

3.2.1.1 Le cœur du modèle : les taxons et les observations

Les taxons (espèces, genres, groupes) sont identifiés par leur code référentiel (cd_ref issu du référentiel national faune/flore de l'INPN), et décrits par leur nom latin, leur nom vernaculaire, et parfois par des attributs supplémentaires tels que leur milieu de vie, statut de protection, ou des commentaires explicatifs.

Les observations sont rattachées à ces taxons à l'aide du même cd_ref, et contiennent :

- Une date d'observation,
- Une localisation géographique (point ou maille),
- Et des informations complémentaires comme l'altitude ou la précision.

3.2.1.2 Des données enrichies : attributs, médias et géographie

Chaque taxon peut être enrichi de :

- Médias (photos, sons, vidéos) pour améliorer l'aspect pédagogique de l'atlas,
- Attributs descriptifs (régime alimentaire, habitat, etc.),
- Et informations de répartition, par maille ou commune, permettant une représentation cartographique simplifiée.

Les données géographiques sont stockées sous forme de polygones (mailles ou communes) permettant d'agréger les observations et de dégrader la précision de la localisation exacte si nécessaire (pour des raisons de préservation de l'espèce).

3.2.1.3 Un modèle optimisé pour la publication

Ce modèle relationnel permet à GeoNature Atlas de générer dynamiquement :

- Des cartes de répartition par taxon,
- Des fiches espèces détaillées,
- Des statistiques par période ou zone géographique.

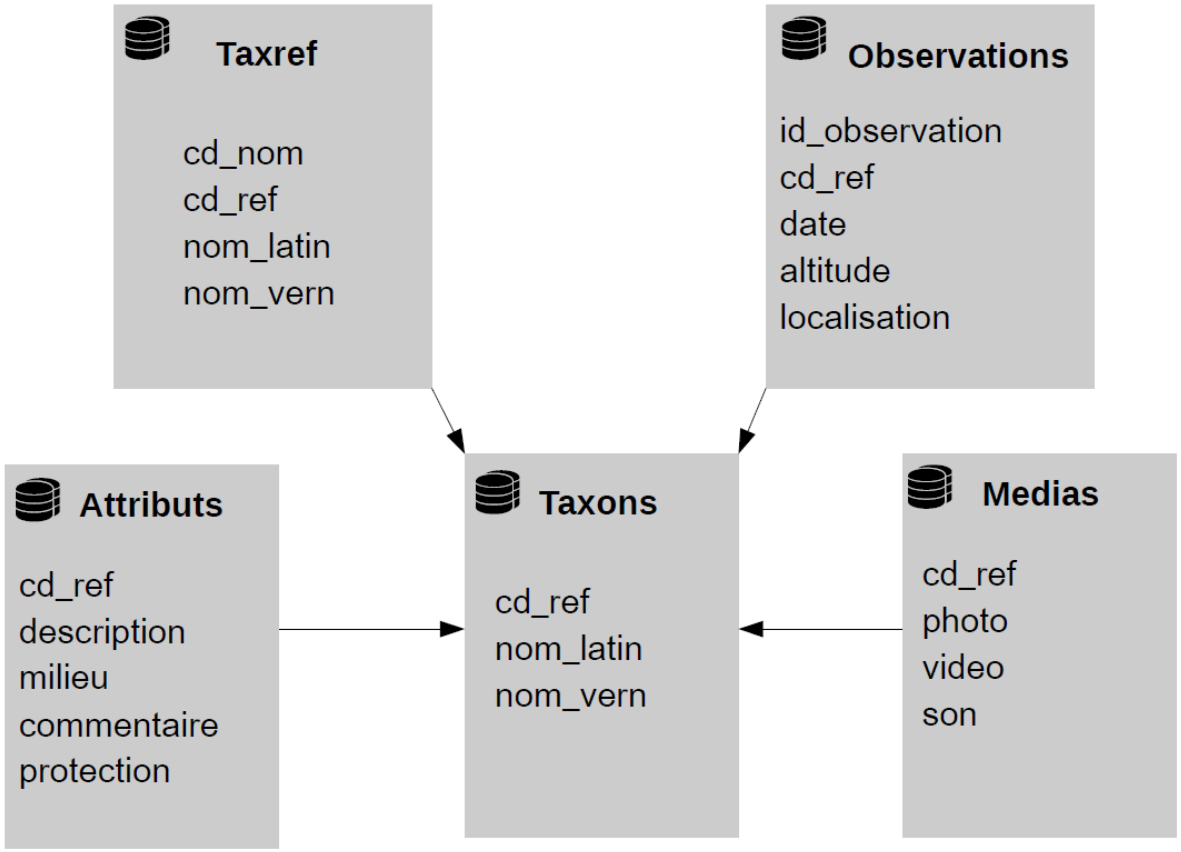


Figure 3-2-Schéma relationnel

3.2.2 Analyse technique du MCD principal de GeoNature Atlas

Le modèle conceptuel de données (MCD) de GeoNature Atlas repose sur une **organisation relationnelle claire et modulaire**, optimisée pour la **diffusion d'observations naturalistes**. Voici les principaux éléments qui structurent la base.

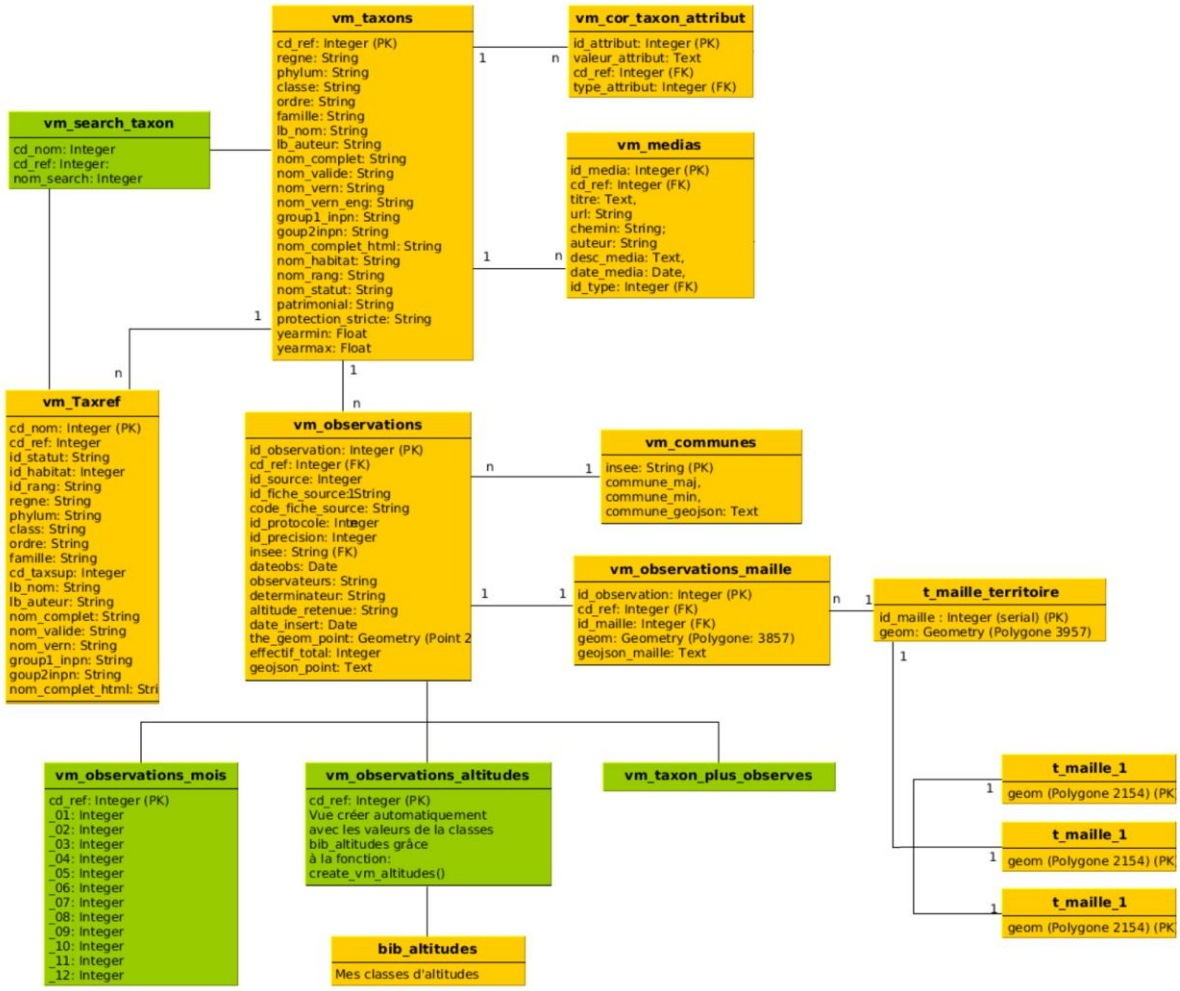


Figure 3-3-MCD principal de geonature-atlas avec création des VM correspondantes

3.2.2.1 Taxonomie (vm_taxons, vm_Taxref)

- **vm_taxons** est la table pivot du référentiel taxonomique. Elle contient toutes les informations nécessaires à l'affichage dans l'atlas :
    - Identifiants taxonomiques **(cd_ref, cd_nom**),
    - Noms (scientifiques, vernaculaires),
    - Hiérarchie (règne, phylum, classe…),
    - Informations de protection, patrimonialité, etc.
- **vm_Taxref** est une table source ou de travail, plus brute, importée depuis le référentiel **TAXREF**. Elle peut être utilisée pour reconstruire ou mettre à jour la table **vm_taxons**.

3.2.2.2 *Observations (vm_observations)*

- Cette table centralise les données d'observation :
  - cd_ref pour relier à l'espèce,
  - insee pour lier à une commune,
  - dateobs, altitude_retenue, the_geom_point, etc.
- On y retrouve aussi les informations sur le protocole, le nombre d'individus observés (effectif_total), les observateurs, et la source.
- Elle alimente des **vues thématiques** :
  - vm_observations_mois : pour les statistiques mensuelles,
  - vm_observations_altitudes : regroupements altimétriques.

3.2.2.3 *Médias (vm_medias)*

- Permet d'associer un ou plusieurs médias (photo, son, vidéo) à un taxon via cd_ref.
- Les champs url, chemin, desc_media, et date_media permettent l'affichage enrichi dans les fiches taxons.

3.2.2.4 *Attributs des taxons (vm_cor_taxon_attribut)*

- Permet de lier des **attributs descriptifs** à chaque taxon (valeur_attribut, type_attribut), en fonction du cd_ref.
- Cela alimente les sections "habitat", "milieu", "description" dans l'interface Atlas.

3.2.2.5 *Géographie (vm_communes, t_maille_territoire)*

- **vm_communes** référence les communes françaises avec un champ geom pour la géométrie.
- **t_maille_territoire** contient les mailles d'analyse spatiale (souvent en projection 2154 ou 3957), utilisées pour agréger les observations de manière anonyme.
- La table vm_observations_maille fait le lien entre une observation et la maille où elle est localisée.

3.2.2.6 Autres vues / tables complémentaires

- vm_taxon_plus_observes : vue générée dynamiquement pour les taxons les plus fréquents.
- bib_altitudes : table paramétrique pour la génération des classes d'altitudes.

3.2.3 Organisation du code

Le code source de GeoNature Atlas est organisé de manière modulaire selon une architecture claire inspirée du modèle MVC (Modèle - Vue - Contrôleur), facilitant la maintenance et l'évolution du projet. Voici les principaux composants :

3.2.3.1 Modèles (Repositories)

Cette couche regroupe les classes de liaison avec la base de données. Chaque entité métier possède son propre *repository* (ex. : TaxonRepository, ObservationRepository), intégrant les requêtes SQL nécessaires à l'interrogation des données (via SQLAlchemy et parfois du SQL brut).
Ces fichiers gèrent la logique d'accès aux données (taxons, observations, communes, etc.).

3.2.3.2 Vues (Templates)

Les vues correspondent aux fichiers HTML générés avec **Jinja**. On y trouve :

- La page d'accueil
- Les fiches espèces
- Les fiches communes
- Les pages par rangs taxonomiques
- Les listes filtrables (groupes, statuts, milieux, etc.)

Chaque vue est conçue pour être dynamique et responsive, avec des données injectées depuis les contrôleurs.

3.2.3.3 Contrôleurs (Routing)

Les contrôleurs gèrent les routes (URLs) de l'application. Chaque route correspond à une vue ou une action. Par exemple :

- /atlas → page d'accueil
- /espece/<cd_ref> → fiche espèce
- /commune/<insee> → fiche commune

- /liste/<cd_ref> → liste filtrée des taxons

Ces routes sont définies en Python via le micro-framework Flask.

3.2.3.4 *Configuration*

- config.py : fichier principal de configuration de l'application (base de données, options d'affichage, chemins d'accès, etc.).
- custom.css : fichier dédié à la personnalisation graphique de l'interface (couleurs, logo, typographie, etc.), permettant d'adapter l'atlas à la charte graphique d'une structure comme Eden 62.

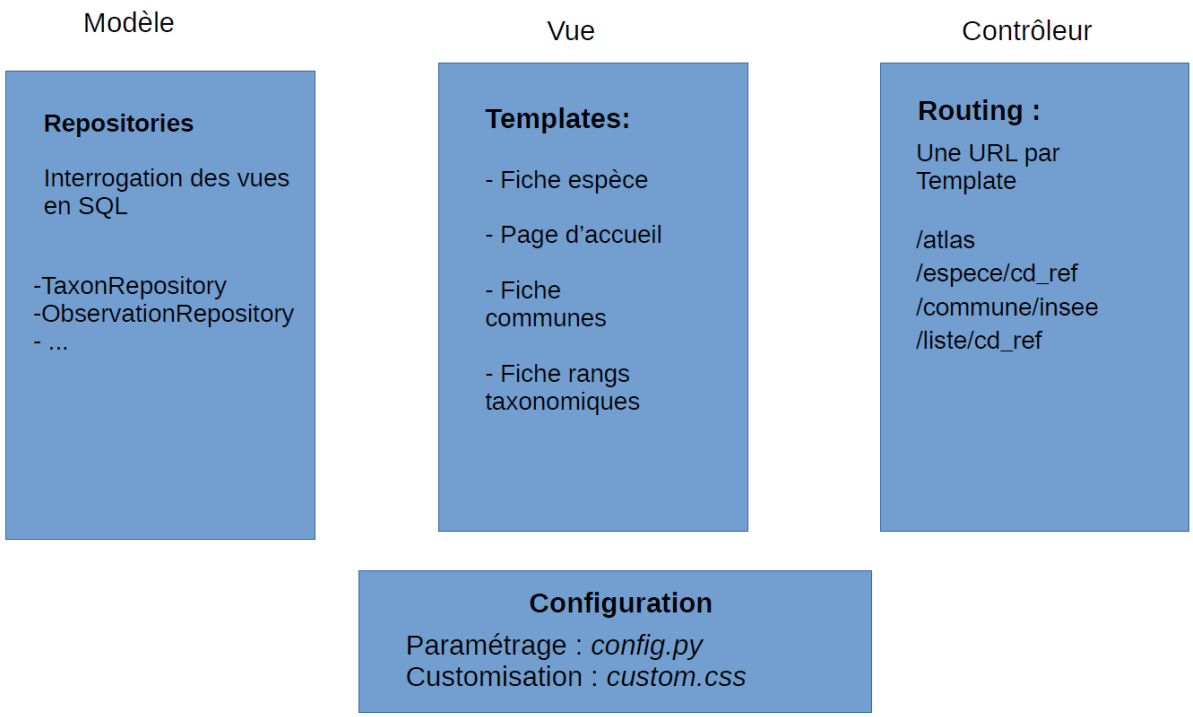


Figure 3-4-Schéma organisationnel de geonature-atlas

3.2.4 Architecture de l'application

L'architecture de GeoNature Atlas repose sur une séparation claire entre les différentes couches logicielles, selon un schéma client-serveur :

3.2.4.1 *Base de données*

Les données naturalistes sont stockées dans une base PostgreSQL enrichie avec l'extension PostGIS pour la gestion des données géographiques. Ces données incluent les observations, la taxonomie (TAXREF), les médias, les attributs, les communes, etc.

3.2.4.2 *Back-end (côté serveur)*

Le back-end s'appuie sur plusieurs composants :

- **ORM SQLAlchemy** : il permet d'interfacer la base de données avec le code Python, facilitant les requêtes et la manipulation des données.
- **Flask** : ce micro-framework gère le routage des requêtes. Chaque URL correspond à une route définie côté serveur qui traite les données et les transmet à une vue (template).
- **Templates Jinja** : les vues HTML sont générées dynamiquement à partir des données renvoyées par le back-end.

3.2.4.3 *Front-end (côté client)*

Le côté client repose sur des technologies légères :

- **JavaScript / jQuery** : pour capturer les actions des utilisateurs (sélection d'espèce, de commune, recherche, navigation, etc.) et envoyer des requêtes au serveur.
- **Leaflet** : pour l'affichage cartographique interactif des données.
- **Morris.js & Bootstrap** : pour les graphiques (histogrammes de répartition temporelle, par exemple) et la mise en forme CSS.

3.2.4.4 *Cycle de fonctionnement*

Lorsqu'un utilisateur effectue une action (ex. : recherche d'une espèce), une requête est envoyée au serveur. Celui-ci interroge la base via l'ORM, transmet les résultats à la vue HTML concernée, puis renvoie la page générée au navigateur. Ce processus est répliqué dynamiquement pour chaque interaction, garantissant une navigation fluide et interactive.

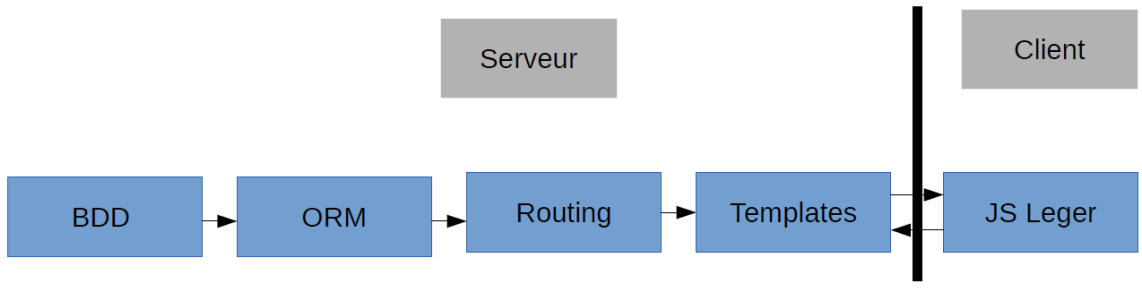


Figure 3-5-Schéma du cycle de fonctionnement de geonature-atlas

3.2.5 Technologies utilisées

GeoNature Atlas repose sur une architecture web moderne combinant des technologies robustes côté back-end et front-end. Ces choix techniques assurent la performance, la maintenabilité et la richesse fonctionnelle de l'application.

Figure 3-6-Outils utilisés

3.2.5.1 *Back-end*

- **Base de données** : PostgreSQL, enrichie avec l'extension **PostGIS** pour le traitement et la requête de données géographiques.
- **ORM** : **SQLAlchemy** est utilisé pour la gestion des interactions entre les objets Python et la base de données relationnelle.
- **Framework web** : **Flask** assure le routage des requêtes HTTP et le traitement de la logique serveur.
- **Moteur de templates** : **Jinja** est employé pour générer dynamiquement les pages HTML côté serveur.

3.2.5.2 Front-end

- **DOM & animations** : **jQuery** est utilisé pour manipuler le DOM, gérer les événements et assurer des animations fluides.

- **Cartographie interactive** : **Leaflet** permet d'afficher les cartes et les observations géoréférencées de manière interactive et légère.

- **Graphiques** : **Morris.js** est intégré pour représenter les données sous forme de graphiques simples et clairs.

- **Interface utilisateur** : **Bootstrap** assure une mise en page responsive et cohérente sur tous les types de terminaux, en facilitant le stylage CSS.

Ces technologies, éprouvées et largement utilisées dans l'écosystème open source, garantissent la stabilité de GeoNature Atlas tout en facilitant son adaptation et sa personnalisation pour les besoins d'acteurs comme Eden 62.

4 Déroulement du stage

4.1 Prise en main de la documentation

Dans un premier temps, j'ai exploré la documentation officielle de GeoNature Atlas, qui constitue la base technique du futur portail de diffusion des données naturalistes. Cette documentation structurée m'a permis d'identifier les étapes indispensables au déploiement de l'application.

GeoNature Atlas nécessitant un système Linux (Debian), j'ai installé une machine virtuelle Debian sous Windows afin de disposer d'un environnement compatible. Une fois la machine configurée, j'ai procédé à l'installation des dépendances (Python, Node.js, PostgreSQL/PostGIS, etc.), au déploiement de l'application web, ainsi qu'à la création de la base de données cible.

4.2 Installation de l'environnement

4.2.1 Création de la machine virtuelle

Pour répondre aux exigences de GeoNature-Atlas, j'ai commencé par créer un environnement Linux à l'aide de **VirtualBox**. J'ai téléchargé l'image ISO de **Debian 11**, puis configuré une **machine virtuelle** dédiée. Une fois le système installé, j'ai suivi les instructions fournies dans la documentation du projet pour effectuer les premières configurations : mise à jour des paquets, installation de certains outils de base comme unzip et sudo.

4.2.2 Mise en place de l'environnement de travail

Après la configuration du système, j'ai téléchargé la dernière version de **GeoNature-Atlas** depuis son dépôt GitHub, sous forme d'archive .zip. J'ai ensuite extrait le contenu dans un répertoire que j'ai nommé atlas, avant de supprimer l'archive pour ne conserver que les fichiers nécessaires.

L'étape suivante consistait à **installer l'environnement logiciel** requis pour faire fonctionner GeoNature-Atlas. Pour cela, j'ai exécuté le script install_env.sh depuis le terminal. Ce script automatise l'installation des dépendances (Python, Node.js, PostgreSQL/PostGIS, etc.) et prépare l'environnement pour le déploiement de l'application.

4.2.3 Mise en place de l'interopérabilité avec la base GENS

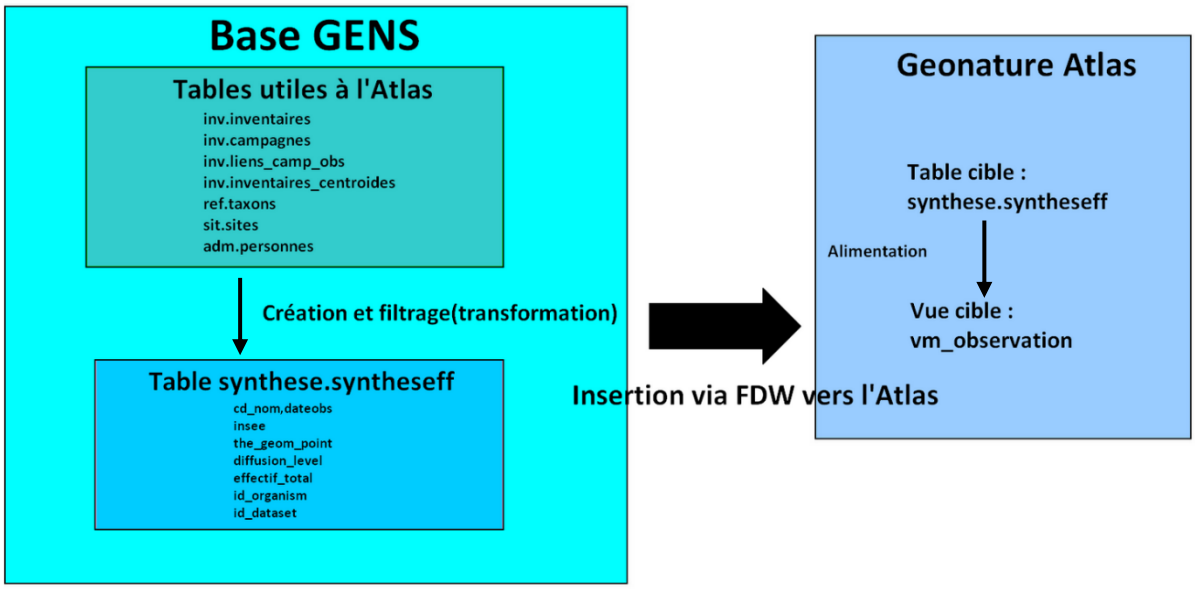


Figure 4-1-Connexion GENS à geonature-atlas

Le cœur du projet reposait sur l'intégration des données naturalistes issues de la base interne d'Eden 62 (nommée **GENS**) dans GeoNature Atlas. Plutôt que de migrer l'ensemble des données, j'ai utilisé une **Foreign Data Wrapper (FDW)** pour établir un lien dynamique entre la base GENS (PostgreSQL) et celle de GeoNature Atlas. Cela permettait de consulter les données en temps réel sans les dupliquer.

Cependant, la structure de GENS ne correspondait pas directement aux exigences de GeoNature Atlas. J'ai donc dû adapter les données via des requêtes SQL complexes pour :

> **Créer les structures attendues par l'Atlas**, notamment la table synthese. syntheseff, conçue pour stocker les données d'observations au format standard de l'application :

```sql
-- Schéma synthese à créer si besoin
CREATE SCHEMA IF NOT EXISTS synthese;

-- Création de la table syntheseff
CREATE TABLE IF NOT EXISTS synthese.syntheseff (
    id_synthese SERIAL PRIMARY KEY,
    id_organism INTEGER DEFAULT 2,
    id_dataset INTEGER,
    cd_nom INTEGER NOT NULL,
    insee CHARACTER(5),
    dateobs DATE NOT NULL DEFAULT now(),
    observateurs VARCHAR(255),
    altitude_retenue INTEGER,
    the_geom_point GEOMETRY(Point, 4326),
    effectif_total INTEGER,
    diffusion_level INTEGER,
    id_site INTEGER,
    nom_site TEXT,
    supprime BOOLEAN DEFAULT false
);
```

Figure 4-2-Code de création du schéma synthese et de la table syntheseff

Préparer l'import des données, en effectuant un traitement sur les tables sources de GENS. J'ai notamment :

- Transformé les géométries issues des sites pour correspondre au SRID attendu (4326) par l'Atlas ;
- Normalisé les effectifs selon les champs disponibles (total ou preci_t) ;
- Concaténé les noms des observateurs pour chaque observation ;
- Extrait uniquement les observations réalisées ou co-réalisées par un agent d'Eden 62.

Extrait de requête utilisée pour remplir la table syntheseff :

```sql
ALTER TABLE synthese.syntheseff
ALTER COLUMN observateurs TYPE text;
INSERT INTO synthese.syntheseff (
    cd_nom,
    dateobs,
    observateurs,
    altitude_retenue,
    the_geom_point,
    effectif_total,
    diffusion_level,
    insee,
    id_organism,
    id_dataset
)
SELECT
    t.cd_nom,
    c.date_deb AS dateobs, --le schema de l'atlas demandait une seule date pou
    obs.observateurs,
    NULL AS altitude_retenue, -- la zone étant plutôt une zone plate on a pas
    ST_Transform(ST_SetSRID(i.geom_centroid, 2154), 4326) AS the_geom_point --
,
    CASE
        WHEN i.total IS NOT NULL THEN i.total
        WHEN i.total IS NULL AND i.preci_t IS NOT NULL THEN 1 --ici il s'agit
        --la logique était que dans le champ total il y'avait les effectifs ré
        ELSE NULL
    END AS effectif_total,
    2 AS diffusion_level, --Instauré à 2 par defaut lors de la création de la
    s.id AS insee, --identifiants de chaque site issu du champ id_site de la t
  --nom de chaque site issu du champ nom de la table site
    2 AS id_organism,    -- Eden 62 la valeur lors de la création de la table e
    1 AS id_dataset      -- Dataset Observations Eden 62 choisit lors de la cré
FROM inv.inventaires i
JOIN inv.campagnes c ON i.ref_campagne = c.id --jointures avec la table campag
JOIN ref.taxons t ON t.id = i.ref_taxon -- jointure avec la table taxons pour
JOIN sit.sites s ON i.ref_site = s.id --jointure avec la table sites pour récu

-- Sous-requête pour récupérer les observateurs concaténés par campagne
LEFT JOIN (
    SELECT
        l.ref_campagne,
        STRING_AGG(p.nom_comp, ' ; ') AS observateurs -- on récupère et on cont
    FROM inv.liens_camp_obs l --table de récupération des identifiants
    JOIN adm.personnes p ON l.ref_personne = p.id --table de jointure pour réc
    GROUP BY l.ref_campagne
) obs ON obs.ref_campagne = c.id --Table pour récupérer les differents endroit

WHERE t.cd_nom IS NOT NULL --on prend uniquement les codes noms valides dans l
  AND (obs.observateurs LIKE '%Eden 62%' OR c.obs_old LIKE '%Eden 62%') --On f
  AND NOT (i.total IS NULL AND i.preci_t IS NULL); -- On instaure la contrainte
```

Figure 4-3-Code de remplissage de la table syntheseff

Déclaré les métadonnées nécessaires au bon affichage dans l'Atlas, comme les informations sur l'organisme Eden 62 (utilisateurs.bib_organismes), le jeu de données associé (gn_meta.t_datasets) et les liaisons entre les deux (gn_meta.cor_dataset_actor).

4.2.3.1 Calcul des centroïdes

La base GENS stocke les géométries des observations sous forme de **polygones** (zones), alors que GeoNature Atlas attend des **points géographiques** (GEOMETRY (Point, 4326)) pour représenter les localisations. Pour répondre à cette contrainte, j'ai procédé au **calcul du centroïde de chaque polygone d'observation** à l'aide de la fonction ST_Centroid ().

Extrait utilisé pour enrichir les géométries de la table inv. inventaires :

```sql
SELECT
    id,
    ST_Centroid(geometrie) AS geom_centroid
FROM
    inv.inventaires
WHERE
    geometrie IS NOT NULL;

ALTER TABLE inv.inventaires
ADD COLUMN geom_centroid GEOMETRY(Point, 2154);
UPDATE inv.inventaires
SET geom_centroid = ST_Centroid(geometrie)
WHERE geometrie IS NOT NULL;

SELECT
    i.id AS inventaire_id,
    CASE
        WHEN EXISTS (
            SELECT 1
            FROM sit.sites s
            WHERE ST_Intersects(s.coord_l93, ST_Centroid(i.geometrie))
        ) THEN 'Oui'
        ELSE 'Non'
    END AS centroid_dans_site
FROM
    inv.inventaires i
WHERE
    i.geometrie IS NOT NULL;
```

Figure 4-4-Code de création des centroïdes dans GENS

Ces étapes m'ont permis d'intégrer avec précision les données d'Eden 62 dans GeoNature Atlas, tout en respectant son modèle de données. Le processus assurait également une évolutivité future en gardant les données dans leur système d'origine (GENS), tout en les rendant visibles et exploitables via l'interface Atlas.

5 Configuration de la base de données

5.1 Paramétrage du fichier settings.ini

Pour permettre à GeoNature Atlas d'accéder aux données réelles de faune et de flore issues de la base interne **GENS**, il a été nécessaire de modifier le fichier de configuration settings.ini, fourni par défaut avec l'application.

Ce fichier centralise les paramètres de connexion aux bases de données. J'y ai apporté les modifications suivantes afin d'établir une communication sécurisée entre la **base GeoNature Atlas** (hébergée sur la machine virtuelle Debian) et la **base GENS** (hébergée sur un autre poste du réseau local) :

- **Définition des identifiants d'accès aux bases** :
    - owner_atlas : geonatadmin / admin123 (propriétaire des objets de la base Atlas)
    - user_pg : geonatatlas / atlas123 (utilisateur courant de la base Atlas)
    - atlas_source_user : postgres / root (compte administrateur de la base GENS)
- **Configuration du serveur distant** :
    - Récupération de l'**adresse IP locale** de la machine hébergeant GENS (192.168.1.163)
    - Définition de cette IP en tant qu'**hôte distant** dans settings.ini, afin que la VM Debian puisse établir une connexion vers la base source.

Ces ajustements étaient essentiels pour assurer une **authentification croisée** entre les deux bases et préparer l'étape suivante : l'accès aux données de GENS via un **Foreign Data Wrapper (FDW)**.

Aucune autre modification majeure n'a été nécessaire dans ce fichier. Une fois les paramètres enregistrés, la communication entre les deux systèmes a pu être établie avec succès.

5.2 Installation et adaptation de la base de données via install_db.sh

Le script install_db.sh est fourni avec GeoNature Atlas pour automatiser la préparation de la base de données, notamment en ce qui concerne l'intégration des données taxonomiques. Toutefois, dans le cadre du projet Eden 62, plusieurs adaptations ont été nécessaires afin de le rendre compatible avec notre environnement spécifique basé sur le référentiel **TAXREF v18 (2025)**.

5.3 Intégration du référentiel TAXREF v18

5.3.1 Téléchargement et nettoyage des fichiers

Par défaut, le script ne télécharge que des fichiers secondaires liés à la réglementation (e.g. espèces protégées), sans inclure le fichier principal de TAXREF. J'ai donc modifié la section de téléchargement pour y intégrer **TAXREF_v18_2025.zip**, correspondant à la version réellement utilisée par Eden 62.

Une étape de nettoyage a également été ajoutée afin de supprimer certaines colonnes inutiles (CD_BA, URL_INPN) à l'aide de la commande cut.

5.3.2 Mise en conformité structurelle

Le script initial était conçu pour fonctionner avec **TAXREF v14**, ce qui posait problème lors du croisement avec notre base GENS. En effet, le nombre de champs et leur structure avaient évolué entre la v14 et la v18.

Pour gérer cette incompatibilité, j'ai créé une **table intermédiaire temporaire (temp_taxref)** contenant tous les champs de la version 18. Cette table permettait d'importer l'intégralité des données, puis d'extraire uniquement les colonnes nécessaires pour peupler la table taxref attendue par l'Atlas. Une fois le traitement terminé, la table temporaire était supprimée.

```sql
DROP TABLE IF EXISTS temp_taxref;
CREATE TEMP TABLE temp_taxref (
    regne TEXT,phylum TEXT,classe TEXT,ordre TEXT,famille TEXT,sous_famille TEXT,tribu TEXT,group1_inpn
TEXT,group2_inpn TEXT,group3_inpn TEXT,         -- <- champ en trop pour ta version cible
cd_nom INTEGER,cd_taxsup INTEGER,cd_sup INTEGER,cd_ref INTEGER,cd_ba TEXT,              -- <- à ignorer
ensuite
rang TEXT,lb_nom TEXT,lb_auteur TEXT,nomenclatural_comment TEXT,  -- <- champ en trop
nom_complet TEXT,nom_complet_html TEXT,nom_valide TEXT,nom_vern TEXT,nom_vern_eng TEXT,habitat TEXT,fr
TEXT,gf TEXT,mar TEXT,gua TEXT,sm TEXT,sb TEXT,spm TEXT,may TEXT,epa TEXT,reu TEXT,sa TEXT,ta TEXT,taaf
TEXT,pf TEXT,nc TEXT,wf TEXT,cli TEXT,url TEXT,url_inpn TEXT            -- <- à ignorer
);

---import taxref--
TRUNCATE TABLE temp_taxref;
COPY temp_taxref
FROM  '/tmp/taxhub/TAXREFv18.txt'
WITH  CSV HEADER
DELIMITER E'\t' QUOTE '"' NULL ''encoding 'UTF-8';

INSERT INTO taxonomie.import_taxref (
    regne, phylum, classe, ordre, famille, sous_famille, tribu,
    group1_inpn, group2_inpn, cd_nom, cd_taxsup, cd_sup, cd_ref,
    rang, lb_nom, lb_auteur, nom_complet, nom_complet_html,
    nom_valide, nom_vern, nom_vern_eng, habitat, fr, gf, mar,
    gua, sm, sb, spm, may, epa, reu, sa, ta, taaf, pf, nc, wf,
    cli, url
)
SELECT
    regne, phylum, classe, ordre, famille, sous_famille, tribu,
    group1_inpn, group2_inpn, cd_nom, cd_taxsup, cd_sup, cd_ref,
    rang, lb_nom, lb_auteur, nom_complet, nom_complet_html,
    nom_valide, nom_vern, nom_vern_eng, habitat, fr, gf, mar,
    gua, sm, sb, spm, may, epa, reu, sa, ta, taaf, pf, nc, wf,
    cli, url
FROM temp_taxref;
```

Figure 5-1-Création de la table temporaire et insertion du taxref

5.3.3 Préservation des fichiers personnalisés

Afin d'éviter que les fichiers modifiés soient écrasés à chaque exécution du script, j'ai modifié son comportement pour :

- Créer manuellement un dossier temporaire (tmp/taxhub/) ;
- Y stocker les fichiers ZIP et SQL corrigés ;
- Les utiliser dans l'installation ;
- Et enfin supprimer ce répertoire en fin de traitement, comme prévu dans le script original.

```
if $install_taxonomie
        then
        mkdir -p /tmp/taxhub
        cp ~/atlas/data_inpn_taxhub.sql /tmp/taxhub/
        cp ~/atlas/*.zip /tmp/taxhub/

        array=( TAXREF_v18_2025.zip BDC-STATUTS-v18.zip ESPECES_REGLEMENTEES_v11.zip
R_FRANCE_20160000.zip )
        for i in "${array[@]}"
                do
        echo "Unzipping $i..."

        unzip -o /tmp/taxhub/$i -d /tmp/taxhub
        done
        # Nettoyage du fichier TAXREF v18 (suppression des colonnes CD_BA et URL_INPN)
        echo "▶ Nettoyage du fichier TAXREFv18..."
        TAXREF_ORIG="/tmp/taxhub/taxrefv18.txt"
        TAXREF_CLEAN="/tmp/taxhub/taxrefv18.txt"

        cut -f1-14,16-42 "$TAXREF_ORIG" > "$TAXREF_CLEAN"
        echo "✅ Fichier nettoyé généré : $TAXREF_CLEAN"
```

Figure 5-2-Extrait du fichier data_taxhub_inpn au niveau du téléchargement des éléments nécessaires pour la base

5.4 Résolution des problèmes liés aux fichiers TaxHub

Lors de l'exécution du script, des erreurs sont survenues concernant le téléchargement des fichiers nécessaires au module TaxHub (taxhubdata.sql, taxhubdb.sql, etc.). Cela était dû à une **variable release non définie**, empêchant le script de localiser la bonne version dans le dépôt GitHub. Il a donc fallu le remplacer par la dernière version de GeoNature en liste : 1.8.1.

```
        echo "Getting 'taxonomie' schema creation scripts..."
        wget https://raw.githubusercontent.com/PnX-SI/TaxHub/1.8.1/data/taxhubdb.sql -P /tmp/taxhub
        wget https://raw.githubusercontent.com/PnX-SI/TaxHub/1.8.1/data/taxhubdata.sql -P /tmp/-
taxhub
```

Figure 5-3-Extrait du fichier data_taxhub_inpn

En attendant une solution pérenne, j'ai renseigné manuellement la **version exacte de GeoNature Atlas** utilisée, ce qui a permis de récupérer les fichiers nécessaires et de poursuivre l'installation.

5.5 Création et ajustement des structures SQL

5.5.1 Tables manquantes et exécution manuelle

Certaines tables importantes n'étaient pas générées automatiquement lors de l'installation, notamment la table vm_cor_taxon_organism, essentielle pour faire le lien entre les taxons et les organismes. J'ai donc récupéré et exécuté manuellement les scripts SQL correspondants à l'aide de **pgAdmin**.

5.5.2 Modification du script without_geonature.sql

Le script without_geonature.sql, utilisé pour créer la table syntheseff, a nécessité plusieurs ajustements pour être compatible avec notre base GENS :

- Le champ observateurs était initialement en VARCHAR (255) ; or, nos données peuvent contenir plusieurs noms concaténés, dépassant cette limite. Je l'ai donc remplacé par le type TEXT, plus souple.

5.6 Personnalisation des données territoriales

Par défaut, les fichiers SQL d'Atlas font référence au **territoire du Parc national des Écrins**. En collaboration avec ma tutrice de stage, nous avons adapté ces fichiers pour y intégrer les **géométries de nos sites (comme étant des communes) et leurs identifiants propres,** en remplacement des codes INSEE classiques. Cette modification garantit la compatibilité avec la logique frontend de l'Atlas (pages espèces par commune).

5.7 Import et adaptation des médias naturalistes

Pour enrichir les fiches taxons avec des **illustrations et contenus multimédias**, j'ai adapté le script run_import.py ainsi que certaines fonctions de functions.py, issus du dépôt TaxHub.

5.7.1 Nettoyage préalable de la table bib_noms

Le téléchargement des médias était bloqué tant que la table bib_noms contenait encore des données héritées de TAXREF v14. J'ai donc remplacé son contenu pour qu'il corresponde à la version v18.

5.7.2 Gestion des droits d'accès

Pour que le processus d'import fonctionne correctement, j'ai attribué les **droits de lecture/écriture** à l'utilisateur geonatatlas et au propriétaire geonatadmin sur la table t_media.

5.7.3 Liaison avec les observations

Enfin, j'ai adapté la logique d'import pour faire correspondre le champ cd_nom de la table syntheseff avec le champ cd_ref de t_media, assurant ainsi une **liaison cohérente entre les observations et les médias illustrant les taxons observés**.

```python
try:
    conn = psycopg2.connect(SQLALCHEMY_DATABASE_URI)
except Exception as e:
    print("Connexion à la base impossible")

try:
    cur = conn.cursor()
    sql = """
        SELECT DISTINCT s.cd_nom
        FROM synthese.syntheseff s
        LEFT JOIN taxonomie.bib_noms n ON s.cd_nom = n.cd_nom
        LEFT JOIN taxonomie.t_medias m ON n.cd_ref = m.cd_ref
        WHERE m.id_media IS NULL
    """
    cur.execute(sql)
    rows = cur.fetchall()
except Exception as e:
    print("❌ Problème lors de la récupération de la liste des cd_nom :", e)
    conn.close()
    exit(1)

# Vérification pour éviter l'appel à main() si rows est vide ou non défini
if not rows:
    print("ℹ️ Aucun taxon à traiter.")
else:
    main(conn, rows, WD_MEDIA_PROP, TAXHUB_MEDIA_ID_TYPE, False, False)
```

Figure 5-4-code de récupération des médias

5.8 Liaison avec la base GENS via Foreign Data Wrapper (FDW)

Afin de permettre à GeoNature Atlas d'accéder directement aux données issues de la base GENS (hébergée sur une autre machine du réseau), j'ai mis en place une connexion inter-base via l'extension **Foreign Data Wrapper** (FDW), spécifiquement l'extension postgres_fdw fournie par PostgreSQL.

5.8.1 Activation de l'extension et déclaration du serveur distant

La première étape consistait à activer l'extension postgres_fdw sur la base GeoNature Atlas, puis à déclarer un **serveur distant** pointant vers la machine hébergeant la base GENS :

```sql
-- Extension FDW
CREATE EXTENSION IF NOT EXISTS postgres_fdw;

-- Serveur FDW
CREATE SERVER geonaturedbserver
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host '192.168.1.163', dbname 'gens', port '5432');
```

Figure 5-5-code de création de la FDW

5.8.2 Mappage des utilisateurs

Afin d'autoriser l'accès aux données distantes, j'ai défini deux utilisateurs mappés pour ce serveur distant :

- **geonatadmin : utilisateur principal de la base Atlas.**
- **postgres : compte administrateur de la base GENS.**

```
-- Utilisateur de connexion
CREATE USER MAPPING FOR geonatadmin
SERVER geonaturedbserver
OPTIONS (user 'geonatadmin', password
OPTIONS (user 'geonatatlas', password

CREATE USER MAPPING FOR postgres
SERVER geonaturedbserver
OPTIONS (user 'postgres', password 'r
```

Figure 5-6-code de création des utilisateurs de la FDW

5.8.3 Import temporaire des schémas distants

Les données des schémas gn_meta, utilisateurs et synthese ont été importées temporairement via FDW, dans des schémas de travail distincts (_tmp), afin de pouvoir effectuer un transfert local propre vers la base de l'Atlas :

5.8.3.1 Transfert du schéma gn_meta

```
---Remplissage de gn_meta

create schema gn_meta_tmp;
import foreign schema gn_meta from server geonaturedbserver into gn_meta_tmp;
INSERT INTO gn_meta.cor_dataset_actor(
    id_cda, id_dataset, id_role, id_organism)
select * from gn_meta_tmp.cor_dataset_actor;
DROP SCHEMA gn_meta_tmp CASCADE;
```

Figure 5-7-code de création du schéma gn_meta

5.8.3.2 Transfert du schéma utilisateurs

```
---Remplissage de utilisateurs

create schema utilisateurs_tmp;
import foreign schema utilisateurs from server geonaturedbserver into utilisateurs_tmp;
INSERT INTO utilisateurs.bib_organismes(
    id_organisme, uuid_organisme, nom_organisme, adresse_organisme,
    cp_organisme, ville_organisme,
    tel_organisme, email_organisme, url_organisme, url_logo)
select * from utilisateurs_tmp.bib_organismes;
DROP SCHEMA utilisateurs_tmp CASCADE;
```

Figure 5-8-code de création du schéma utilisateurs

5.8.3.3 Transfert du schéma synthese

```
---Remplissage de synthese

create schema synthese_tmp;
import foreign schema synthese from server geonaturedbserver into synthese_tmp;
INSERT INTO synthese.syntheseff(
    id_synthese, id_organism, id_dataset, cd_nom, insee, dateobs,
    observateurs, altitude_retenue, the_geom_point, effectif_total,
    diffusion_level, supprime)
select * from synthese_tmp.syntheseff;
DROP SCHEMA synthese_tmp CASCADE;
```

Cette méthode permet un **transfert sécurisé, contrôlé et ponctuel** des données nécessaires à l'Atlas, tout en préservant l'intégrité de la base source (GENS).

5.8.4 Actualisation des vues matérialisées

Une fois les données importées, j'ai procédé à l'actualisation des **vues matérialisées** utilisées par GeoNature Atlas pour la diffusion des données sur l'interface (observations, taxons, altitudes, etc.). Cette étape garantit que les données consultables dans l'interface utilisateur de l'Atlas sont à jour, cohérentes et exploitables immédiatement.

6 Configuration du serveur Apache2 et finalisation de l'affichage

Comparée aux étapes liées à la base de données, la configuration du serveur web **Apache2** a été relativement simple à mettre en œuvre.

6.1 Création du fichier de configuration

J'ai commencé par créer le fichier de configuration du site Apache, intitulé **atlas.conf**, en suivant les indications de la documentation officielle de GeoNature Atlas. Ce fichier contient les directives nécessaires pour que le serveur sache où chercher les fichiers de l'application et comment les interpréter.

6.2 Personnalisation du fichier config.py

Une fois le fichier atlas.conf en place, j'ai procédé à quelques modifications dans le fichier config.py de l'application, afin d'adapter le comportement de l'Atlas au contexte local d'Eden 62 :

- **Nom de l'application** : renommé pour refléter le projet (Eden62).
- **Racine d'accès à l'application** : le paramètre url_application a été remplacé par **application_root = '/'**, car l'application devait être **accessible directement à la racine du serveur** (et non via un sous-répertoire ou un nom de domaine spécifique, cette installation étant à but interne et non définitive).
- **Nom du serveur** : j'ai défini **ServerName atlas.local** dans le fichier Apache pour faciliter l'accès en local.

6.3 Lancement et test du serveur

Après ces ajustements, j'ai démarré l'application via le script de lancement, puis activé le **service Apache2** pour qu'il serve l'Atlas. Un test dans le navigateur a permis de valider que l'interface était bien accessible via l'URL http://atlas.local.

Ces étapes ont permis de finaliser l'affichage et de rendre l'application GeoNature Atlas pleinement fonctionnelle dans l'environnement local prévu pour le projet.

```
<VirtualHost *:80>
    ServerName atlas.local

    <Location />
        ProxyPass        http://127.0.0.1:8080/
        ProxyPassReverse http://127.0.0.1:8080/
    </Location>

    ErrorLog ${APACHE_LOG_DIR}/atlas_error.log
    CustomLog ${APACHE_LOG_DIR}/atlas_access.log combined
</VirtualHost>
```

Figure 6-1-Création du fichier atlas.conf

7 Interface de l'application

Une fois la base de données configurée et les dépendances installées, l'application GeoNature Atlas a pu être lancée. Plusieurs **erreurs d'exécution et de rendu** sont apparues à ce stade, nécessitant des ajustements spécifiques pour garantir un affichage conforme à nos données et à notre territoire.

7.1 Résolution d'un bug sur les fiches espèces

Lors de l'affichage des fiches espèces, un **bug empêchait le chargement correct des données**. Après analyse, ce problème provenait d'une **incohérence de type de données** sur le champ insee :

- Typé en **integer** dans la vue vm_communes
- Typé en **text** dans la vue vm_observations

Pour corriger cette incompatibilité, j'ai modifié les fichiers Python **tCommunesRepository.py** et **vmCommunesRepository.py** afin d'uniformiser les types de données (insee en text) dans l'ensemble de l'application.

7.2 Personnalisation de la carte et du zoom

Par défaut, GeoNature Atlas est centré sur le territoire du Parc national des Écrins. J'ai modifié le fichier **config_schema.py** afin d'ajuster le **niveau de zoom** et le **centre de la carte** sur le département du **Pas-de-Calais**, correspondant au périmètre d'Eden 62.

7.3 Nettoyage des observateurs affichés

Les données issues de GENS contiennent une **liste d'observateurs** pour chaque observation. Lors de l'affichage, un problème est apparu : plusieurs noms se répétaient dans différentes listes, ce qui faussait le **nombre d'observateurs affichés** par rapport au total réel.

Plutôt que de modifier en profondeur le code source de l'application, j'ai choisi une solution modulaire :

- Création d'un fichier **filters.py** *(cf. Annexe-6 : Code fichier filters.py)*, chargé d'extraire les noms depuis les listes d'observateurs.
- Ce script filtre les doublons et reconstitue une **liste propre et unique**.
- J'ai ensuite intégré cette fonction dans **app.py**, le fichier principal de l'application, afin d'appliquer ce filtre lors de l'affichage des fiches.

Ce correctif a permis d'améliorer la lisibilité et la cohérence des données affichées à l'utilisateur, sans perturber le fonctionnement global de l'application.

8 Personnalisation de l'interface utilisateur

La personnalisation de l'interface GeoNature Atlas a été réalisée afin d'adapter l'apparence et l'organisation des contenus aux besoins spécifiques d'Eden 62, tout en respectant sa charte graphique et son territoire.

8.1 Modifications graphiques (CSS et JavaScript)

J'ai d'abord modifié le fichier **customs** pour adapter :

- Les **couleurs principales** de l'interface (fonds, boutons, liens) ;
- Les **polices de caractères** utilisées dans les titres et les paragraphes ;
- Certains aspects de l'apparence générale (marges, alignements, icônes...).

Le fichier **map-custom.js** *(cf. Annexe-12 : maps-custom.js)*, chargé de gérer le rendu cartographique, a également été adapté. En particulier, j'ai modifié le **code couleur des couches cartographiques** pour refléter les différentes localisations du point pouvant être approximative ou précise.

8.2 Réorganisation de la page d'accueil

Dans le répertoire home, le fichier principal main.html structure les différentes sections de la page d'accueil. J'ai modifié **l'ordre d'affichage** des blocs pour prioriser les contenus les plus pertinents :

1. **Bloc d'introduction** (également retravaillé)
2. **Statistiques globales** des observations
3. **Carte interactive** des observations des 30 derniers jours
4. Rubrique **"À voir en ce moment"**
5. **Nouvelle espèce observée**
6. **Espèce à découvrir**
7. Section **Partenaires**

Cette réorganisation a permis de rendre l'interface plus claire et orientée vers la valorisation active des données.

8.3 Ajustements des composants HTML

D'autres fichiers ont été adaptés pour affiner la présentation globale :

- **footer.html** : personnalisation des mentions et des liens de bas de page.
- **navbar.html** *(cf. Annexe-8 : navbar.html)* : modification des éléments du menu de navigation.
- **presentation.html** : réécriture de l'introduction et ajout de contenus spécifiques à Eden 62.

8.4 Modifications spécifiques aux fiches espèces

Sur les fiches espèces, j'ai effectué plusieurs ajustements :

- Dans **charts.html** *(cf.Annexe-9 : Code chart.html)*, j'ai **supprimé l'histogramme des altitudes**, jugé peu pertinent pour un territoire majoritairement plat comme le Pas-de-Calais.

- Dans **charts.html** *(cf.Annexe-9 : Code chart.html)*, **charts.js** *(cf.Annexe-10 : Code chart.js)* et **atlasRoute.py**, j'ai **rajouté un histogramme des sites** après avoir créé un script python adapté (**vmSiteRepository.py,** *cf. Annexe-11 : Code vmSiteRepository.py*), jugé pertinent pour permettre de savoir dans quels sites l'espèce recherchée est le plus observée, cet histogramme ne prend en compte que les 5 premiers ayant le plus d'observations

J'ai également **adapté l'affichage des observateurs**, en intégrant la fonction de nettoyage développée dans **filters.py** *(cf. Annexe-6 : Code fichier filters.py)*, afin d'éviter les doublons et de garantir une meilleure lisibilité des contributeurs.

- Ensuite j'ai modifié le fichier tabTaxons.html qui contient aussi les observateurs pour avoir le bon nombre d'observateurs.

9 Présentation de l'application

Dans cette partie, les différentes fonctionnalités de l'atlas seront présentées, en spécifiant les développements effectués pour chacune d'entre elles.

9.1 La page d'accueil – Portail de valorisation des observations

La page d'accueil de GeoNature Atlas constitue le point d'entrée principal vers les données naturalistes d'Eden 62. Elle a été pensée comme un portail dynamique et pédagogique, offrant une vue synthétique de l'activité d'observation sur le territoire.

Conformément aux objectifs du projet, cette page met en avant trois types d'accès complémentaires aux données :

- Une entrée grand public et géographique, centrée sur une carte interactive affichant les observations des 30 derniers jours. Cette carte permet également une recherche ciblée par site, facilitant l'exploration locale des espèces.

- Une entrée taxonomique, permettant la recherche par nom scientifique ou vernaculaire. L'utilisateur peut ainsi retrouver une espèce précise à partir de son nom, avec un accès direct à sa fiche descriptive.

- Une entrée visuelle, via une galerie photo dynamique présentant les espèces les plus observées récemment. Ce composant vise à attirer l'attention de l'utilisateur en valorisant les espèces emblématiques du moment, avec leurs images et noms directement consultables.

Cette structuration permet de toucher à la fois un public non spécialiste, curieux de découvrir la biodiversité autour de chez lui, et des utilisateurs plus experts recherchant une espèce ou un taxon en particulier. Elle reflète également les missions d'Eden 62 en matière de sensibilisation, de gestion écologique et de diffusion des connaissances.

9.1.1 Bloc de présentation

Ce bloc affiche un paragraphe de présentation retraçant la mission de l'outil (valorisation des données naturalistes, sensibilisation du grand public, transparence des actions) et une photographie d'une espèce présente sur le territoire vient illustrer ce propos et ancrer immédiatement l'utilisateur dans le territoire couvert.

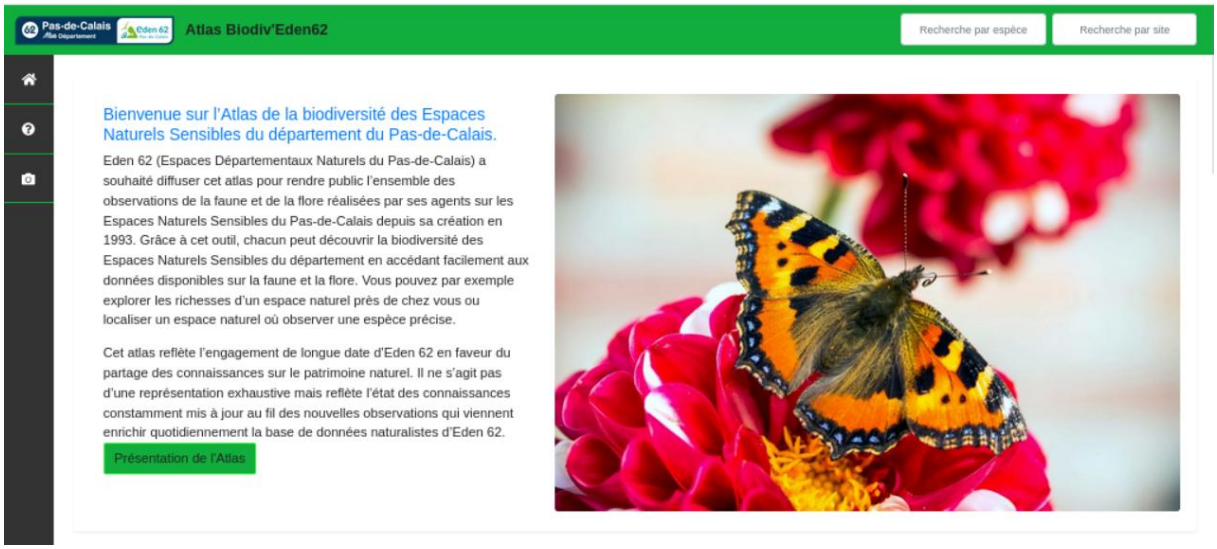


Figure 9-1- Page d'accueil : Introduction de l'Atlas

9.1.2 Bloc "Partenaires"

Cette section présente les logos (ou noms) des adhérents au Syndicat Mixte (collectivités, organismes) qui contribuent à la structure et à son fonctionnement.

Figure 9-2-Page d'accueil : Bandeau logo

9.1.3 Bloc "Statistiques globales"

On retrouve dans ce bloc un tableau de bord de l'atlas contenant les différentes statistiques de la base de données notamment le nombre d'observations, le nombre d'espèces présentes, le nombre de sites et le nombre de médias présents dans la galerie photo. Dans ce bloc on retrouve aussi les deux entrées de recherche par site et par espèce présent plus haut dans l'atlas.

Figure 9-3-Page d'accueil : Statistique de la base de données de l'atlas

9.1.4 Bloc “Carte interactive”

La carte géolocalise en temps réel les observations des 30 derniers jours. Elle permet de zoomer et de filtrer par site, zone ou période. Un clic sur un point ouvre la fiche détaillée de l’observation correspondante. Pour l’instant il n’y a rien sur l’image car les dernières données en date n’ont pas été mise à jour, mais normalement dans le coin droit de la carte s’affiche la liste des dernières observations des 30 derniers jours.

Figure 9-4-Page d'accueil : Carte des dernières observations datant des 30 derniers jours

9.1.5 Bloc “À voir en ce moment”

Ce carrousel ou liste dynamique propose une sélection d’espèces. Il oriente l’utilisateur vers les espèces plus observées durant la saison en cours.

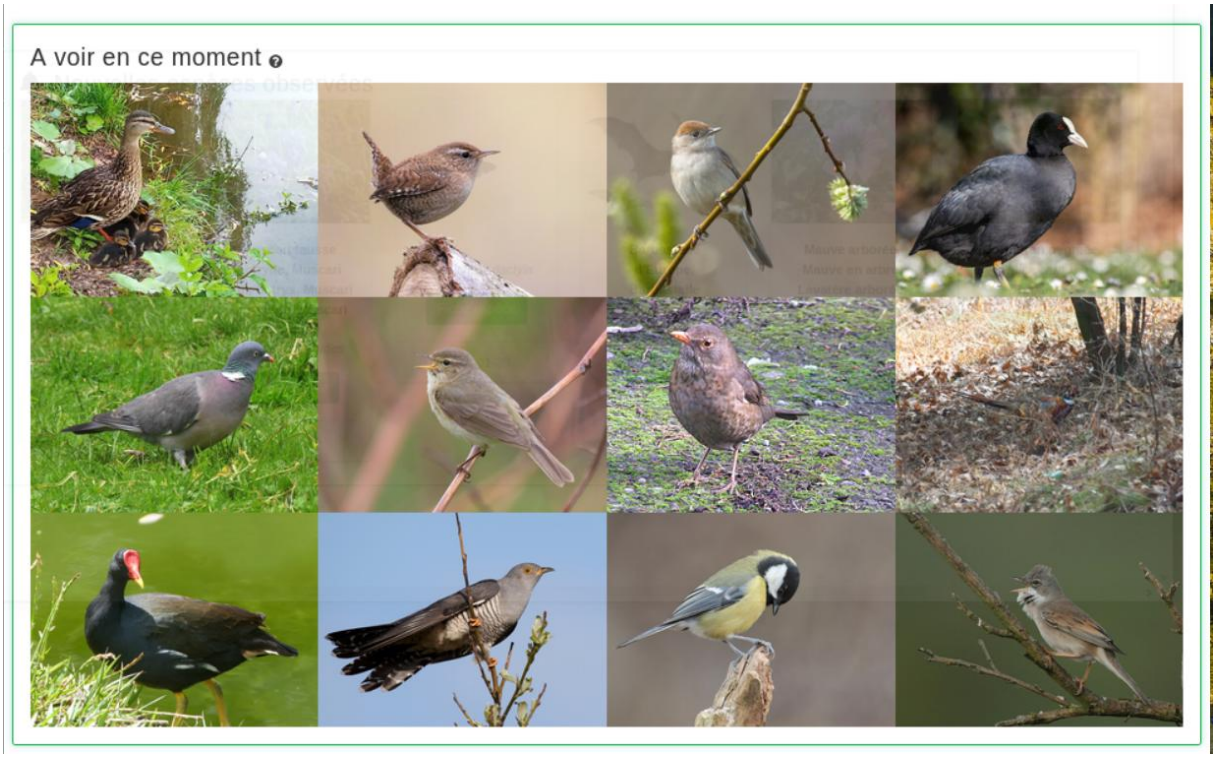


Figure 9-5-Page d'accueil : À voir en ce moment

9.1.6 Bloc “Nouvelle espèce observée”

Chaque jour, une espèce nouvellement enregistrée dans la base est mise en lumière : photo, nom scientifique et vernaculaire, date et lieu de la première observation. L’objectif est de susciter la curiosité et de montrer l’évolution continue de la biodiversité locale.

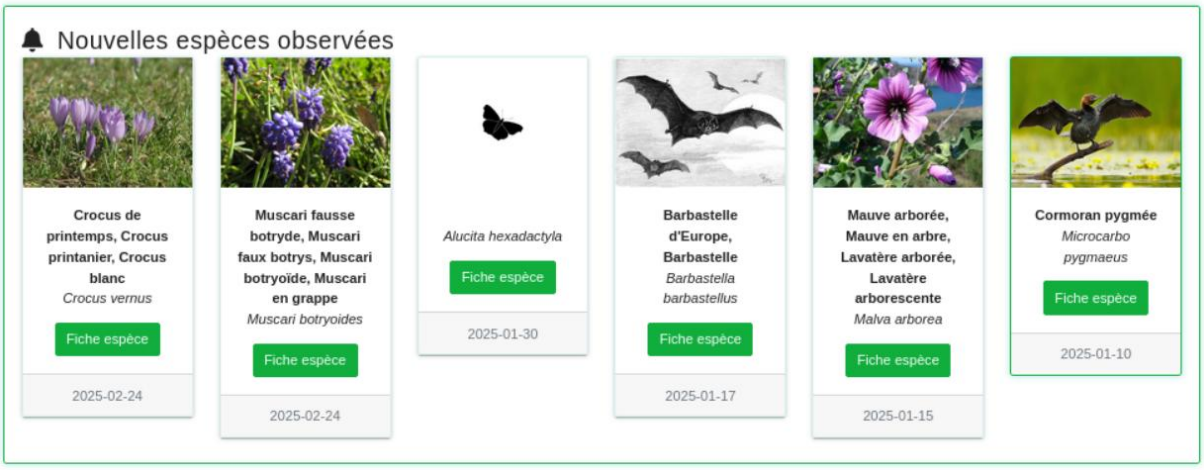


Figure 9-6-Page d'accueil : Nouvelle espèce observée

9.1.7 Bloc “Espèce à découvrir”

Ce bloc présente des espèces encore peu observées sur le territoire, avec une courte description. Il encourage le grand public à découvrir des espèces récemment enregistrées dans la base.

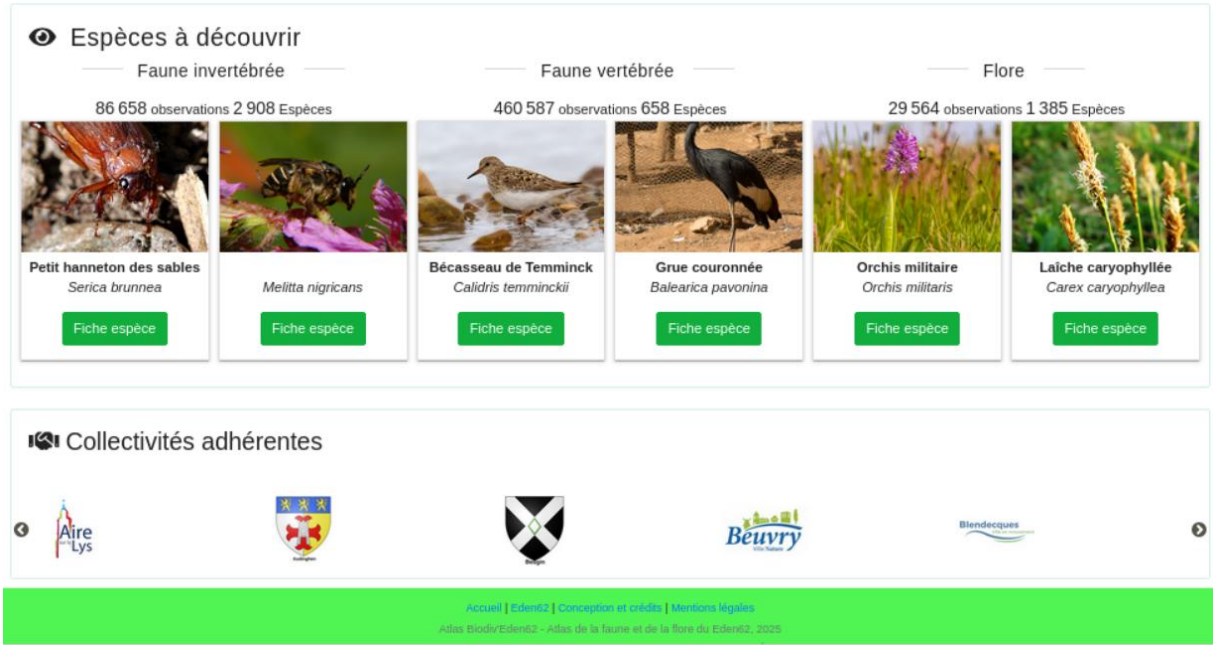


Figure 9-7-Page d'accueil : Espèce à découvrir avec bandeau logo des collectivités adhérentes

9.2 La fiche espèce – Une visualisation complète et interactive des données

Chaque fiche espèce dans GeoNature Atlas présente une interface structurée en **trois blocs principaux**, permettant d'explorer de manière détaillée les données relatives à une espèce donnée sur le territoire géré par Eden 62.

9.2.1 Cartographie des observations

La première section affiche une **carte interactive** représentant les différentes localisations où l'espèce a été observée. Les données sont présentées selon un **maillage spatial**, et chaque maille est associée à un **centroïde coloré** dont la teinte varie en fonction de **la localisation du point (approximative ou précise)**.

Lorsqu'un utilisateur survole ou clique sur une maille, une **info-bulle** s'affiche contenant le nombre d'observations et l'année de leur observation tandis que lorsqu'on survole les points après un zoom les principales informations de l'observation s'affichent : noms des observateurs, date, effectif observé et altitude. La carte permet également :

- Le **zoom dynamique** sur le territoire,
- Le **changement de fond cartographique** (OpenStreetMap, satellite, etc.),
- L'utilisation d'un **baromètre temporel** (curseur) permettant de filtrer les observations par année.

En complément, une section placée sous la carte répertorie les **noms des sites** où l'espèce a été recensée, ainsi que les **observateurs associés**, accompagnés de leur nombre.

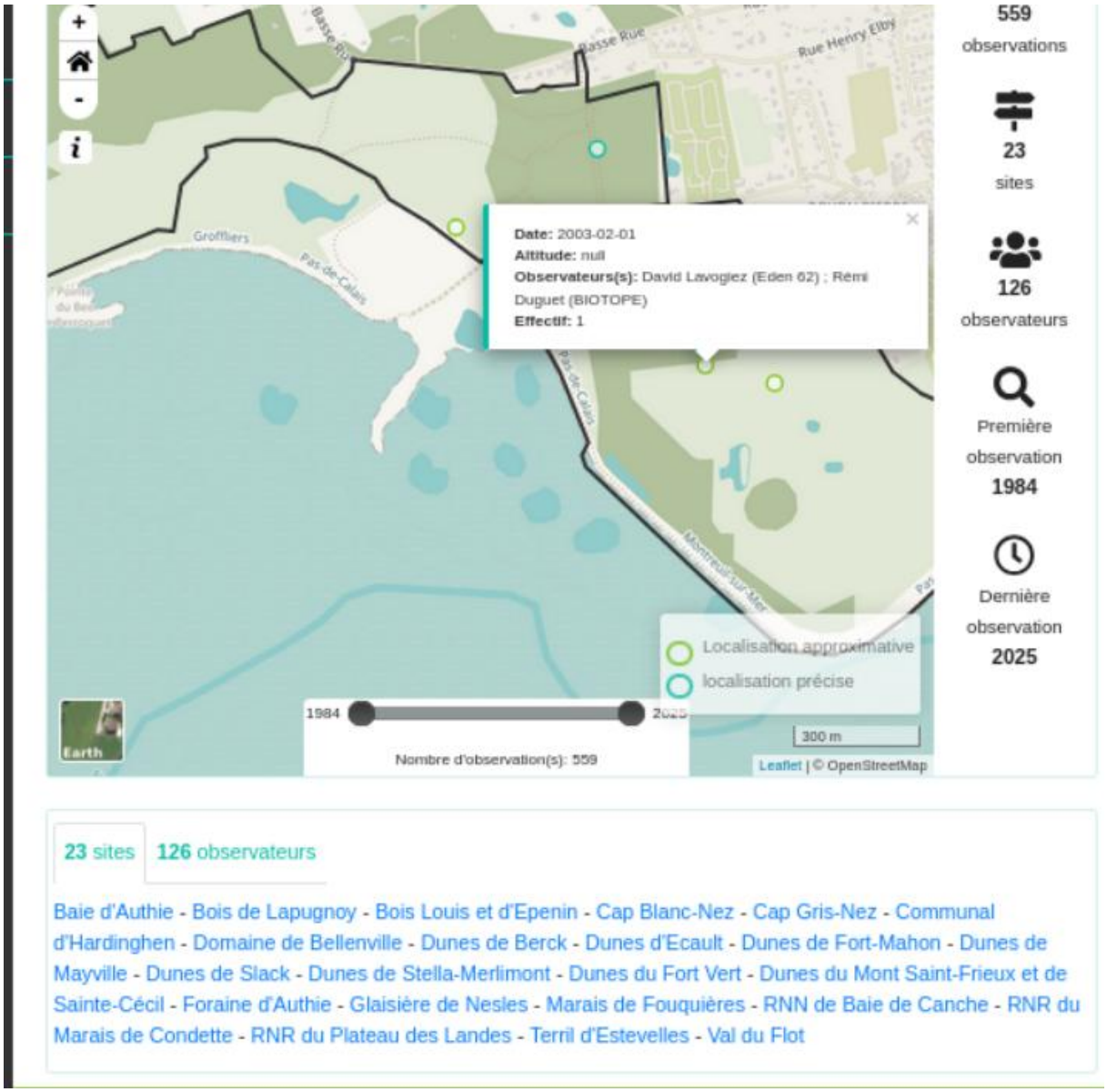


Figure 9-8-Partie carte de la fiche espèce

9.2.2 Informations naturalistes

Le second bloc regroupe les **informations descriptives** de l'espèce :

- Une **description générale**,
- Le **milieu naturel** dans lequel elle est observée,
- Sa **répartition sur le territoire**,

- Ses **synonymes taxonomiques** éventuels.

Cette section comprend également un **histogramme mensuel** affichant la distribution des observations au fil de l'année, permettant de visualiser la **phénologie** de l'espèce (saisonnalité).

9.2.3 Galerie et classification

Le dernier bloc présente une **galerie illustrée** de l'espèce, avec une photographie, son **nom scientifique (latin)** et son **nom vernaculaire (français)**. Les principales **caractéristiques taxonomiques** sont également indiquées : classe, ordre, famille, sous-famille, tribu et genre.

Chacune de ces catégories est **cliquable** et redirige vers la page correspondante listant **toutes les espèces appartenant à ce groupe taxonomique**, facilitant ainsi la navigation dans le référentiel.

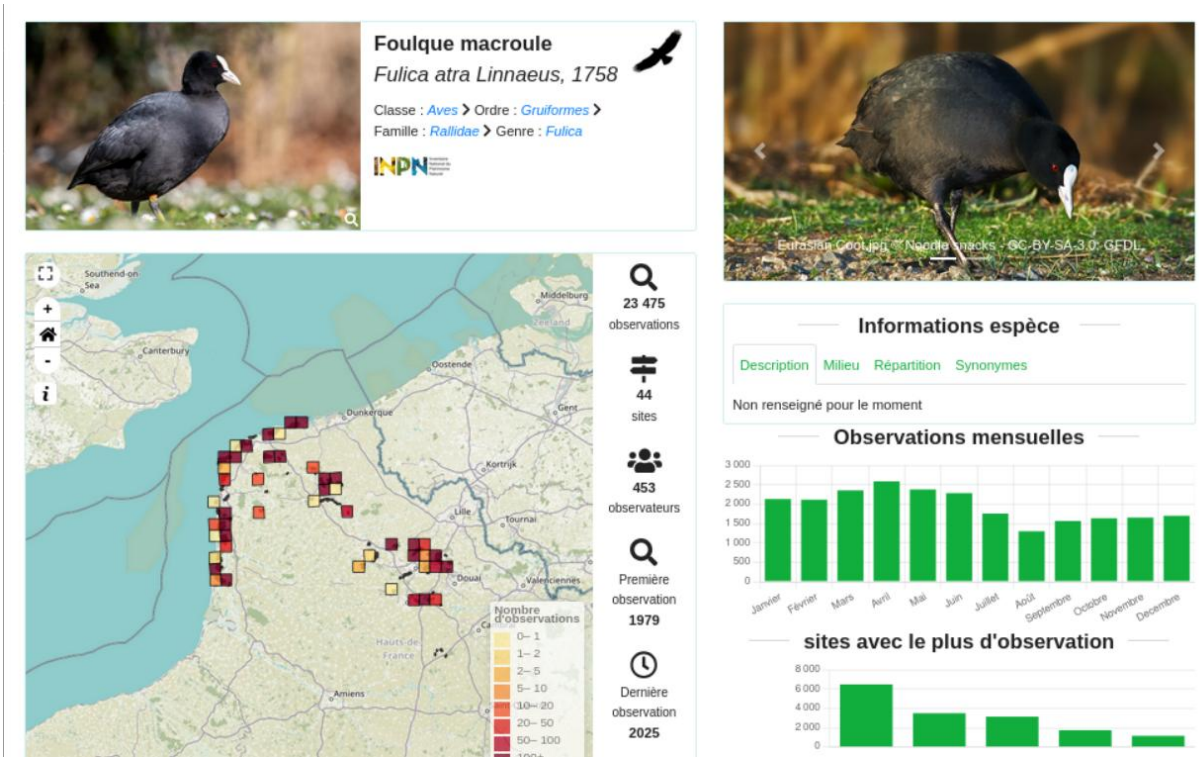


Figure 9-9-Fiche Espèce complète de l'Atlas

9.3 La fiche site – Une synthèse des observations à l'échelle territoriale

La fiche associée à un **site** permet de consulter de manière centralisée **toutes les observations naturalistes** réalisées sur ce territoire administratif.

Elle se compose de plusieurs sections :

- **Carte interactive** : elle localise précisément le site et affiche les **100 dernières espèces observées sur son périmètre**. Des filtres permettent de restreindre les résultats par groupes

taxonomiques ou par périodes.

- **Liste des espèces recensées** : une table dynamique recense toutes les espèces observées dans le site, avec leur nom scientifique, nom vernaculaire, et un lien vers leur fiche dédiée.

- **Statistiques** : un encadré présente des indicateurs synthétiques comme le nombre total d'observations, le nombre d'espèces différentes, et les périodes d'observation les plus fréquentes.

Cette fiche permet ainsi d'obtenir un **portrait écologique local**, utile pour les élus, les gestionnaires de site, ou les citoyens souhaitant découvrir la biodiversité de leur site.

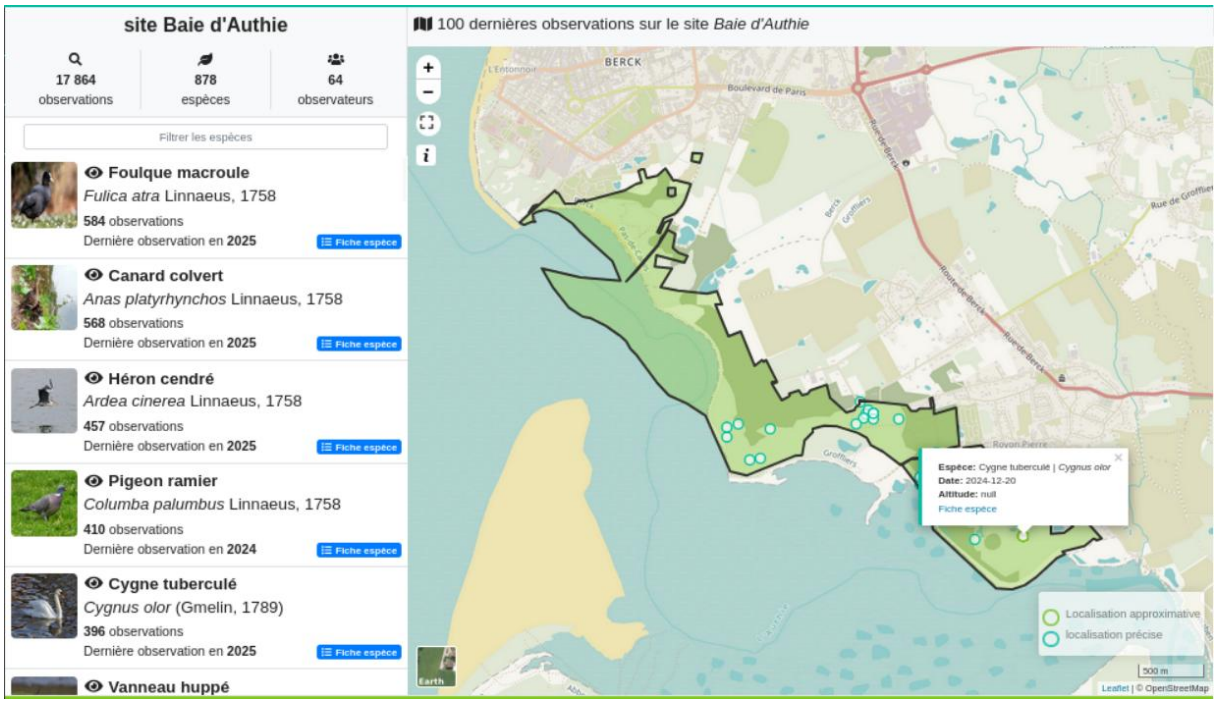


Figure 9-10-Fiche site de l'Atlas

9.4 La fiche taxon – Une vue hiérarchique et filtrée du référentiel

La **fiche taxon** offre une vue regroupant **tous les taxons** appartenant à une **même catégorie taxonomique**, qu'il s'agisse d'un **règne, d'une classe, d'un ordre, d'une famille, d'un genre ou d'une sous-famille**.

Elle se structure comme suit :

- **Titre hiérarchique** : affichage du nom de la catégorie (ex. : *Lepidoptera*), accompagné d'un rappel de sa position dans la classification.

- **Liste filtrable des espèces associées** : un tableau présente les espèces rattachées à ce taxon, avec pour chacune :

- Une image,
- Le nom scientifique,
- Le nom vernaculaire,
- Un lien vers la fiche espèce.

Ce type de fiche permet d'explorer le **référentiel taxonomique de manière structurée**, en facilitant la navigation au sein de la biodiversité locale.

Figure 9-11-Fiche Taxon de l'Atlas

9.5 La galerie photo – Valorisation visuelle de la biodiversité

La **galerie photo** constitue un espace dédié à la **valorisation visuelle des espèces observées** sur le territoire d'Eden 62. Elle regroupe l'ensemble des **médias naturalistes** associés aux taxons présents dans la base GENS, en particulier les photographies importées et liées aux observations.

Fonctionnalités principales :

- **Filtrage par groupe taxonomique** :
  L'utilisateur peut filtrer les espèces affichées en sélectionnant un **groupe d'animaux** représenté par une **icône illustrée** (mammifères, oiseaux, insectes, amphibiens, etc.). Ce système visuel rend la navigation intuitive, même pour un public non spécialiste.

- **Tri dynamique des espèces** :
  La galerie propose un **tri des espèces illustrées** selon leur **fréquence d'observation**. Cela

permet de faire ressortir les espèces les plus communes ou emblématiques du moment.

- **Recherche par nom** :
  Une **barre de recherche** permet de retrouver rapidement une espèce en saisissant son **nom scientifique ou vernaculaire**. L'accès à la fiche espèce est immédiat depuis la galerie.

Cette fonctionnalité enrichit l'expérience utilisateur en **favorisant une découverte visuelle** de la biodiversité locale. Elle répond également à un objectif pédagogique et de sensibilisation, en rendant les données naturalistes **plus accessibles, attractives et engageantes**.

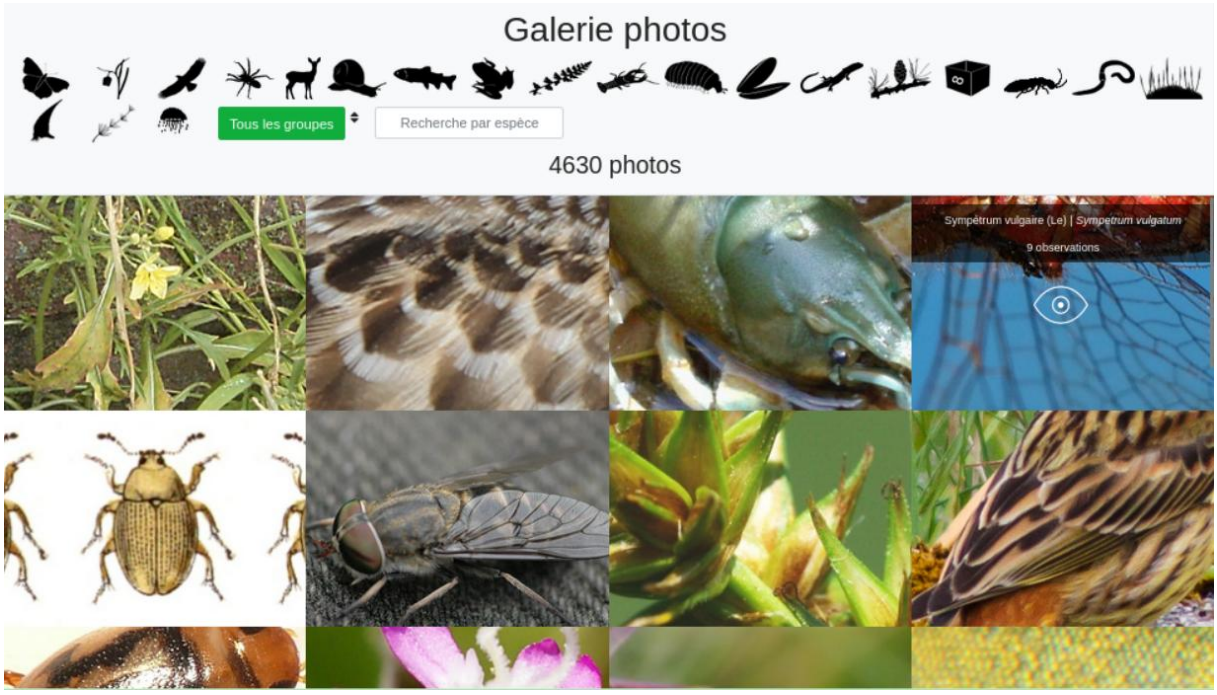


Figure 9-12-Galerie photo de l'Atlas

10 Conclusion et perspectives

Ce stage m'a offert l'opportunité de m'immerger dans un projet concret, mêlant des compétences en **développement web**, **administration de bases de données**, **interopérabilité entre systèmes** et **visualisation de données naturalistes**.

L'objectif principal consistait à **déployer et adapter l'application GeoNature Atlas** afin de valoriser les données issues de la base interne **GENS**, développée par Eden 62. Plusieurs étapes techniques ont été nécessaires : la configuration d'un environnement Linux dédié, la personnalisation du socle applicatif, l'adaptation de la base de données (notamment avec l'intégration de **TAXREF v18**), ainsi que la mise en œuvre d'un **lien dynamique entre bases via un Foreign Data Wrapper (FDW)**.

J'ai également travaillé sur la **personnalisation de l'interface utilisateur**, la correction de **bugs liés à la structure des données**, l'adaptation des vues, ainsi que l'intégration de **contenus multimédias** depuis Wikidata. Ce travail a permis de produire une version fonctionnelle et ergonomique de l'Atlas, cohérente avec les objectifs de **transparence**, de **sensibilisation** et de **valorisation de la biodiversité** poursuivis par Eden 62.

Au-delà des aspects techniques, ce projet m'a permis d'approfondir ma compréhension des **enjeux liés à la gestion et à la standardisation des données environnementales**, ainsi que de renforcer mes compétences en **SQL, Python, architecture web, SIG et interconnexion de systèmes**. Il s'agit d'une expérience formatrice et enrichissante qui m'a conforté dans mon intérêt pour les projets croisant **open data, écologie et technologies de l'information**.

10.1 Perspectives

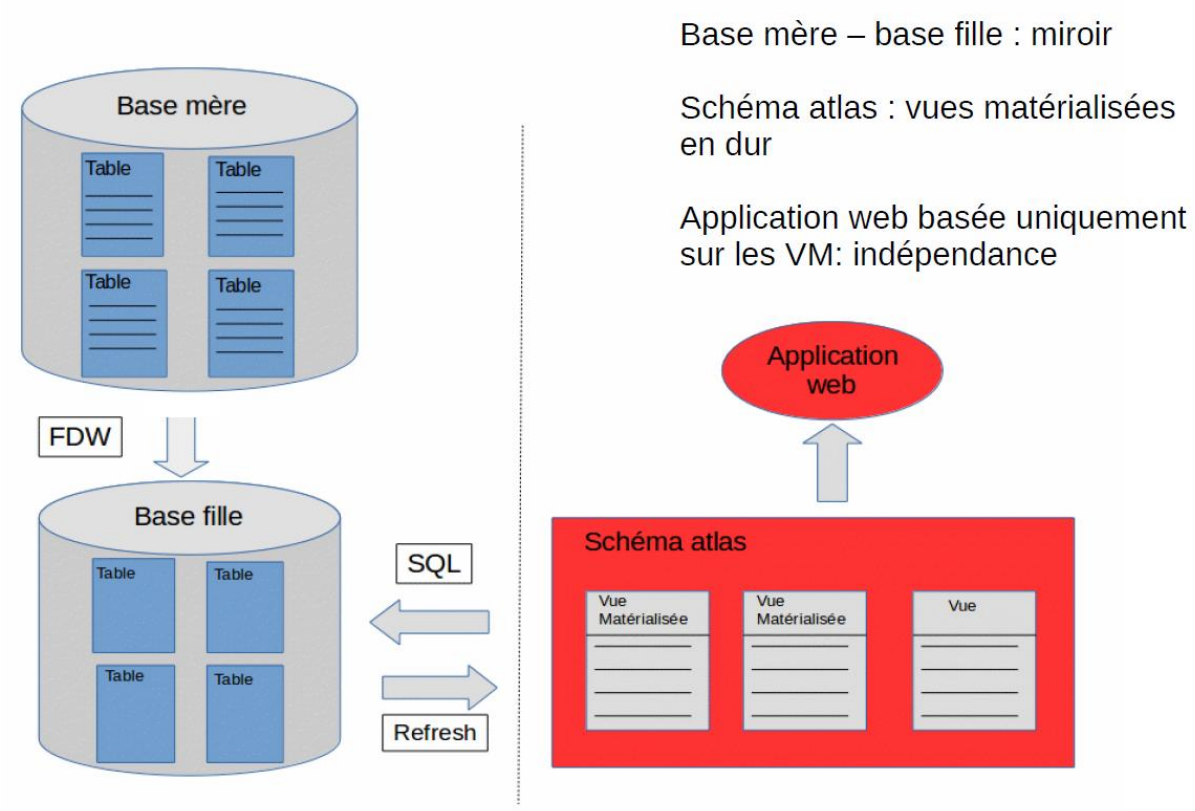
À l'issue de la période de stage évaluée, le GeoNature Atlas est **déployé en local** sur une machine virtuelle dédiée, à des fins de démonstration et de test. Cependant, des **perspectives concrètes de mise en production** ont déjà été envisagées pour la suite du projet :

- **L'acquisition d'un serveur physique Linux** spécifiquement dédié à l'hébergement de l'Atlas,
- **L'achat d'un nom de domaine personnalisé**, permettant un **accès public à l'application via Internet**,
- Et, à terme, **la mise en ligne officielle de l'Atlas d'Eden 62**, à destination des citoyens, des collectivités institutionnels et scientifiques.

Cette future mise en production assurera une diffusion élargie des données naturalistes, en phase avec la mission de sensibilisation et de transparence environnementale portée par Eden 62.

11 Annexes

11.1 Annexe-1 : schéma base fille (fourni par Geonature)

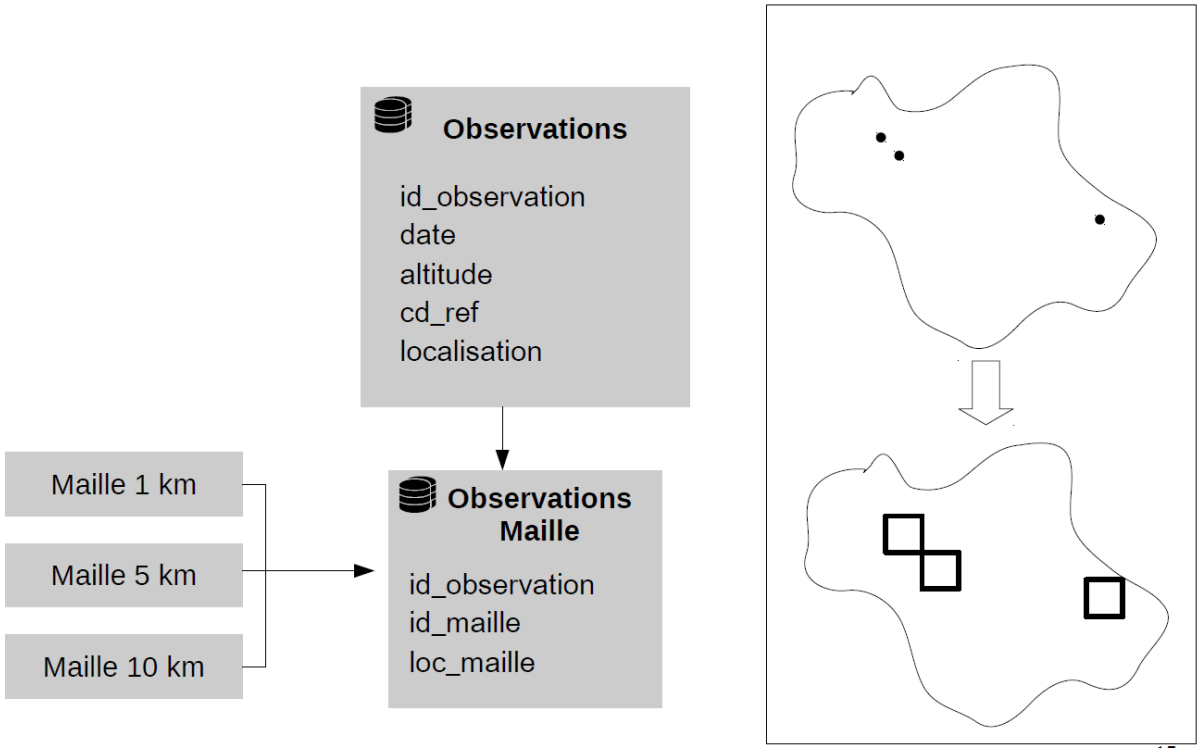


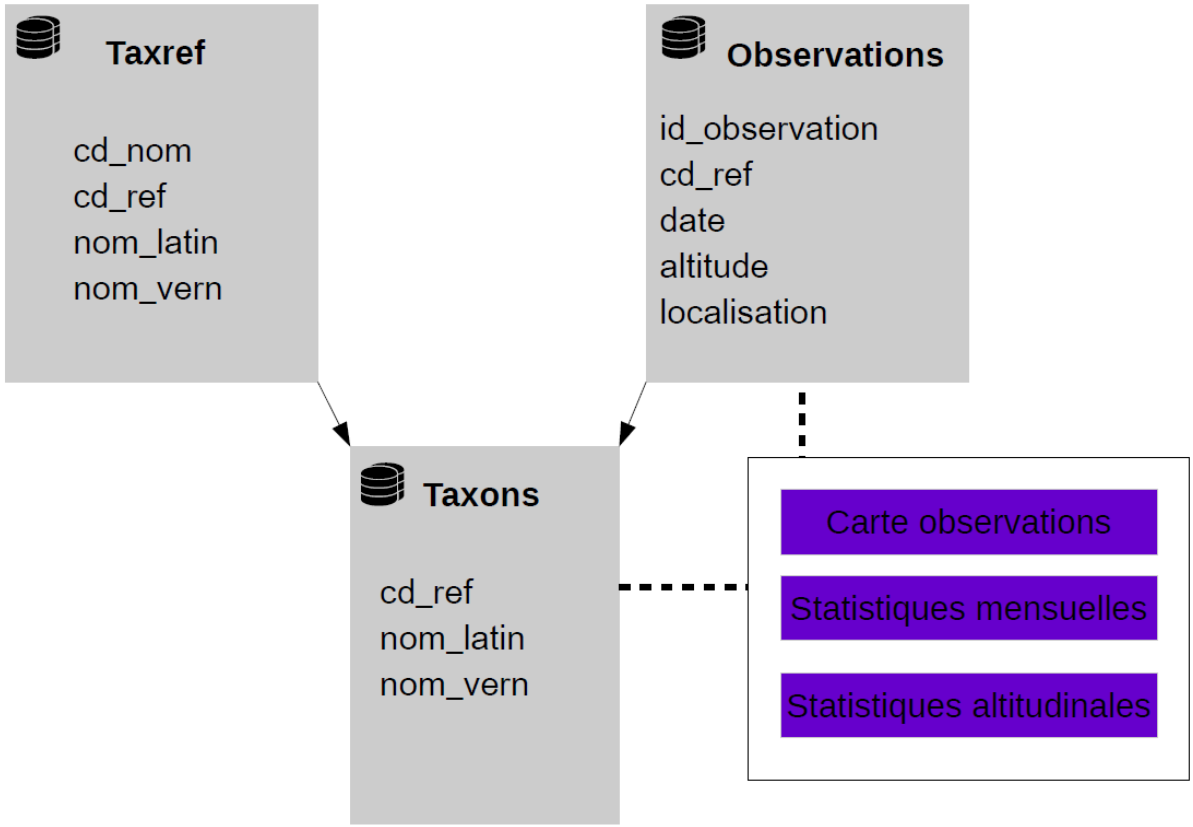
11.2 Annexe-2 : Code de création de la base fille avec mise à jour des vues

```
-- Extension FDW
CREATE EXTENSION IF NOT EXISTS postgres_fdw;

-- Serveur FDW
CREATE SERVER geonaturedbserver
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host '192.168.1.163', dbname 'gens', port '5432');

-- Utilisateur de connexion
CREATE USER MAPPING FOR geonatadmin
SERVER geonaturedbserver
OPTIONS (user 'geonatadmin', password 'admin123');
OPTIONS (user 'geonatatlas', password 'atlas123');

CREATE USER MAPPING FOR postgres
SERVER geonaturedbserver
OPTIONS (user 'postgres', password 'root');

-- Ou pour postgres :
-- CREATE USER MAPPING FOR postgres ...

---Remplissage de gn_meta

create schema gn_meta_tmp;
import foreign schema gn_meta from server geonaturedbserver into gn_meta_tmp;
INSERT INTO gn_meta.cor_dataset_actor(
    id_cda, id_dataset, id_role, id_organism)
select * from gn_meta_tmp.cor_dataset_actor;
DROP SCHEMA gn_meta_tmp CASCADE;

---Remplissage de utilisateurs

create schema utilisateurs_tmp;
import foreign schema utilisateurs from server geonaturedbserver into utilisateurs_tmp;
INSERT INTO utilisateurs.bib_organismes(
    id_organisme, uuid_organisme, nom_organisme, adresse_organisme,
    cp_organisme, ville_organisme,
    tel_organisme, email_organisme, url_organisme, url_logo)
select * from utilisateurs_tmp.bib_organismes;
DROP SCHEMA utilisateurs_tmp CASCADE;
```

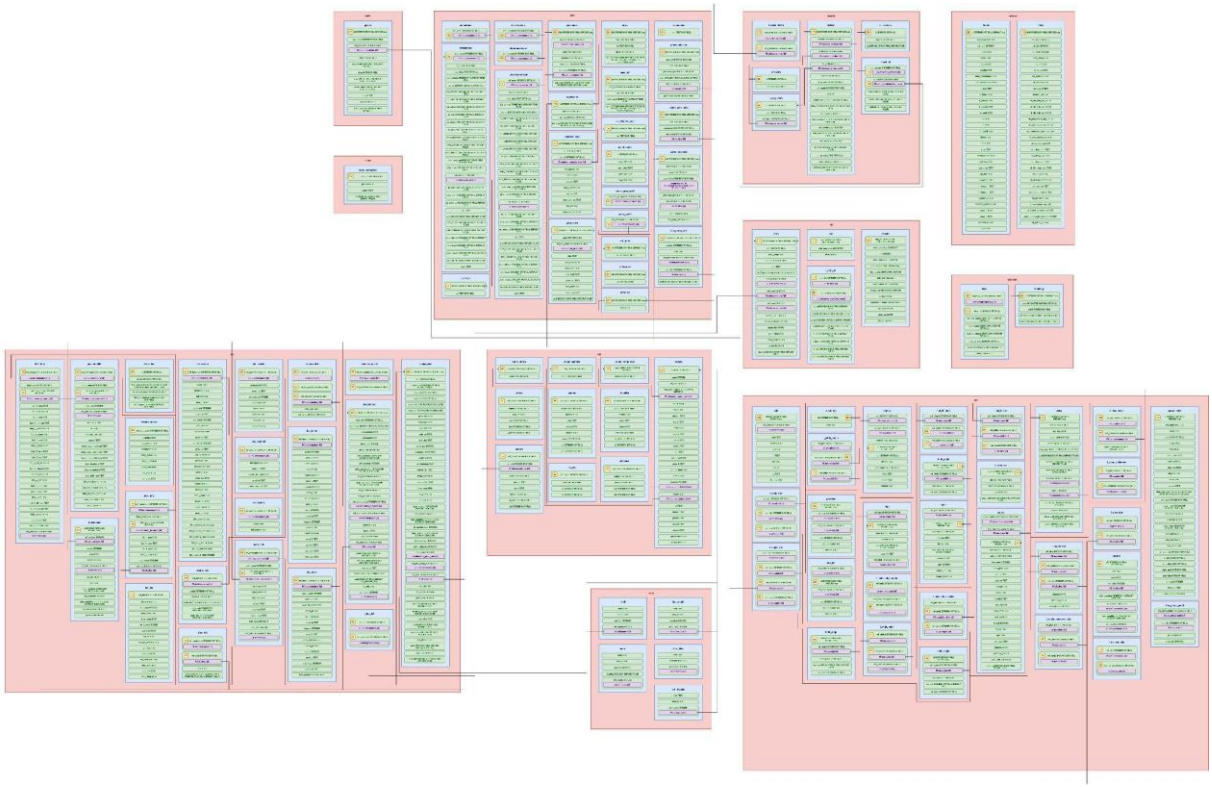
```
---Remplissage de synthese

create schema synthese_tmp;
import foreign schema synthese from server geonaturedbserver into synthese_tmp;
INSERT INTO synthese.syntheseff(
    id_synthese, id_organism, id_dataset, cd_nom, insee, dateobs,
    observateurs, altitude_retenue, the_geom_point, effectif_total,
    diffusion_level, supprime)
select * from synthese_tmp.syntheseff;
DROP SCHEMA synthese_tmp CASCADE;

select pg_get_viewdef('atlas.vm_observations',true);
---mise à jour des vues
REFRESH MATERIALIZED VIEW atlas.vm_observations;
REFRESH MATERIALIZED VIEW atlas.vm_taxons;
REFRESH MATERIALIZED VIEW atlas.vm_search_taxon;
REFRESH MATERIALIZED VIEW atlas.vm_altitudes;
REFRESH MATERIALIZED VIEW atlas.vm_mois;
REFRESH MATERIALIZED VIEW atlas.vm_taxons_plus_observes;
REFRESH MATERIALIZED VIEW atlas.vm_cor_taxon_organism;
REFRESH MATERIALIZED VIEW atlas.vm_observations_mailles;
```

11.3 Annexe-3 Autre MCD de l'atlas




Taxref
cd_nom
cd_ref
nom_latin
nom_vern
Observations
id_observation
cd_ref
date
altitude
localisation
Taxons
cd_ref
nom_latin
nom_vern
Carte observations
Statistiques mensuelles
Statistiques altitudinales

1.1 Annexe-4 : Mcd complet de GENS



11.4 Annexe-5 : Code run_import.py

```
import configparser

from config import SQLALCHEMY_DATABASE_URI


"""
    Exemple d'utilisation de la fonctionnalité importer média depuis médiawiki
    Usage :
        - créer un lien symbolique de config.py
            pour récupérer les paramètres de connexion à la base
        - choisir une requête sql qui récupère la liste des taxons
            pour lequels récupérer des médias
        - paramétrer la fonction main

    Librairies requises (à installer via pip dans un virtualenv de préférence)
        lxml
        psycopg2
        requests
        SPARQLWrapper
        xmltodict
"""
# Constantes type de média et id_type média
# Images
WD_MEDIA_PROP = "P18"
TAXHUB_MEDIA_ID_TYPE = "2"
# Audios
# WD_MEDIA_PROP='P51'
# TAXHUB_MEDIA_ID_TYPE='5'

try:
    conn = psycopg2.connect(SQLALCHEMY_DATABASE_URI)
except Exception as e:
    print("Connexion à la base impossible")
```

```
try:
    cur = conn.cursor()
    sql = """
        SELECT DISTINCT s.cd_nom
    FROM synthese.syntheseff s
    LEFT JOIN taxonomie.bib_noms n ON s.cd_nom = n.cd_nom
    LEFT JOIN taxonomie.t_medias m ON n.cd_ref = m.cd_ref
    WHERE m.id_media IS NULL
    """
    cur.execute(sql)
    rows = cur.fetchall()
except Exception as e:
    print("❌ Problème lors de la récupération de la liste des cd_nom :", e)
    conn.close()
    exit(1)

# Vérification pour éviter l'appel à main() si rows est vide ou non défini
if not rows:
    print("ℹ️ Aucun taxon à traiter.")
else:
    main(conn, rows, WD_MEDIA_PROP, TAXHUB_MEDIA_ID_TYPE, False, False)


conn.close()
```

11.5 Annexe-6 : Code fichier filters.py

```python
# atlas/filters.py

import re

def unique_observers(value):
    """
    Si `value` est une chaîne, on la découpe sur , ou ;.
    Si `value` est une liste, on considère que chaque élément
    est soit déjà un nom, soit une chaîne à découper.
    On renvoie une liste de noms uniques (ordre d'apparition).
    """
    parts = []
    if isinstance(value, (list, tuple)):
        for v in value:
            if isinstance(v, str):
                parts.extend(re.split(r'[;,]', v))
    elif isinstance(value, str):
        parts = re.split(r'[;,]', value)
    else:
        return []

    seen = set()
    uniques = []
    for p in parts:
        name = p.strip()
        if not name:
            continue
        key = name.lower()
        if key not in seen:
            seen.add(key)
            uniques.append(name)
    return uniques

def init_filters(app):
    app.jinja_env.filters['unique_obs'] = unique_observers
```

11.6 Annexe-7 : Code custom.ss

```css
/* Couleurs principales */

:root {
    --main-color: #00bfa6; /* turquoise Eden */
    --second-color: #82c91e; /* vert Eden */
    --map-maille-border-color: black;
}


/* Barre de navigation */
.navbar {
    background-color: var(--main-color) !important;
}
.navbar-brand img {
    height: 60px;
    padding: 5px;
}

/* Liens dans la navbar */
.navbar a {
    color: var(--text-color) !important;
}
.navbar a:hover {
    color: #e0f7fa !important;
}

/* Footer */
footer {
    background-color: var(--second-color) !important;
    color: var(--text-color);
}

/* Boutons */
.btn-primary {
    background-color: var(--main-color);
    border-color: var(--second-color);
    color: var(--text-color);
}
.btn-primary:hover {
    background-color: var(--second-color);
    color: var(--text-color);
```

```
.btn-primary:hover {
    background-color: var(--second-color);
    color: var(--text-color);
}

/* Cartes et survols */
.card {
    border: 1px solid #d0f0f0;
}
.card:hover {
    border-color: var(--main-color);
    box-shadow: 0 0 10px rgba(0, 180, 180, 0.3);
}

/* Titre page accueil */
.introduction-header {
    background-size: cover;
    background-position: center;
    height: 300px;
    color: var(--text-color);
    display: flex;
    align-items: center;
    justify-content: center;
    text-shadow: 1px 1px 2px rgba(0,0,0,0.5);
}

/* Texte principal */
body {
    color: #222;
    background-color: #ffffff;
}

/* Ajout d'un espace vide lorsque la section est masquée
#bandeaulogoshome {
    display: none !important;
}

#bandeaulogoshome-placeholder {
    height: 10px;  ajuster en fonction de la hauteur de la section
```

11.7 Annexe-8 : navbar.html

```
{% block navbar %}
{% if configuration.MULTILINGUAL %}
<script type="text/javascript">
        var language = '{{ '/' + g.lang_code }}';
</script>
{% else %}
<script type="text/javascript">
        var language = '' ;
</script>
{% endif %}

<link href="https://cdnjs.cloudflare.com/ajax/libs/flag-icon-css/3.1.0/css/flag-icon.min.css"
rel="stylesheet">

<nav class="navbar navbar-expand-md navbar-light fixed-top" style="background-color: #e3fdfd;"
role="navigation">
        <!-- LOGO -->
        <a class="navbar-brand" {% if configuration.CUSTOM_LOGO_LINK %}-
href="{{ configuration.CUSTOM_LOGO_LINK }}" target="_blank"{% else %}href="{{ url_for('main.index') }}"{%
endif %}>
        <img class="logoStructure" src="{{ url_for('static', filename='custom/images/structure.png') }}"
alt="Logo Eden62" style="height: 60px;"/>
        </a>

        <!-- TITRE -->
        <a class="navbar-brand titreAppli" href="{{ url_for('main.index') }}" style="font-weight: bold;
color: #009688;">
        {{ configuration.NOM_APPLICATION }}
        </a>

        <!-- BOUTON COLLAPSE -->
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
        </button>
```

```
        <!-- CONTENU NAVBAR -->
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <!-- BARRE GAUCHE -->
        <div class="navbar-nav mr-auto">
                <!-- Ajouter ici des liens si nécessaire -->
        </div>

        <!-- BARRE DROITE -->
        <div class="form-inline my-2 my-lg-0">

                <!-- Recherche espèces -->
                <form method="POST" action="" id='searchFormTaxons' role="search" class="mr-sm-2">
                <input id="searchTaxons" type="text" class="form-control ajax-search small-placeholder"
                        placeholder="{{ _('search.species') }}" loading="false" style="width: 175px;" />
                <input id="hiddenInputTaxons" type="hidden" name="cd_ref" />
                </form>

                <!-- Recherche par site (anciennement commune) -->
                <form method="POST" action="" id='searchFormCommunes' role="search" class="mr-sm-2"
                        onsubmit="completeAction('#searchFormCommunes', hiddenInputCommunes)">
                        <input id="searchCommunes" type="text" class="form-control ajax-search small-
placeholder"
                        placeholder="Recherche par site" style="width: 175px;" />
                        <input id="hiddenInputCommunes" type="hidden" name="insee" />
                </form>


                <!-- Sélecteur de langue -->
                {% if configuration.MULTILINGUAL %}
                <div class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" id="dropdown09" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">
                        <span class="{{ 'flag-icon ' + configuration.AVAILABLE_LANGUAGES[g.lang_code]-
['flag_icon'] }}"></span>
```

```
                        <span class="d-lg-block d-sm-none">  
{{ configuration.AVAILABLE_LANGUAGES[g.lang_code]['name'] }}</span>
                </a>
                <div class="dropdown-menu" aria-labelledby="dropdown09">
                        {% for language in configuration.AVAILABLE_LANGUAGES %}
                        <a class="dropdown-item" href="{{ url_for(request.endpoint, lang_code=language,
**request.view_args) }}">
                        <span class="{{ 'flag-icon ' + configuration.AVAILABLE_LANGUAGES[language]-
['flag_icon'] }}"></span>
                        <span class="d-lg-block d-sm-none">  
{{ configuration.AVAILABLE_LANGUAGES[language]['name'] }}</span>
                        </a>
                        {% endfor %}
                </div>
                </div>
                {% endif %}

                <!-- Zones personnalisées -->
                {% if configuration.EXTENDED_AREAS %}
                {% include 'templates/core/extended_areas_search.html' %}
                {% endif %}
        </div>
        </div>
</nav>
{% endblock %}
```

11.8 Annexe-9 : Code chart.html

```
{% block charts %}
   <!-- <div class="card mt-4" id="graphBloc">
        <h4 class="title-bar title-spaced center strong">{{ _('alt.classes.obs') }}</h4>
        <div class="chart-container" style="position: relative; height:200px; width:100%">
            <canvas id="altiChart"></canvas>
        </div>-->
        <h4 class="title-bar title-spaced center strong">{{ _('monthly.obs') }}</h4>
        <div class="chart-container" style="position: relative; height:200px; width:100%">
            <canvas id="monthChart"></canvas>
        </div>
       <h4 class="title-bar title-spaced center strong">{{ _('sites avec le plus d\'observation') }}</h4>
<div class="chart-container" style="position: relative; height:200px; width:100%">
        <canvas
        id="topSitesChart"
        data-labels='{{ top_sites | map(attribute="site") | list | tojson }}'
        data-values='{{ top_sites | map(attribute="value") | list | tojson }}'
        data-metadata='{{ top_sites | tojson }}'>
        </canvas>
</div>


    </div>
{% endblock %}
```

11.9 Annexe-10 : Code chart.js

```
// ChartJS Graphs
const chartMainColor = getComputedStyle(document.documentElement).getPropertyValue('--main-color');
const chartHoverMainColor = getComputedStyle(document.documentElement).getPropertyValue('--second-color');

const getChartDatas = function (data, key) {
    let values = [];
    for (var i = 0; i < data.length; i++) {
        values.push(data[i][key])
    }
    return values
};

//Generic vertical bar graph
genericChart = function (element, labels, values) {
    return new Chart(element, {
        type: 'bar',
        data: {
            labels: labels,
            datasets: [{
                label: 'observations',
                data: values,
                backgroundColor: chartMainColor,
                hoverBackgroundColor: chartHoverMainColor,
                borderWidth: 0
            }]
        },
        options: {
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true
                    }
                }],
                xAxes: [{
                    gridLines: {
```

```
                    gridLines: {
                        display: false
                    }
                }]
            },
            maintainAspectRatio: false,
            plugins: {
                legend: {
                  position: 'top',
                  display: false
                },
            }
        }
    });
};
var topSitesChartElement = document.getElementById('topSitesChart');
if (topSitesChartElement) {
    const labels = JSON.parse(topSitesChartElement.dataset.labels);
    const values = JSON.parse(topSitesChartElement.dataset.values);
    const metadata = JSON.parse(topSitesChartElement.dataset.metadata);

    new Chart(topSitesChartElement, {
        type: 'bar',
        data: {
            labels: labels,
            datasets: [{
                label: 'observations',
                data: values,
                backgroundColor: chartMainColor,
                hoverBackgroundColor: chartHoverMainColor,
                borderWidth: 0
            }]
        },
        options: {
            plugins: {
```

```
            plugins: {
                tooltip: {
                    callbacks: {
                        label: function (context) {
                            const i = context.dataIndex;
                            const debut = metadata[i].annee_debut;
                            const fin = metadata[i].annee_fin;
                            return `De ${debut} à ${fin}`;
                        }
                    }
                },
                legend: {
                    display: false
                }
            },
            scales: {
                y: {
                    beginAtZero: true
                },
                x: {
                    grid: {
                        display: false
                    }
                }
            },
            maintainAspectRatio: false
        }
    });
}

var monthChartElement = document.getElementById('monthChart');
const monthChart = genericChart(monthChartElement, months_name, getChartDatas(months_value, 'value'));
```

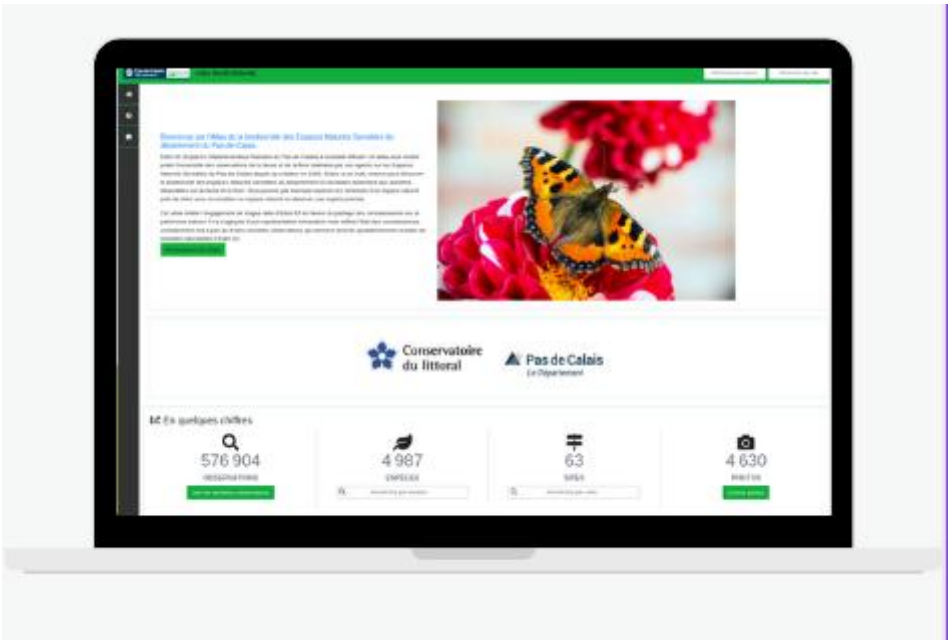
11.10 Annexe-11 : Code vmSiteRepository.py

```
from sqlalchemy.sql import text

def getTopSitesByTaxon(connection, cd_ref, limit=5):
    sql = """
        SELECT
            c.commune_maj AS site,
            f.insee,
            COUNT(*) AS obs_count,
            MIN(EXTRACT(YEAR FROM f.dateobs))::int AS annee_debut,
            MAX(EXTRACT(YEAR FROM f.dateobs))::int AS annee_fin
        FROM synthese.syntheseff f
        JOIN atlas.vm_communes c ON f.insee = c.insee::text
        WHERE f.cd_nom = :cd_ref AND f.supprime IS FALSE
        GROUP BY c.commune_maj, f.insee
        ORDER BY obs_count DESC
        LIMIT :limit
    """
    result = connection.execute(text(sql), {"cd_ref": cd_ref, "limit": limit})
    return [{
        "site": row["site"],
        "value": row["obs_count"],  # Pour les barres
        "annee_debut": row["annee_debut"],
        "annee_fin": row["annee_fin"]
    } for row in result]
```

11.11 Annexe-12 : maps-custom.js

```
// Fonction style d'affichage des points dans la fiche espèce
// Voir documentation leaflet pour customiser davantage l'affichage des points: http://leafletjs.com/-
reference-1.3.0.html#circlemarker-option
var pointDisplayOptionsFicheEspece = (pointDisplayOptionsFicheCommuneHome = function(feature) {
  var color = "#999999"; // couleur par défaut si aucun niveau reconnu

  switch (feature.properties.diffusion_level) {
        case 0:
        color = "#1e88e5"; // Bleu foncé : diffusion standard (mailles, ZNIEFF, etc.)
        break;
        case 1:
        color = "#fdd835"; // Jaune : flouté à la commune
        break;
        case 2:
        color = "#82c91e"; // Vert  : flouté à la maille 10x10
        break;
        case 3:
        color = "#fb8c00"; // Orange : flouté au département
        break;
        case 4:
        color = "#ef5350"; // Rouge : non diffusé (devrait ne pas s'afficher)
        break;
        case 5:
        color = "#00bfa6"; // Turquoise : données précises (non floutées)
        break;
  }

  return {
        radius: 6,
        color: color,
        fillOpacity: 0.6,
        weight: 2,
        fillColor: "#ffffff"
  };
});
```

```
var divLegendeFicheEspece = (divLegendeFicheCommuneHome =
  '\
        <p><div>\
        <i class="circle" style="border: 3px solid #82c91e; border-radius: 50%; width: 20px; height:
20px;"></i>Localisation approximative\
        </div></p>\
        <p><div>\
        <i class="circle" style="border: 3px solid #00bfa6; border-radius: 50%; width: 20px; height:
20px;"></i>localisation précise\
        </div></p>\
');
```



Du 07/04/2025 au 27/06/2025

Par SITA-MOKOKO

Etudiante en BUT2 de Science des Données parcours VCOD