Université Grenoble-Alpes
Master 2 DCISS

Rapport de mi-stage

Refonte de GeoNature pour moderniser et modulariser les applications web

Rapport présenté par :
Quang PHAM
Master 2 DCISS

Maître de stage :
Camille Monchicourt
Parc National des Écrins

Tuteur pédagogique
Jean-Michel Adam
Responsable du Master DCISS

Table des matières

1. Contexte :

1.1 Présentation de la structure d'accueil :

Créé en mars 1973, le Parc national des Écrins est un établissement public à caractère administratif placé sous la tutelle du ministère de l'Ecologie, du Développement durable et de l'Energie.

Ses personnels développent des missions de connaissance, de préservation des espèces et des milieux, d'accueil du public et d'accompagnement du développement du territoire.

Avec une petite centaine de salariés, le Parc national des Écrins est organisé autour d'équipes de terrain qui animent les sept secteurs géographiques du territoire.

Le Parc national des Écrins est un établissement public qui a pour première vocation la préservation de la biodiversité et la mise en place d'une politique de développement durable sur son territoire. Ce territoire de haute-montagne s'étend sur 53 communes entre les départements des Hautes-Alpes - région PACA - et de l'Isère - région Rhône-Alpes-Auvergne.

L'équipe permanente du PNE est constituée d'environ 100 personnes réparties entre le siège à Gap, et sept secteurs qui couvrent l'ensemble du territoire du parc (figure 1). Le siège est composé de quatre services : le service scientifique, le service aménagement, le service communication et le service général.

1.2 L’équipe de travail :

Mon stage se déroule au sein du service scientifique du PNE. Ce service est lui même divisé en deux pôles:

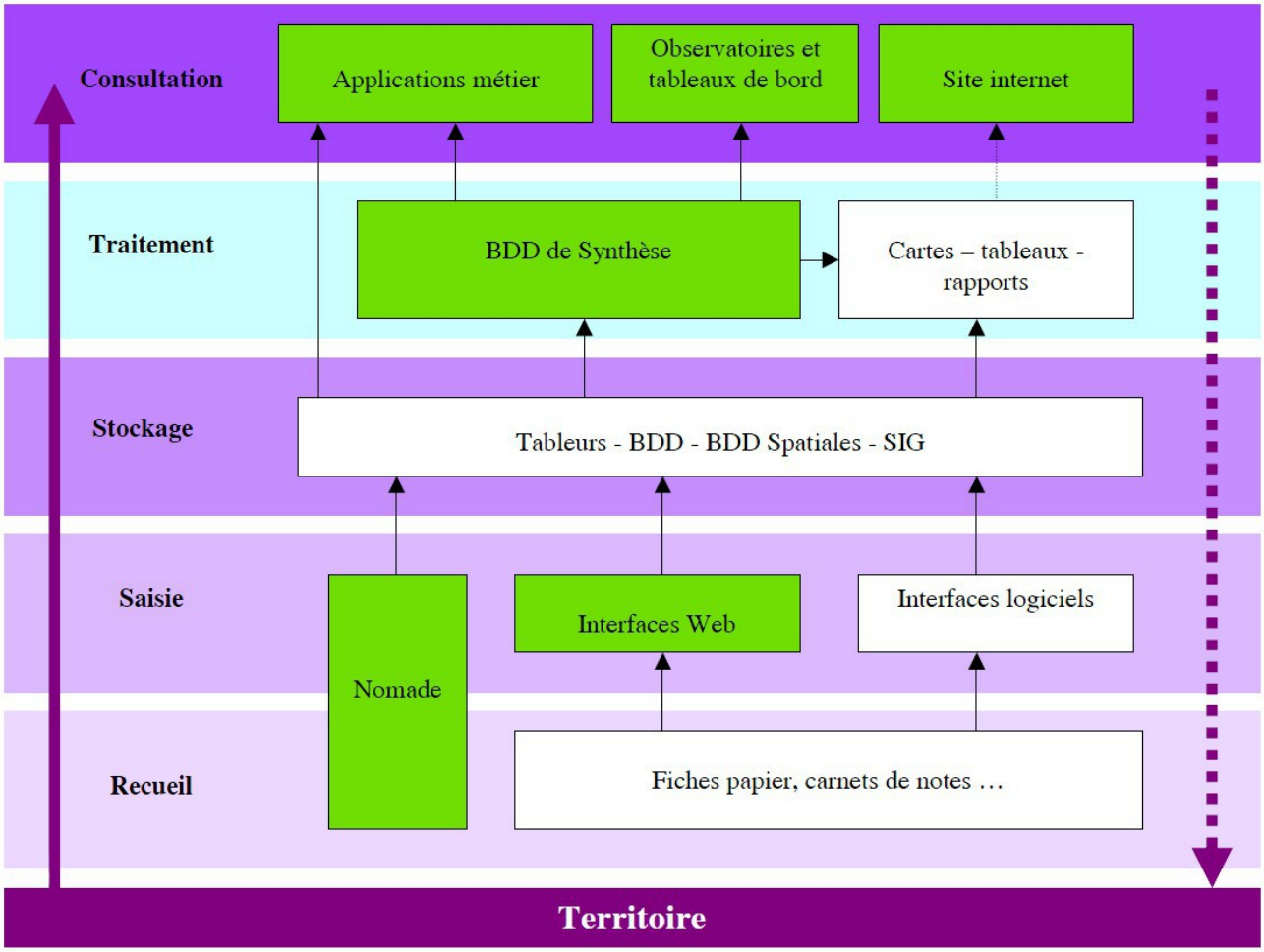
- Le « pôle connaissance » qui travaille sur la mise en place de protocoles de suivi scientifique (faune, flore et mesures physiques)
- Le pôle système d'informations, dans lequel je me trouve, qui s'occupe de la géomatique et de l'informatique. Il est constitué d'un chargé de mission base de données et développement web, d'un chargé de mission administration réseau, téléphonie et informatique, et d'un géomaticien, chef du pôle.

1.3 Les missions du SI :

Le pôle SI occupe une position transversale puisqu'il est amené à travailler avec tous les services du Parc. Il assiste aussi bien le service scientifique dans la mise en place de protocoles de suivi faune-flore, que le service aménagement dans le suivi du patrimoine bâti et de l'agriculture, ou encore le service communication dans le mise en place d'outils de mise en valeur des sentiers de randonnées et l'animation du site web.

De part ses missions de protection de la faune et de la flore, le parc national est amené à collecter des quantités importantes de données spatialisées. Le rôle du SI au sein du parc est donc d'organiser et de faciliter la collecte de ces données, de les gérer mais également de créer des outils pour les analyser. Une grande composante métier du SI tient donc dans l'administration de bases de données.

Le PNE a été novateur dans la mise en place de la collecte et le stockage des données sur informatique et possède aujourd'hui une architecture de base de données et des outils structurés. Le schéma ci-dessous (figure 2) résume la modernisation de la stratégie générale du SI, du recueil de la donnée jusqu’à son traitement et sa consultation. En fonction des protocoles et des besoins des agents, plusieurs chaînes de travail ont été mises en place. La collecte sur le terrain, anciennement effectuée sur papier (en blanc sur le schéma) est aujourd'hui saisie sur des outils nomades (applications mobiles sur tablette). Pour les protocoles importants, l'ensemble de ces données sont ensuite centralisées dans des bases de données PostgreSQL (avec extension PostGIS pour les besoins spatiaux). A partir de ces bases de données, des applications de consultation métier ou grand public sont développées soit par des prestataires extérieurs , soit directement par le SI.



La chaîne de travail du SI : du recueil à la consultation des données (document interne).

On distingue en vert la chaîne de travail actuelle, et en blanc l'ancienne.

2. Objectif général, Geonature, Taxhub :

2.1 Objectif général :

Refondre les applications utilisées par les parcs nationaux (ObsOcc et GeoNature) pour en faire un outil open source plus modulaire, plus moderne, plus ouvert, qui intègre de nouvelles fonctionnalités et permet à chaque structure d'y intégrer des modules pour ses protocoles spécifiques

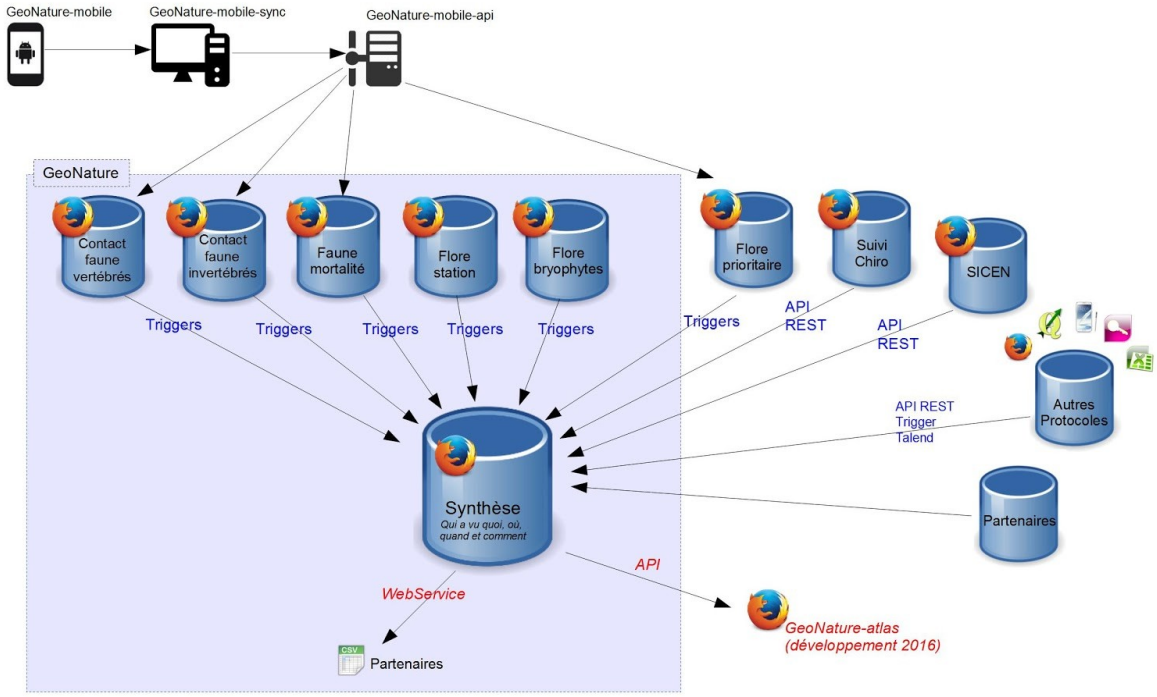
2.2 Contexte:

Actuellement les parcs nationaux utilisent les outils ObsOcc (aka SICEN) et GeoNature (+ KaruNati au PNG et PNRun) pour la gestion de leurs protocoles et données Faune et Flore. Ces 2 outils sont développés en interne depuis 2010 et deviennent obsolescents techniquement (technologies vieillissantes, non maintenues, présentant donc des défauts de compatibilité, d'ergonomie et de performances). Ils sont partiellement redondants et ont été développés dans des contextes spécifiques. Le premier n'est pas compatible avec la stratégie scientifique inter-parc, ni la stratégie commune SI qui est en train d'être construite. Le second n'est pas assez modulaire pour permettre à chacun de l'adapter à ses besoins et protocoles. Par ailleurs les protocoles, le contexte et les besoins des structures ont évolué depuis la conception de ces outils.

2.3 Besions:

- Répondre aux besoins de connaissance et de gestion locaux en matière de faune et de flore
- Faciliter les échanges de données locaux et nationaux (avec instances et partenaires)
- Disposer d'une chaîne de travail et d'outils performants, modernes, modulaires et complets
- Disposer d'une organisation des données souple et solide, ouverte vers les partenaires, structurée à partir d'un socle et de référentiels communs tout en facilitant à chaque structure la possibilité d'y intégrer ses propres protocoles
- Agglomérer, valoriser et diffuser la connaissance naturaliste des parcs nationaux.

2.4 Geonature:



Architecture de la BDD faune-flore du PNE (document interne)

GeoNature est une application web métier qui permet :

- De saisir des données de différents protocoles
- D'intégrer des données externes en gardant leur structuration brute (partenaires ou outils de saisie externes),
- De sourcer l'ensemble de ces données (protocole, programme, jeu de données, organisme producteur)
- D'agglomérer automatiquement le tronc commun de ces données (Qui a vu quoi, où et quand), dans une Synthèse.
- De faire des recherches avancées dans cette synthèse et d'en exporter le résultat (XLS des observations ou des statuts, SHP des observations)
- De prédéfinir des exports sur mesure et de les rendre accessibles à certains partenaires ou certains agents internes
- De préparer les données de la Synthèse pour les mettre à disposition de tous dans l'outil grand public GeoNature-atlas GeoNature est construit dans une approche par protocole selon le principe « Un besoin = Une question = Un protocole = Un modèle de données = Un outil »

2.5 Taxhub :

Application web de gestion centralisée des taxons basée sur le référentiel TAXREF (http://inpn.mnhn.fr/programme/referentiel-taxonomique-taxref) du MNHN.

TaxHub permet de gérer la liste des taxons présents dans un territoire, d'y greffer des informations spécifiques, de définir des listes de taxons et des filtres en fonction des besoins des différents protocoles.

L'application permet aussi de gérer les descriptions et les médias des taxons pour leur affichage sur GeoNature-atlas.

TaxHub permet d'administrer un schéma 'taxonomie' dans une BDD PostgreSQL. Celui-ci comprend les tables de TAXREF et leur contenu complet.

Il comprend aussi des tables complémentaires qui permettent de créer des listes de taxons (pour limiter la saisie aux taxons déjà vus une fois sur le territoire, ne garder en saisie que certains synonymes ou pour créer des sous-ensembles de taxons pour des protocoles spécifiques).

3. Cahier des charges:

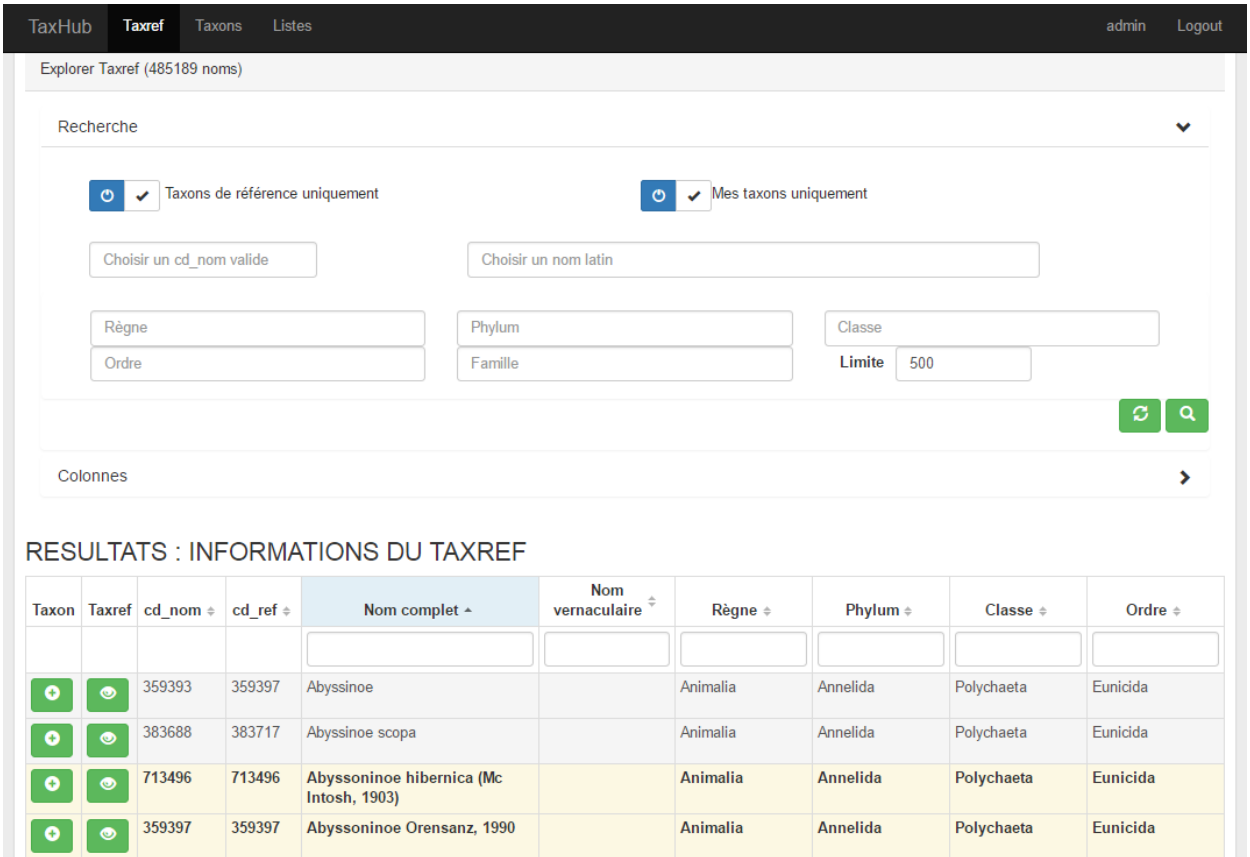
3.1 Cahier des charges – Taxhub :

3.1.1 Fonctionnalités existantes :

- Recherche d'un taxon (référence ou synonyme) dans TaxRef (plus de 450 00 noms),
- Administration d'une liste des taxons présents sur le territoire d'une structure,
- Renseignement des attributs complémentaires nécessaires à GeoNature ou pour qualifier les taxons.
  - Les attributs complémentaires sont gérés de manière générique dans la BDD. Chaque structure peut ainsi se créer autant d'attributs que nécessaire.
  - Un attribut a un nom, un type (booléen, texte, liste 1-n ou n-n…), des valeurs, une description.
  - Un attribut peut n'être proposé que pour un règne ou un Groupe2 INPN.
  - Un attribut peut être associé à un thème pour faire des groupes d'attributs, regroupés dans le formulaire de saisie.
- TaxHub permet aussi de gérer les descriptions et les médias publiés sur GeoNature-atlas.
  - Description, En savoir plus, Milieu(x), Chorologie

- Les médias peuvent être des photos, des audios, des vidéos, des liens HTML ou des PDF. Il est possible de les uploader sur le serveur de TaxHub ou bien de pointer vers des médias existants en renseignant leur URL. Les médias ont un type, un titre, une légende, un auteur, une description. Ils peuvent être publics ou non.

- Enfin TaxHub permet d'associer un taxon à une ou plusieurs listes de taxons.
- L'authentification est centralisée dans l'application UsersHub. Il faut donc disposer d'une application nommée « TaxHub » dans la liste des applications gérées par UsersHub et y intégrer des groupes et/ou des utilisateurs disposant de droits spécifiques.
  - Niveaux de droits utilisés dans TaxHub et fonctionnalités correspondantes :
    - 2 = Gestion des médias uniquement
    - 3 = Idem 2 + Gestion des attributs de GeoNature-atlas
    - 4 = Idem 3 + Possibilité d'ajouter des taxons, de les mettre dans des listes et de renseigner tous leurs attributs (notamment ceux utilisés par GeoNature)
    - 6 = Administrateurs



Application Taxhub

3.1.2 Développements à réaliser dans TaxHub :

TaxHub utilise déjà les technologies cibles de cette refonte (Python Flask, AngularJS, Bootstrap). Il ne sera donc pas refondu et n'est concerné que par quelques évolutions.

- Suivi des tickets : https://github.com/PnX-SI/TaxHub/issues
- Suivi des évolutions : https://github.com/PnX-SI/TaxHub/releases
- Documentation sur http://taxhub.readthedocs.io
- Démonstration sur http://92.222.107.92/taxhub/ (admin / admin)

Module listes :

La gestion des listes et des noms de taxons qu'elles contiennent doit se faire en base de données. Aucune interface dans l'application TaxHub ne permet de consulter, créer, modifier une liste ou de gérer le contenu d'une liste. L'administration des listes doit donc se faire en SQL directement dans la base de données.

Le travail consiste donc à créer un module « listes » permettant de réaliser ces opérations avec l'application TaxHub.

Fonctionnalités attendues pour le module « Listes » :

- Afficher toutes les listes des noms avec les informations de chaque liste : le nom de la liste, les filtres « règne » et « groupe2 INPN », le nombre des noms dans une liste. Le tableau présentant les listes peut être filtré par nom latin, nom français, cd_nom et id_nom
- Voir le détail d'une liste avec sa description et lister les noms de taxons appartenu à la liste avec leurs nom français, nom latin, règne, groupe2 INPN, et identifiants.
- Exporter une liste au format de fichier .csv.
- Gérer les noms appartenant à une liste (ajouter et retirer des noms de taxons).
- Éditer et créer une liste (identifiant, nom, description, picto, règne et groupe2 INPN).

3.2 Cahier des charges – Geonature 2 :

3.2.1 Statistiques :

A réaliser :

- Refonte technologies et ergonomie >> Python, Flask, Angular 4, Bootstrap, Leaflet
- Statistiques globales présentant les données dans les BDD, par organisme, par règne, par groupe INPN, par année, par protocole...

- Accès à la synthèse et aux applications des différents protocoles (intégrés ou non dans GeoNature) selon les droits de l'utilisateur
- Boutons d'exports basés sur des vues dans la BDD accessibles selon les droits de l'utilisateur. Pour créer un bouton d'export, il faut créer une vue dans la BDD correspondant aux données nécessaires et à leur forme souhaitée puis y indiquer les utilisateurs ou groupes pouvant y accéder. Cela permet de rendre les partenaires autonomes pour l'extraction de données ou de créer des exports pour les besoins récurrents d'agents internes.

3.2.2 Contact (Faune/Flore) :

Le contact faune flore est un processus d'observation de la faune ou de la flore. Il est caractérisé par l'absence de protocole conduisant à l'observation. L'observation se fait de manière aléatoire dans le temps et dans l'espace : pas de méthode de prospection ou recherche, pas de constante (heure, durée, surface, date, température,…).

Présentation du protocole : http://geonature.fr/protocoles.html#contact-faune

A réaliser :

- Refonte technologies et ergonomie >> Python, Flask, Angular2/4, Bootstrap.
- Appuyer sur le catalogue de méthodes et protocoles du MNHN : http://campanule.mnhn.fr
- Utiliser la liste des techniques de CAMPANULE du Museum national pour associer une technique d'observation à chaque observation.
- L'idée est de faire 3 contacts avec leur schéma de BDD et leur propre formulaire (Faune vertébré, Faune invertébré, Flore/Fonge) et de faire un outil Bibliographie à part, un pour les collections, un pour les pelotes...
- Un outil avec son schéma/formulaire pour les contacts spécifiques à un groupe ou une espèce ou si on privilégie la logique de pseudo-champs.

A réaliser :

- Refonte technologies et ergonomie >> Python, Flask, Angular 4, Bootstrap, Leaflet

3.2.3 Flore Station :

Ce protocole permet d'observer une liste d'espèces dans le milieu physique qu'elles occupent http://geonature.fr/protocoles.html#flore-station

Il consiste à relever l'ensemble des éléments d'une station floristique :

- Données de l'observation (ou métadonnées)

- Données stationnelles
- Espèces présentes dans les limites du relevé, avec leurs abondances relatives notée pour chaque strate occupée
- Outil de Consultation / Recherche / Saisie / Modification / Export

A réaliser :

- Refonte technologies et ergonomie >> Python, Flask, Angular 4, Bootstrap, Leaflet

3.2.4 Flore prioritaire :

Suivi de la flore prioritaire (espèces patrimoniales et invasives).

Ce protocole consiste à noter toutes les informations susceptibles d'aider à la compréhension des évolutions des populations d'espèces patrimoniales ou invasives.

Il s'agit de délimiter les surfaces occupées par l'espèce considérée (aires de présences) et les surfaces prospectées où l'espèce n'a pas été vue. L'ensemble des 2 surfaces forme la zone de prospection.

Pour chaque aire de présence, la quantité d'individus présents est mesurée par la fréquence d'occurrence de l'espèce le long de transects non permanents.

Outil de Consultation / Recherche / Saisie / Modification / Export

http://geonature.fr/protocoles.html#flore-prioritaire

A réaliser :

- Refonte technologies et ergonomie >> Python, Flask, Angular2/4, Bootstrap, Leaflet

3.2.5 Synthèse:

Carte / Liste / Filtres / Détail / Export

QUOI : Filtre par règne, patrimonialité, protection, taxon, ensemble de taxons

QUAND : Filtre par date ou période

OU : Filtre par commune, par massif, par zonage (réserve, N2000…), par dessin d'un polygone sur la carte ou par import d'un polygone

QUI : Par observateur ou structure

COMMENT : Par protocole, programme, lot de données

A réaliser :

- Refonte technologies et ergonomie >> Python, Flask, Angular2/4, Bootstrap, Leaflet

4. Suivi de projet utilisé GitHub :

4.1 Github, qu’est-ce que c’est?:

GitHub (exploité sous le nom de *GitHub, Inc.*) est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. Ce site est développé en Ruby on Rails et Erlang par Chris Wanstrath, PJ Hyett et Tom Preston-Werner. GitHub propose des comptes professionnels payants, ainsi que des comptes gratuits pour les projets de logiciels libres. Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki pour chaque projet.

En avril 2016, GitHub a annoncé avoir dépassé les 14 millions d'utilisateurs et plus de 35 millions de dépôts de projets le plaçant comme le plus grand hébergeur de code source au monde.

Le nom GitHub est composé du mot « git » faisant référence à un système de contrôle de version open-source et le mot « hub » faisant référence au réseau social bâti autour du système Git.

4.2 Suiviles projets inter-parc avec GitHub :

Tous les projets de Parc national des Ecrins, sont des projets open source et GiHub est un outil indispensable à la gestion du développement des projets. On utilise Github pour:

- Héberger le code des projets.
- Travailler sur des projets à plusieurs parcs.
- Documenter les projets.
- Suivre le développement des autres.
- Demander d’ajouter des fonctionnalités.
- Signaler des problèmes de code.
- Proposer des modifications de code à d'autres
- https://github.com/PnEcrins

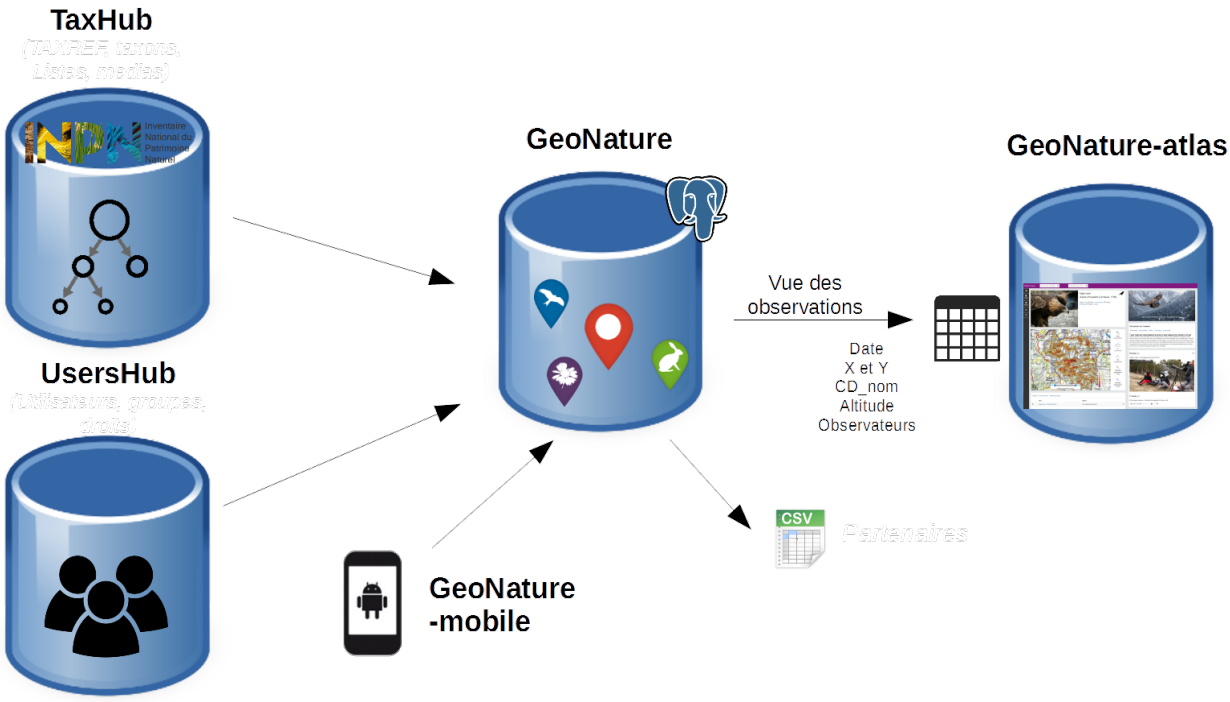
5. Les projets existants et choix des technologies :

5.1 Les projets existants :

Plusieurs projets réalisés ces dernières années ont montré qu'il était possible de développer des outils pour les besoins spécifiques tout en les concevant de manière générique pour permettre à d'autres structures d'en bénéficier. Ainsi Geotrek a été développé par 2 parcs nationaux en 2011 dans le cadre d'une prestation de 70.000 euros. L'outil est aujourd'hui utilisé par une quarantaine de structures en France de types variés (PNX, PNR, ComCom, Départements, CDRP, CDT...) organisé en communauté (http://www.ecrins-parcnational.fr/actualite/geotrek-avenementcommunaute-utilisateurs), bénéficiant chacune des évolutions apportées par les autres. L'outil a aujourd'hui une valeur estimée à plus d'un million d'euros et la plupart des structures ont pu le déployer librement grâce à sa licence open source.

Le projet GeoNature 2 s'appuiera sur cette expérience en organisant plus en amont la gouvernance et le travail collectif des partenaires et en s'appuyant sur des développements internes pour réduire les coûts, ne pas dépendre d'un prestataire et avoir une meilleure maîtrise de ce projet au cœur des SI des utilisateurs.

Les développements s'appuieront sur les outils réalisés depuis 5 ans par le Parc national des Ecrins et le Parc national des Cévennes : GeoNature, GeoNature-mobile, GeoNatureatlas, Projet_Suivi, TaxHub et UsersHub en y intégrant des fonctionnalités existantes dans ObsOcc (aka SICEN). La valeur de ses outils open source est estimée à 1,7 millions d'euros (source : https://www.openhub.net).



Les projets existantes – Parc National Des Ecrin

5.2 Choix des technologies :

5.2.1 Technologies TaxHub :

TaxHub utilise les technologies qui sont retenues pour le projet GeoNature2.

Afin de m'adapter aux technologies de GeoNature 2 et au mode de travail de l'équipe SI, pendant le premier mois mon travail a été de réaliser le module « Listes » de TaxHub avec les technologies suivantes :

- Serveur : Debian ou Ubuntu
- Langages : Python, HTML, JS, CSS
- Framework python : flask
- Framework JS : AngularJS
- Framework CSS : Bootstrap
- BDD : PostgreSQL, PostGIS
- Gestion du code : Git et Github
- Poste de travail sous linux (Ubuntu 16)

Il y a 3 modules dans l'application « Taxref », « Taxons » et « Listes ».

« Taxref » et « Taxons » ont déjà été réalisés avant mon arrivée.

5.2.2 Technologies GeoNature :

Inconvénient des technologies dans Geonature 1:

PHP symfony 1.4: vieille version d'un framework complexe, peu modulaire et impossible de le faire fonctionner sur php 7.

Extjs : ce framework n'est quasiment plus utilisé ; les versions utilisées ne sont plus du tout maintenu et des incompatibilités avec l'implémentation javascript des navigateurs modernes sont déjà survenues (corrigées) mais pourrait devenir bloquantes. Extjs n'est pas réellement open source, son interface est vieillissante. Et surtout, c'est le framework js qui prend en charge totalement le DOM (c'est le JS qui construit tout le HTML). Il est donc difficile de personnaliser l'interface.

Renforcer le projet avec Python Flask et Angular 4:

Python Flask: un micro-framework, il est facile d'apprendre à l'utiliser, c'est un petit framework, qui de plus consomme moins de ressources et impose souvent moins de contraintes et le moins de lignes de code possible.

Flask a été utilisé dans le projet TaxHub et GeoNature Atlas. Dans Geonature 2 Flask remplacera Symfony.

Angular 4: Mon stage a donc été l'occasion pour le parc national de faire de nouveaux choix en terme de technologie, en accord les SI inter-parcs.

On a discuter pour choisir entre Angular 4 et VueJs.

VueJs est un framework intéressant qui monte fortement, il est très adapté à des projets léger ou pour des développeurs moins expérimentés.

Angular 4 est un framework récent, il est sorti en décembre 2016. Il est soutenu par Google. C'est un framework lourd mais adapté à la taille du projet, avec une forte rigueur d'organisation de l'architecture du code, notamment pour obtenir un découpage pertinent et fonctionnel en modules suffisamment autonomes.

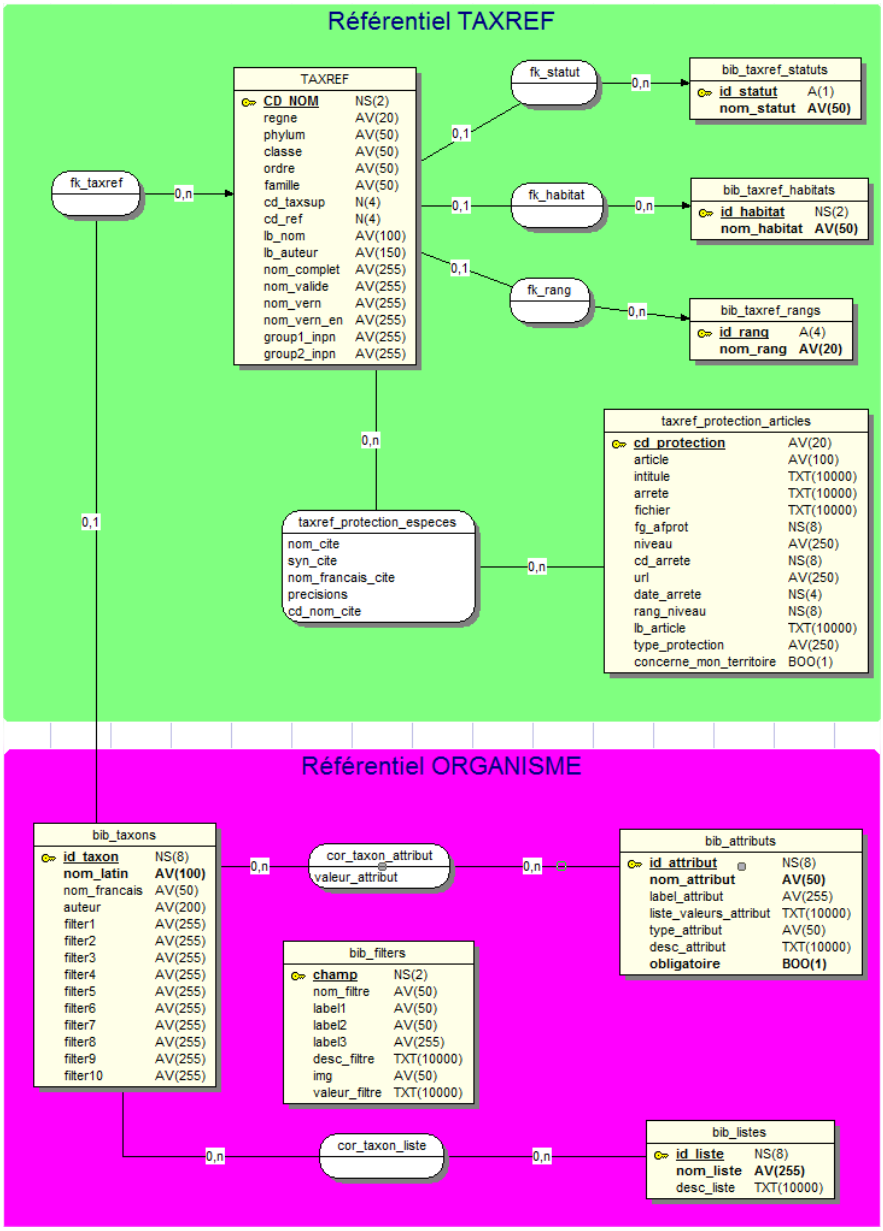
Finalement, l'équipe inter-parc a choisi Angular 4 parce qu'il est plus tôt adapter à un grand projet comme GeoNature 2.

	Langages	Bdd	Back-end	Front-end	Carto
Geonature v1	Php, html, js, css	Postgresql, postgis	Php : symfony	Extjs	Geoextjs, mapfish openlayers 2
Geonature v2	Python, js html, css	Postgresql, postgis	Python : flask	Angular 4 Bootstrap	Leaflet

6. Conception – Réalisation :

6.1 TaxHub – Module Listes:

6.1.1 Modèle de donnée, structure de l'application :

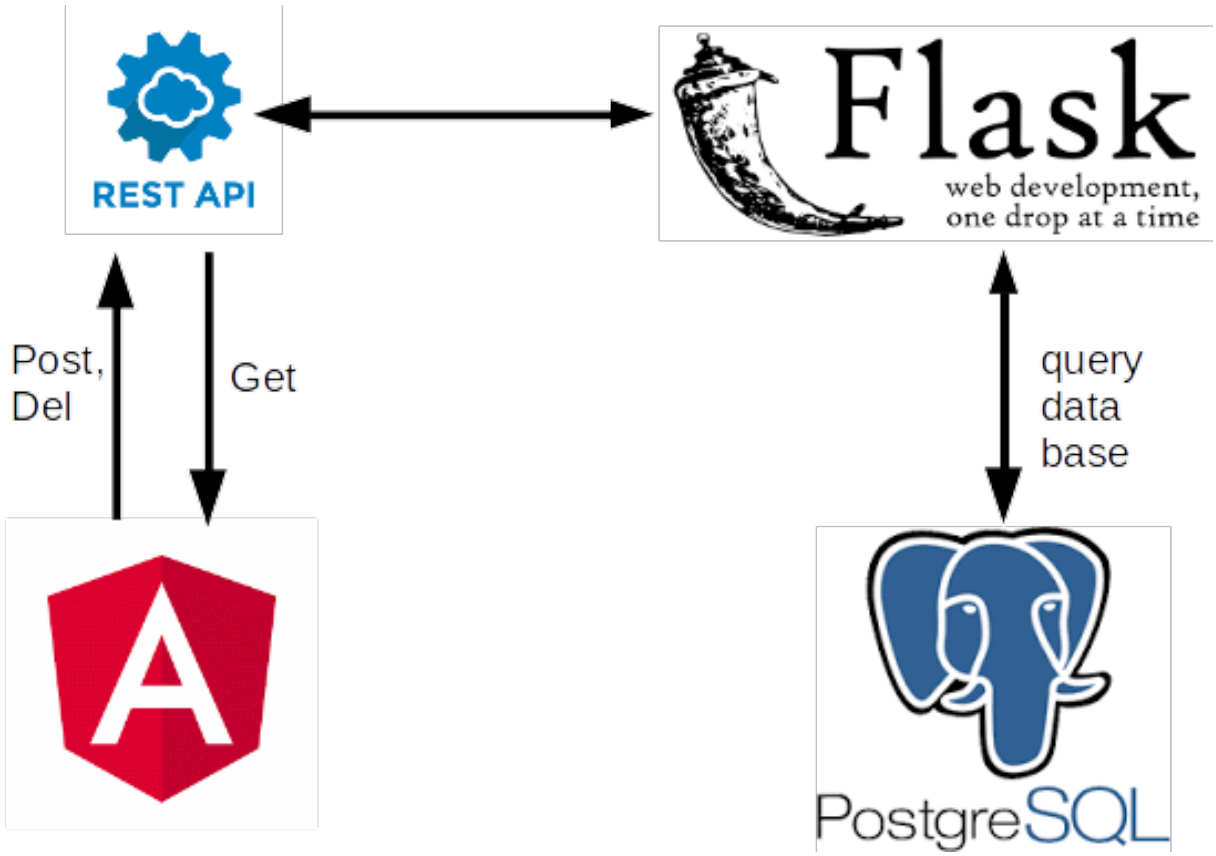


Taxhub - Modèle de donnée

Sur ce modèle de données, les interfaces pour TAXREF et bib_taxons sont déjà réalisées. Mon premier travail a été d'apprendre le plus vite possible les technologies et les concepts nécessaires à la réalisation du module « Listes ».

Pour travailler sur le module « Listes », Il faut bien connaître trois aspects de l'application:

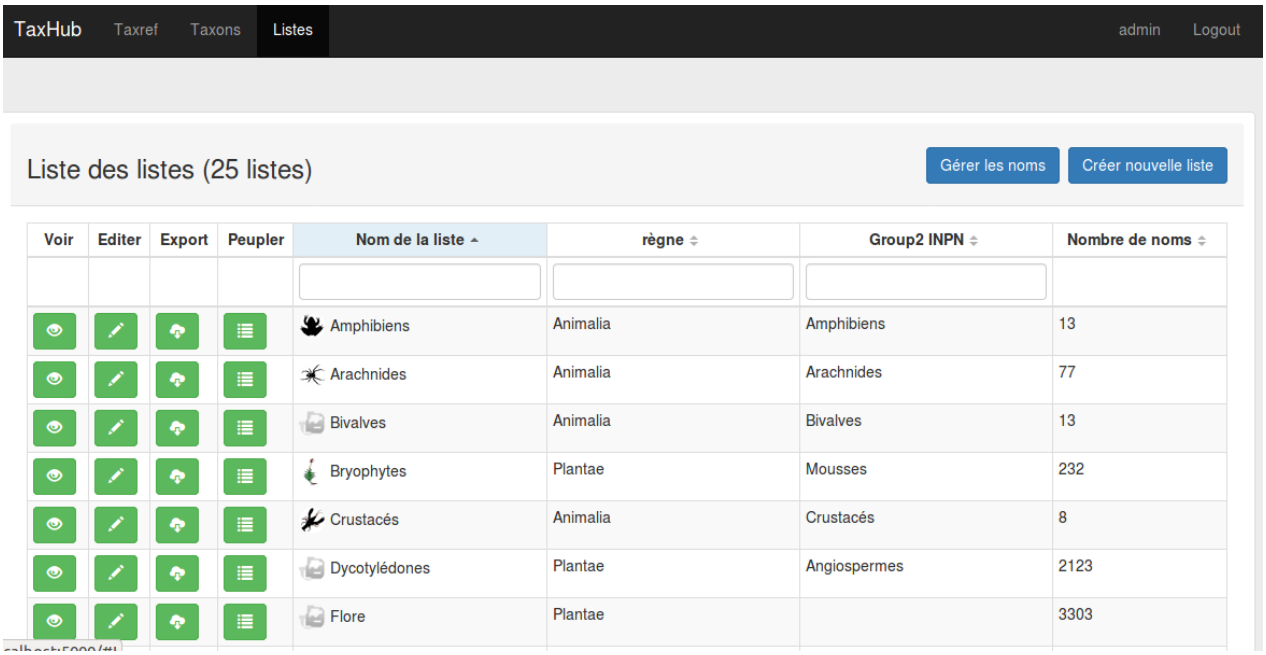
- Font-end : Angularjs 1.6
- Back-end : Python Flask
- Base de donné : PostgreSQL



Structure d'application Taxhub

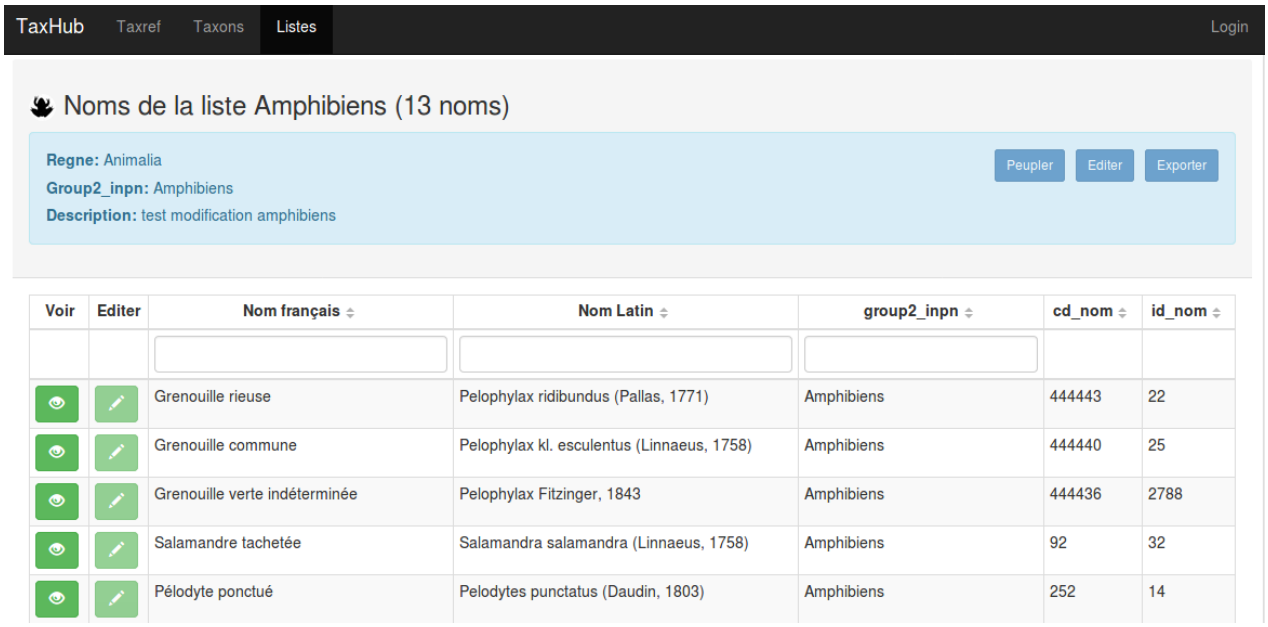
6.1.2 Réalisation :

Conception font-end :



Interface du module Listes

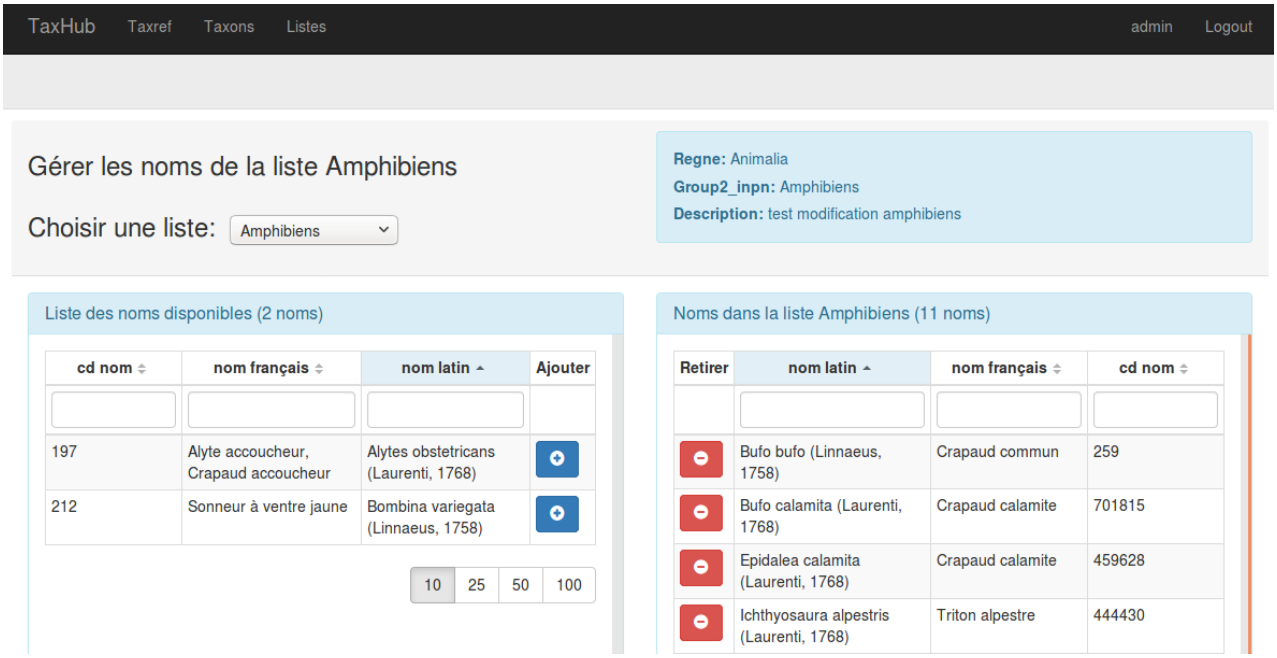
- Afficher le nombre de listes existantes
- Afficher toutes les listes avec les informations de chacune des listes
    - Le nom de la liste, le règne, le groupe2 INPN, le nombre de noms dans la liste.
    - Des boutons Voir détail, Éditer, Exporter, Peupler, Gérer les noms, Créer une liste
    - Un pictos à côté du nom de la liste
- Niveaux de droits – Login – Désactiver les buttons (et interdire l'accès aux routes coté backend-end) si le niveau de droit ne correspond pas avec les fonctionnalités



Voir	Editer	Nom français	Nom Latin	group2_inpn	cd_nom	id_nom
		Grenouille rieuse	Pelophylax ridibundus (Pallas, 1771)	Amphibiens	444443	22
		Grenouille commune	Pelophylax kl. esculentus (Linnaeus, 1758)	Amphibiens	444440	25
		Grenouille verte indéterminée	Pelophylax Fitzinger, 1843	Amphibiens	444436	2788
		Salamandre tachetée	Salamandra salamandra (Linnaeus, 1758)	Amphibiens	92	32
		Pélodyte ponctué	Pelodytes punctatus (Daudin, 1803)	Amphibiens	252	14

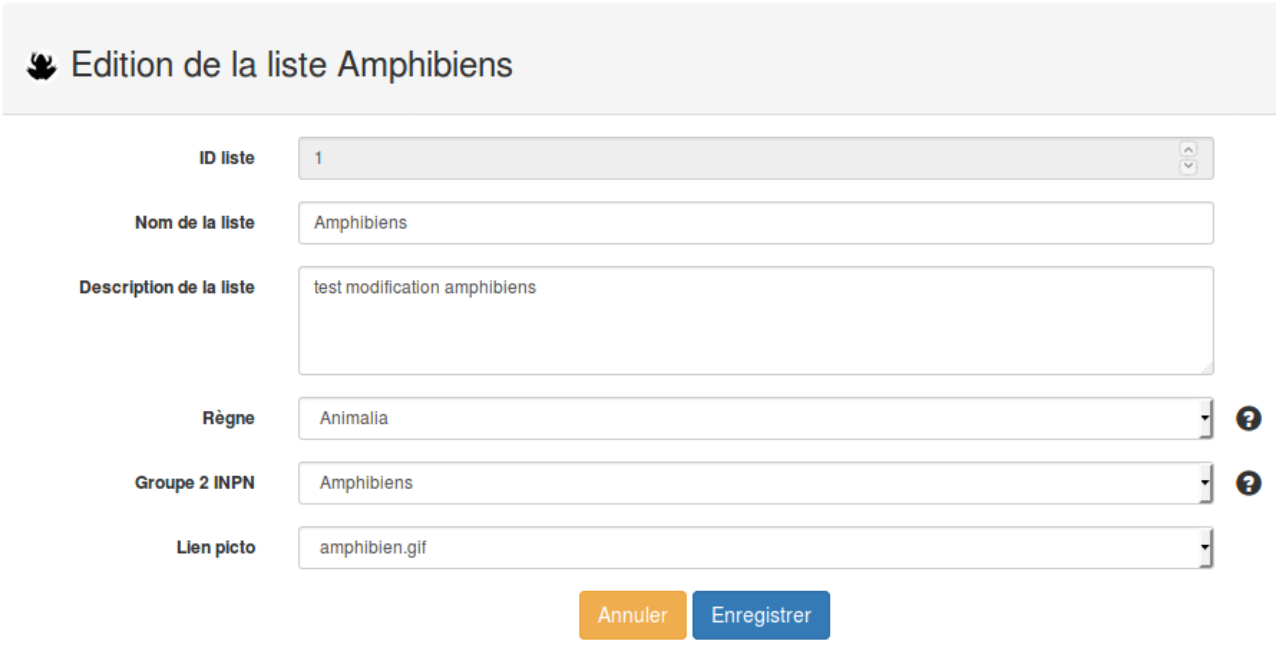
Interface pour consulter le détail d'une liste

- Voir le détail d'une liste avec sa description en haut.
- Présenter sous forme de tableau les noms de la liste avec leurs nom français, nom latin, règne, groupe2 INPN, cd_nom, id_nom.
- Possibilité de filtrer les colonnes par nom français, nom latin, group2 INPN.
- Voir le détail d'un nom. Cette fonctionnalité est liée avec le module Taxons pour détailler les donnée d'un nom : Attributs (patrimonial, protégé), Taxref, les listes qui portent ce nom, et les médias attachés à ce nom.
- Éditer le détail d'un nom. Cette fonctionnalité est liée avec le module Taxons pour modifier les données d'un nom : Attributs (patrimonial, protégé), Taxref, les listes portent le nom, et les médias de nom.
- Exporter une liste au format de fichier .csv.
- Niveaux de droits – Login – Désactiver les buttons si le niveau de droit ne correspond pas avec les fonctionnalités.



Interface gérer les noms

- Gérer les noms dans une liste (ajouter et retirer les noms).
- Détailler l'information de la liste (règne, groupe2 INPN, description)
- Les boutons ajouter et retirer un nom.
- Le panel à gauche liste les noms disponibles : il proposer les noms disponibles pour pouvoir les ajouter à la liste. Ces noms sont une intersection entre règne et group2_inpn et ils ne doivent pas être présents pas dans le panel à droite.
- Niveaux de droits – Login – Désactiver les boutons si le niveau de droit ne correspond avec les fonctionnalités.
- Bouton de sélection en haut pour changer de liste.



Interface – Éditer une liste

- Éditer une liste (le nom, la description, le picto, le règne et le groupe2 INPN).
- Désactiver id liste, identifiant unique de la liste dont la modification n'est pas permise en interface.
- Proposer une liste d'item dans « groupe 2 inpn » en lien avec le choix du règne
- Proposer les pictos disponibles pour la liste
- Point d'interrogation à côté de « Règne » et « Groupe 2 INPN » pour un tooltip qui précise la fonction du champ « règne » et « groupe2 INPN ».
- Niveaux de droits – Login – Désactiver les boutons si le niveau de droit ne correspond avec les fonctionnalités.

  Pendant la conception de front-end, nous avons discuté ensemble des idées d'organisation des composants de l'interface (les boutons, les panels, les contenus afficher, le contenu des tooltip...).

Conception backend :

```python
from pypnusershub import routes as fnauth

db = SQLAlchemy()
adresses = Blueprint('bib_listes', __name__)


@adresses.route('/', methods=['GET'])
@json_resp
def get_biblistes(id = None):
        """
        retourne les contenu de bib_listes dans "data"
        et le nombre d'enregistrements dans "count"
        """
        data = db.session.query(BibListes, func.count(CorNomListe.id_nom).label('c'))
            .filter(BibListes.id_liste == CorNomListe.id_liste)\
            .group_by(BibListes)\
            .order_by(BibListes.nom_liste).all()
        maliste = {"data":[],"count":0}
        maliste["count"] = len(data)
        for l in data:
            d = l.BibListes.as_dict()
            d['nb_taxons'] = l.c
            maliste["data"].append(d)
        return maliste
```

Les routes du back-end

- 16 routes qui sont créées pour le module Listes.
- Pour le nom des routes, il faut suivre le standard REST
- Pour chacune des routes il faut vérifier le niveau de droit de l'utilisateur authentifié.
- Les requêtes sqlAchemy, doivent être optimisées. Le temps nécessaire à la récupération des données dans la base peut parfois être très long.
    - ***Solution 1 récupérer donnée avec les clés étranger taxref : 981kb de donnée --> 12,69 seconds***

```
def get_bibtaxons():
data = db.session.query(BibNoms).all()
taxonsList = []
for r in data :
  obj = r.as_dict()
  obj['taxref'] = r.taxref.as_dict()
  taxonsList.append(obj)
return taxonsList
```

- ***Solution 2 sqlachemy sqldirect: 981kb de donnée --> 389 miliseconds***

```
def get_bibtaxons():
data = db.engine.execute("\
  select tbn.cd_ref,tbn.id_nom, tbn.cd_nom, tbn.nom_francais,
tt.nom_complet \
  from taxonomie.bib_noms tbn, taxonomie.taxref tt \
  where tbn.cd_nom = tt.cd_nom")
results = []
for row in data:
  data_as_dict = {
    'nom_complet' : row.nom_complet,
    'nom_francais': row.nom_francais,
    'cd_nom': row.cd_nom,
    'id_nom': row.id_nom,
    'cd_ref': row.cd_ref}
  results.append(data_as_dict)
return results
```

- ***Solution 3 utiliser filter() pour optimiser: 981kb de donnée --> 420 miliseconds***

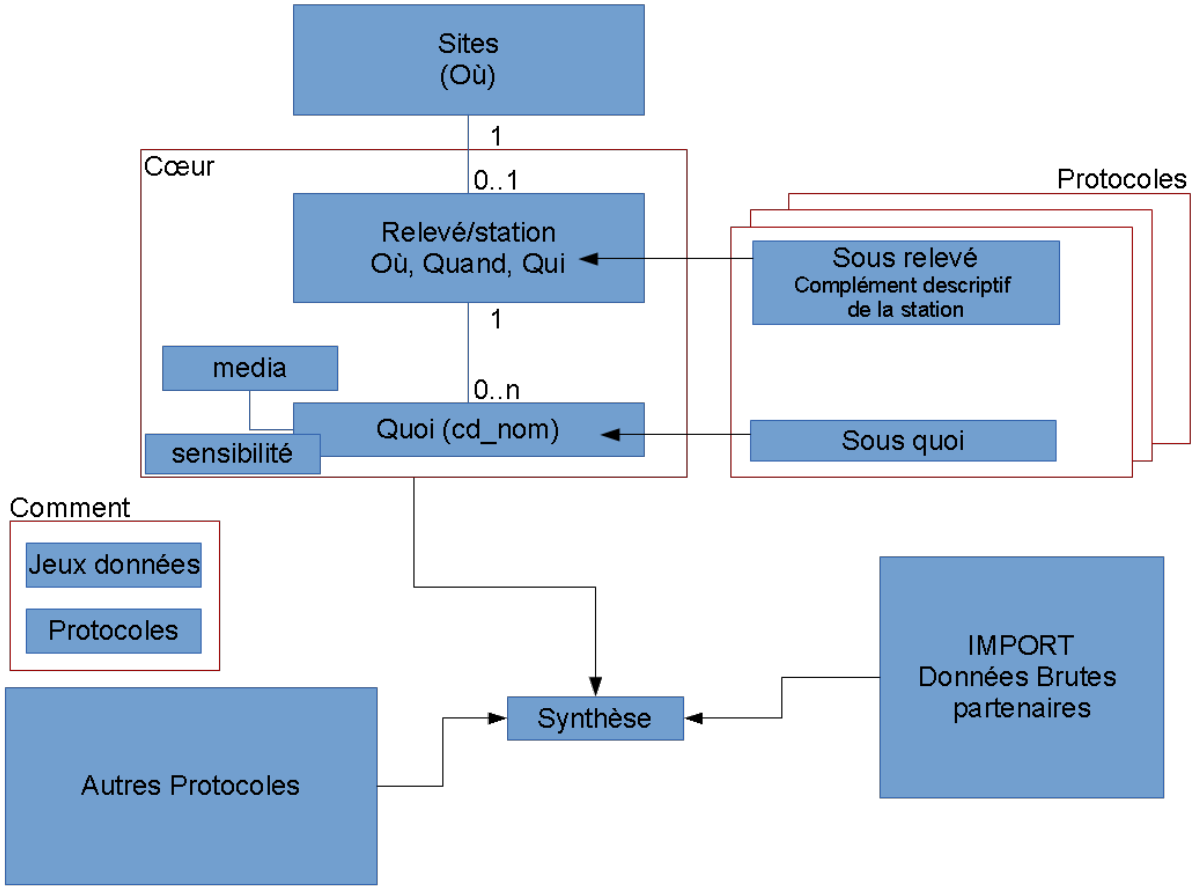
```
def get_bibtaxons():
 data = db.session.query(BibNoms, Taxref.nom_complet,
Taxref.regne, Taxref.group2_inpn).filter(BibNoms.cd_nom ==
Taxref.cd_nom).all()
  results = []
  for row in data:
    data_as_dict = row.BibNoms.as_dict()
    data_as_dict['nom_complet'] = row.nom_complet
    data_as_dict['regne'] = row.regne
    data_as_dict['group2_inpn'] = row.group2_inpn
    results.append(data_as_dict)
  return results
```

- On peut voir <solution 2> c'est le plus efficace (~33 fois plus vite que solution 1)

6.2 Géonature :

6.2.1 Modèle de donnée, structure de l'application :

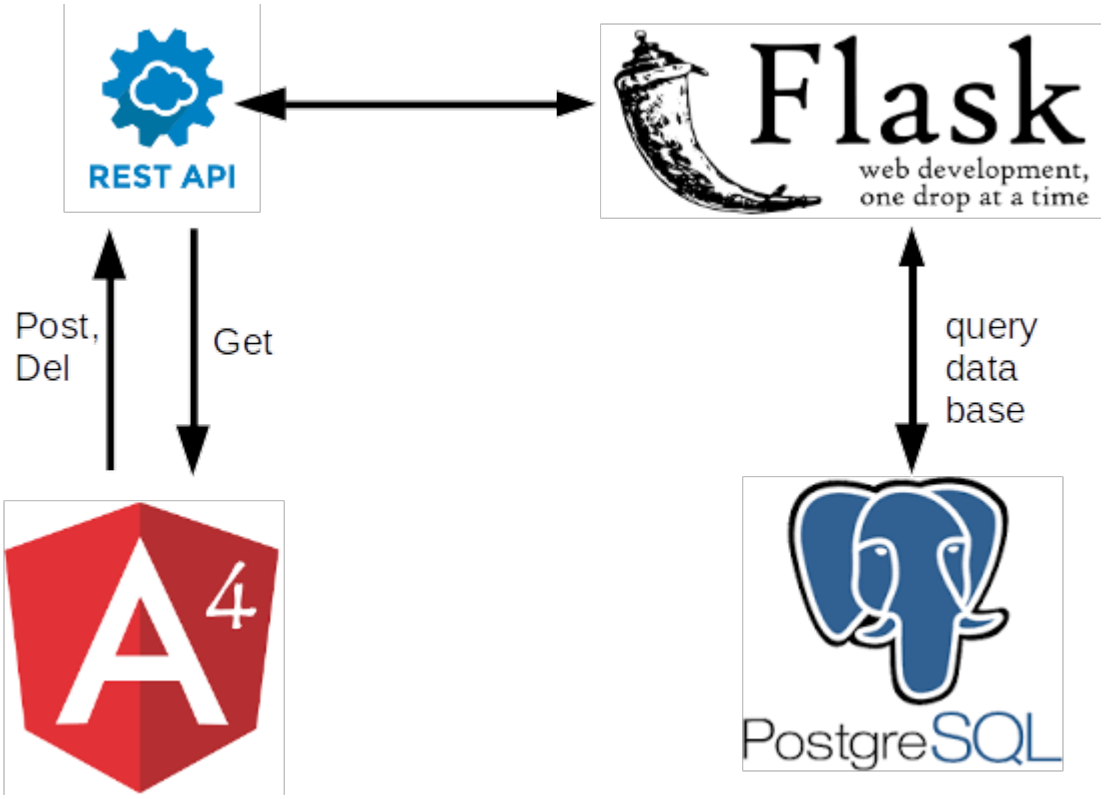
Modèle de donnée



Ebauche du modèle de données de GeoNature 2

Détail du modèle de données de GeoNature 1 :
https://github.com/PnEcrins/GeoNature/blob/master/docs/2017-01-mcd_geonaturedb_1.8.2.png

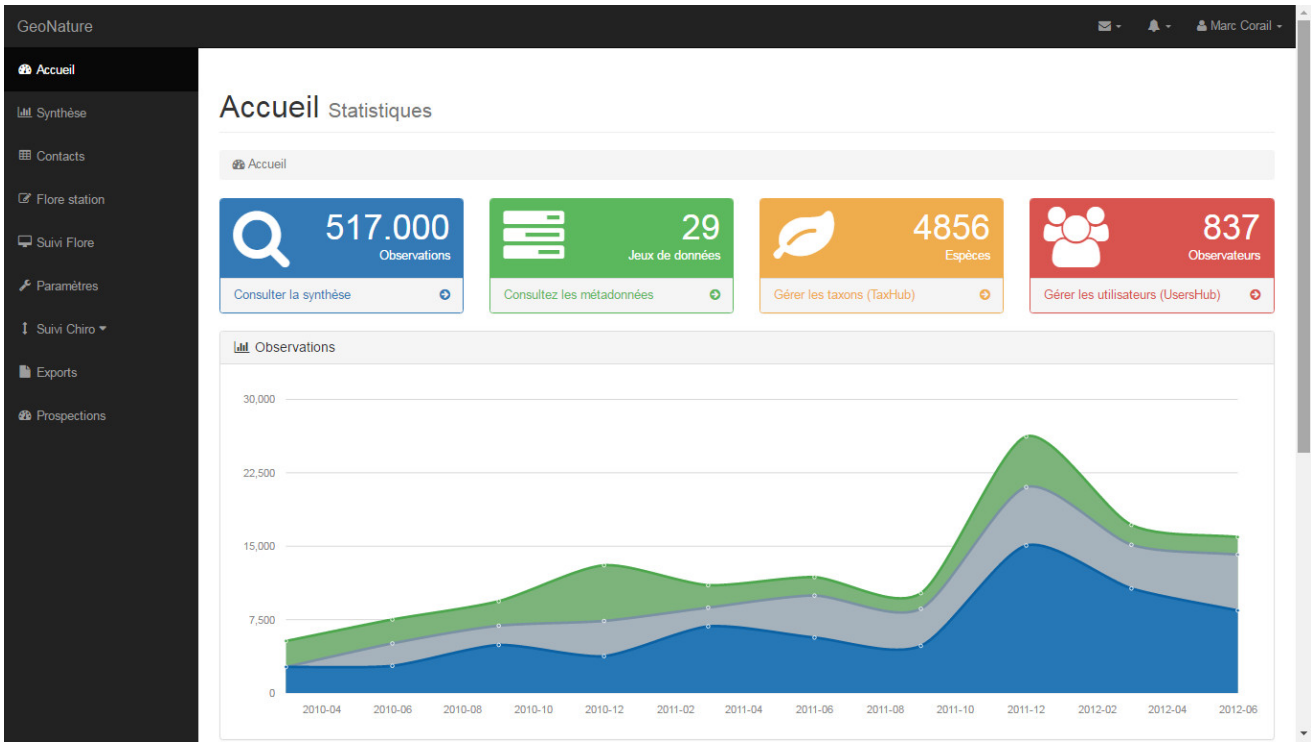
Structure de l'application :



Structure d'application Geonature 2

6.2.2 Réalisation :

Ce sont les parties à compléter durant les 4 mois de stage.



Maquette de la page d'accueil de GeoNature 2

7. La phase de tests :

7.1 Test fonctionnalité :

Toutes les fonctionnalités seront testées selon 2 cas : normal et anormal.

Il faut s'assurer que l'application capture bien dans toutes les erreurs. J'ai essayé de créer des erreurs pendant la phase de test ; comme casser des routes, envoyer des données incorrectes au backend et à la base de données pour m'assurer que l'application a bien capturé toutes les erreurs. Il subsiste des bugs et des fonctionnalités à améliorer :

- Listes édition : le formulaire se vide lors d'une erreur d'enregistrement. Issue #117 dans Github https://github.com/PnX-SI/TaxHub/issues/117
- Listes : La liste des listes n'est pas mise à jour après un update/insert. Issue #115 dans Github https://github.com/PnX-SI/TaxHub/issues/115

7.2 Test d'intégration:

Au début, j'ai développé l'application avec une base de donnée de petite taille. Toutes les fonctionnalités ont bien fonctionné mais quand on a branché l'application sur une base de donnée de taille beaucoup plus grande, un bug d'affichage a été détecté.

Ce bug a été corrigé.

7.3 Test performance:

Certaines requêtes posaient de problèmes de performances.

Pour tester, j'ai écrit au nivaux du backend des fonctions différentes pour récupérer les mêmes données dans la base de données. J'ai pu identifier qu'avec une bonne requête, le temps de récupération peut être jusqu'à 34 fois plus rapide qu'avec une mauvaise requête.

7.4 Test avec outils de test :

Il y a beaucoup d'outils pour tester le code. Dans application Geonature v2 nous allons utiliser Karma pour faire les tests unitaires avec Angular 4 et Pytest pour tester au niveau du backend python.

8. La conclusion :

Après deux mois, ce stage m'a permis de développer de nombreuses nouvelles compétences. J'ai énormément appris, autant sur la partie architecture et développement web, que sur la partie base de données, mais aussi sur les nouvelles technologies (python Flask, Angularjs, Angular2/4, PostgreSQL, REST API). J'ai également eu la chance de suivre et participer à la conception du projet GeoNature 2 (définition du cahier des charges), à sa réalisation (choix des technologies et de l'architecture, développement), ce qui fut très enrichissant.