



Rapport de stage Création d'un atlas dynamique de la faune et de la flore au Parc National des Écrins



Rapport présenté par :
Théo Lechémiat
Master 2 DCISS

Maitre de stage :
Camille Monchicourt
Parc National des Ecrins
Tuteur pédagogique
Jean-Michel Adam
Responsable du Master DCIS

Table des matières

Introduction :.....	3
1) Présentation de la structure d'accueil et du service.....	4
Le Parc National des Écrins.....	4
Les missions du SI.....	5
2. L'atlas faune – flore du Parc national des Écrins : cahier des charges et architecture.....	7
Historique et contexte du projet de l'atlas faune-flore.....	7
Données naturalistes et dynamiques de l'open-data.....	8
Contenu de l'atlas.....	10
3. Travaux réalisés.....	11
3.1 Gestion de projet et définition du contenu de l'atlas.....	11
Revue des atlas faune-flore existant.....	11
Réunions de coordination du projet.....	12
3.2 Dimension technique du travail.....	12
3.2.1 Mise en place de l'environnement de travail.....	12
3.2.2 Découverte des applications et des bases de données du parc national.....	13
3.2.3 Choix de l'architecture web mise en place.....	14
3.2.4 La partie base de données.....	17
3.4 La phase de développement.....	19
Planning et gestion du développement.....	19
Effort de généricité du code.....	22
Aperçu de l'application.....	23
Conclusion et perspectives.....	27
Annexes.....	29

Introduction :

Le Parc national des Écrins (PNE), où je réalise actuellement mon stage, est un établissement public national dépendant du ministère de l'Environnement. En France, les Parcs nationaux ont été créés par la loi du 22 juillet 1960 ; ils ont pour principale mission la gestion et la protection d'un territoire exceptionnel en terme de faune, de flore, ou encore de paysage.

Le Parc National des Écrins a lui été officiellement créé en 1973. Ses missions de gestion et de protection du milieu naturel ont amené les agents du parc à collecter depuis cette date des centaines de milliers de données naturalistes. Historiquement, ces données servaient à alimenter des études scientifiques et étaient avant tout utilisées en interne. Depuis 2006 et l'instauration d'une « charte des Parc Nationaux », un dynamisme d'ouverture et de diffusion des données s'est mis en place. Ces données doivent non seulement être mises à disposition des partenaires institutionnels (ministères, financeurs), mais également être accessibles au grand public.

Ces obligations, concordantes à l'arrivée de nouvelles technologies, devaient nécessairement entraîner un bouleversement dans la collecte, l'organisation, et le traitement de l'information. Le PNE a donc procédé à une réorganisation de ses services en mutualisant le service informatique et SIG dans un pôle SI (Système d'Informations), afin de mener à bien ces missions. Depuis une dizaine d'années le SI a développé une chaîne de travail complète: passant de la collecte de données sur papier et la saisie manuelle sur des tableurs à une collecte numérique et un transfert automatique dans des bases de données spatiales centralisées.

Cette architecture a permis au Parc de développer des outils de diffusion et de vulgarisation des données scientifiques. Une des stratégies récentes de l'établissement est le SIT (Système d'information territorial), elle consiste à développer des applications web qui valorisent auprès du grand public la singularité du territoire du PNE ainsi que le travail des agents du parc. Deux exemples récemment mis en ligne illustrent cette stratégie :

- l'application « Bouquetins » qui permet de suivre en temps réel le parcours de bouquetins équipés de capteurs GPS (<http://bouquetins.ecrins-parcnational.fr/>) .
- « Rando Écrins » : une application permettant la visualisation et la recherche de randonnées, de patrimoines et d'hébergements sur le parc (<http://rando.ecrins-parcnational.fr/fr/>).

Dans le cadre du SIT, le PNE souhaite aujourd'hui développer une application de visualisation des ses données faunes et flores à travers un atlas en ligne. C'est sur ce projet que porte mon stage de 5 mois au PNE.

Dans ce rapport de stage, je présenterai d'abord brièvement, le service et l'équipe avec laquelle je travaille ainsi que le rôle que j'occupe. Dans un deuxième temps nous nous attarderons sur la présentation du projet de l'atlas (cahier des charges, objectifs et attentes du parc) ainsi que l'architecture de base de données du parc sur laquelle s'appuiera cet atlas. Enfin dans une dernière partie nous parlerons du travail réalisé jusqu'à aujourd'hui autant en terme de gestion de projet, de choix technologiques que de développement.

1) Présentation de la structure d'accueil et du service

■ Le Parc National des Écrins

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 et de l'Isère et les régions PACA et 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 (voir figure 1). On distingue au siège quatre services : le service scientifique, le service aménagement, le service communication et le service général.

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 scientifiques (faune, flore et mesures physiques), et le pôle système d'information, 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 SI.

Durant ce stage, ma mission sera en grande partie le développement web de l'atlas. Pour ce projet je serai amené à travailler en grande partie avec Gil Deluermoz (le chargé de mission développement Web et BDD) et Camille Monchicourt (chef du pôle SI) qui coordonnera le projet. Pour la partie thématique, nous travaillerons également avec le pôle connaissance du service scientifique et avec le service communication.



Figure 1: Carte de localisation du PNE et de ses secteurs

▪ Les missions du SI

Le pôle SI occupe une position assez 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 scientifique, que le service aménagement dans le suivi du patrimoine bâti et de l'agriculture, ou encore avec le service communication dans la mise en place d'outil de mise en valeur des sentiers de randonnées et l'animation du site web.

De part ces missions de protection de la faune et de la flore, le parc 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 interpréter. Une grande composante métier du SI tient donc dans l'administration de base 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 traitement ont été mise en place. La collecte sur le terrain, anciennement effectuée sur papier (en blanc sur le schéma) est aujourd'hui saisie sur des outils nomades (application smartphone et 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 métiers ou grand public sont développées soit par des prestataires extérieurs, soit directement par le SI.

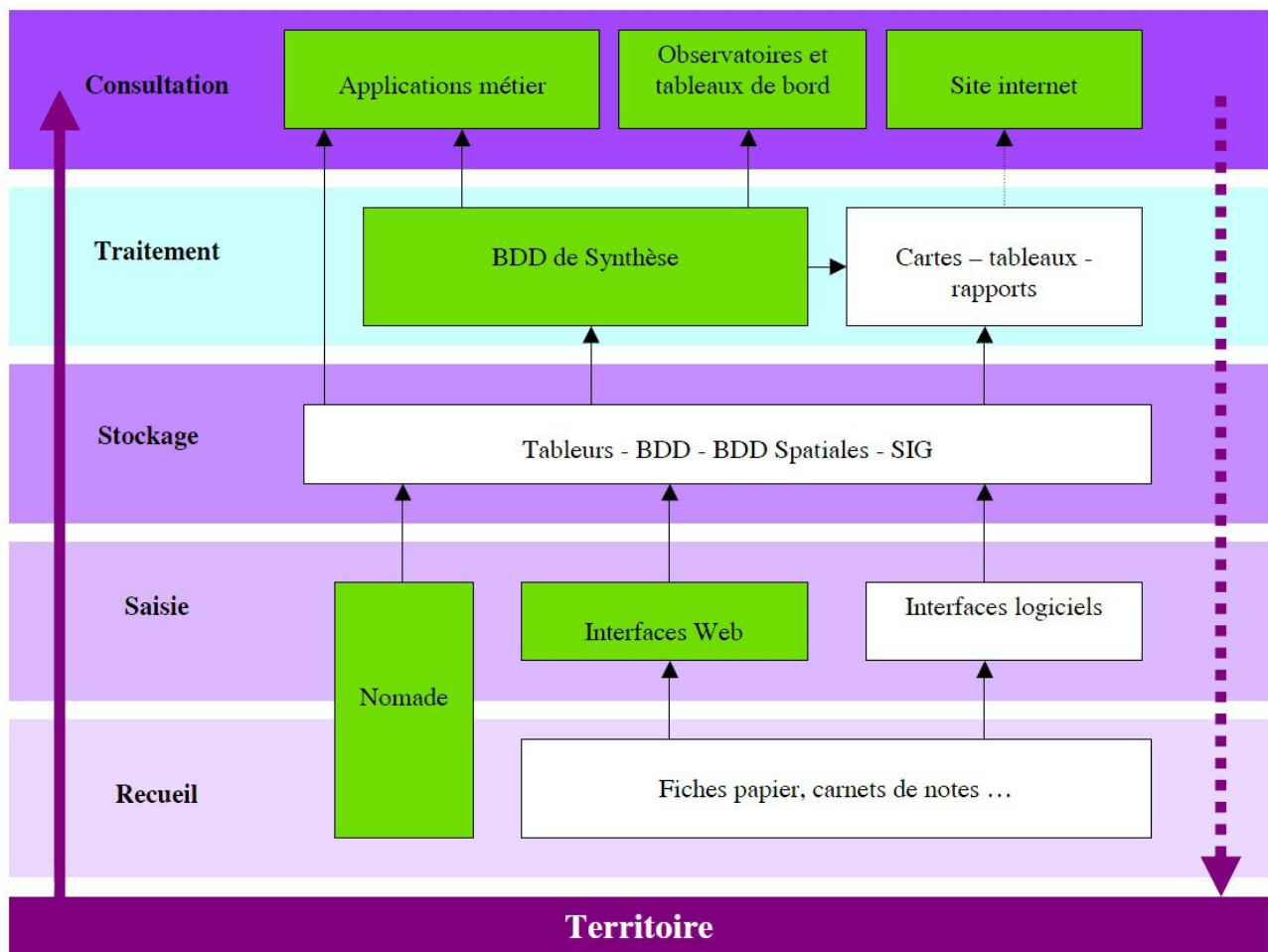


Figure 2: La chaîne de traitement du SI : du recueil à la consultation des données (document interne). On distingue en vert la chaîne de traitement actuelle, et en blanc l'ancienne.

2. L'atlas faune – flore du Parc national des Écrins : cahier des charges et architecture

■ Historique et contexte du projet de l'atlas faune-flore

Fort d'une base de plusieurs milliers de données naturalistes récoltées quotidiennement par les agents, le PNE souhaite aujourd'hui développer une application pour diffuser et partager ses connaissances en terme de faune et de flore.

Depuis 1973 le parc réalise un « inventaire permanent » de la faune et de la flore. En plus des protocoles spécifiques de suivi de certaines espèces, les agents notent quotidiennement les espèces qu'ils observent sur le terrain. Le parc dispose ainsi d'environ 360 000 données d'observation faune et 200 000 de flore représentant en tout plus de 5000 espèces différentes observées dans les Écrins. Historiquement les agents récoltaient manuellement ces données. Depuis quelques années ils sont équipés d'un smartphone et d'applications qui leur permettent de noter toutes les observations qu'ils effectuent. Ces applications alimentent au jour le jour différentes bases de données PostgreSQL.

L'architecture est construite autour du principe : un protocole = un outil = une BDD + BDD de SYNTHESE regroupant les données des différents protocoles sur la base des champs communs à tous les protocoles (QUI a vu QUOI, OU et QUAND) (figure 3).

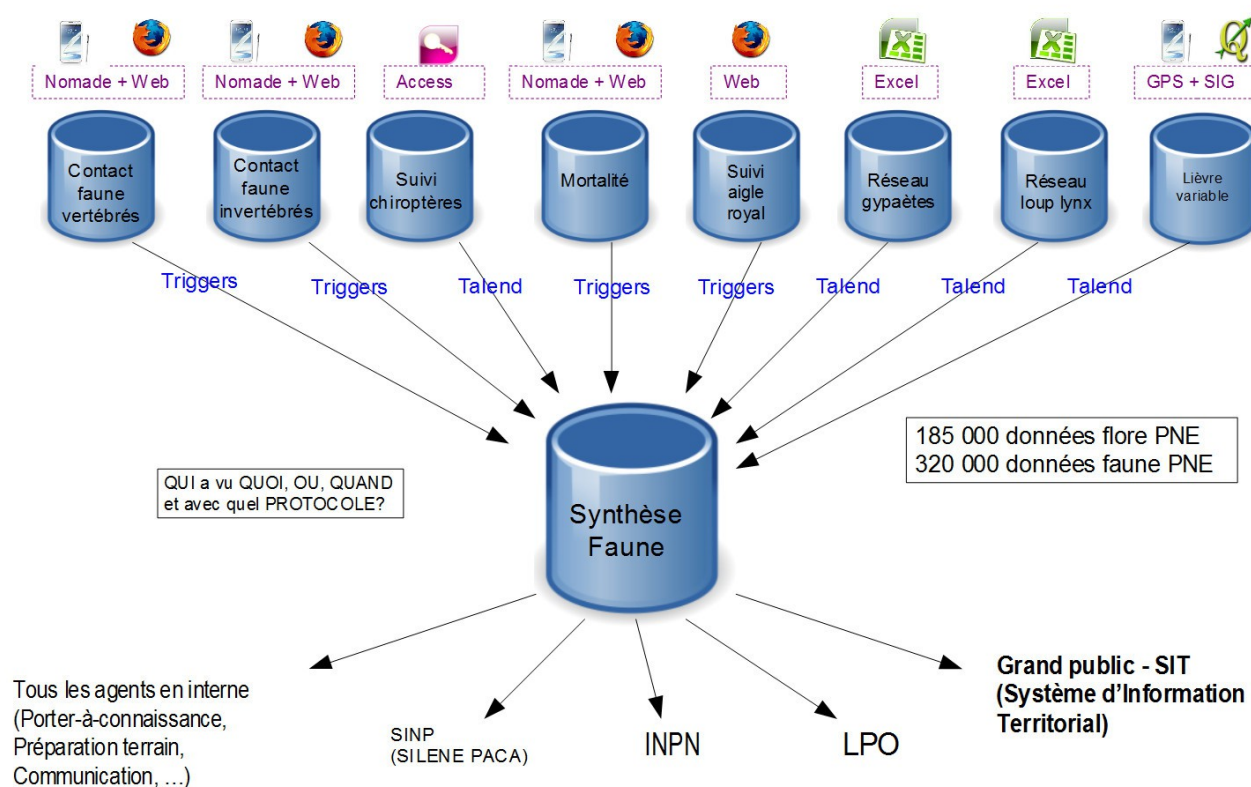
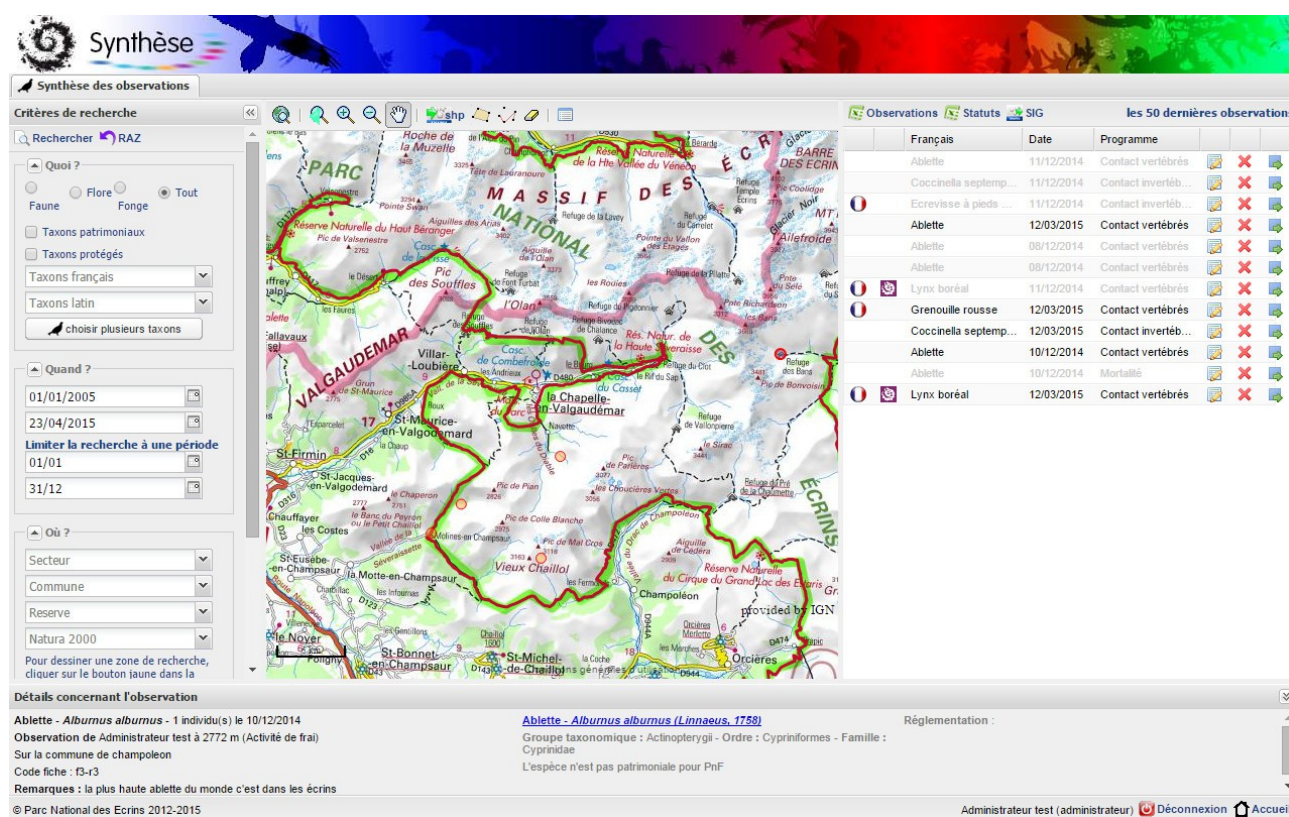


Figure 3: Illustration 2: Architecture de la BDD faune du PNE (document interne)

Chaque protocole de suivi faune-flore dispose de son propre outil de stockage des données (base postgresQL, access, excel etc.). En amont se situent deux bases : UsersHub (pour la gestion centralisée des utilisateurs) et TaxHub qui gère les caractéristiques des espèces présentes dans le parc. Par des triggers (action déclenchée en BDD), ou des outils Talend (outil de traitement automatisé des données), l'ensemble des données sont regroupées dans une grande BDD « synthèse faune-flore ». Cette base centralisée sert de point de départ aux applications métiers et grand public du parc et à la diffusion vers les partenaires.

Une application existante en interne appelée « GeoNature » (<https://github.com/PnEcrins/GeoNature>) se rapproche quelque peu de l'atlas que le parc veut mettre en place. Elle permet de visualiser toutes les données de la base «synthèse faune-flore ». L'application, qui est un Web-SIG, permet de savoir où a été vue une espèce à un moment X, puis d'exporter ces résultats sous différents formats (format shapefile, .kmz, tableaux etc...) C'est un outil intéressant pour l'étude du territoire et de son milieu naturel : les agents peuvent en effet à tout moment exporter les données qui les intéressent pour rédiger des rapports, faire des portées à connaissance auprès d'élus ou du grand public. Cependant, cette application est actuellement accessible simplement aux agents du parc et paraît peu adaptée pour être diffusée telle quel au grand public (figure 3).



L'application GeoNature a été développée entre 2012 et 2015 par le SI. Elle fait la synthèse de toutes les observations faune-flore effectuées dans le PNE. Les agents peuvent réaliser des recherches en filtrant par espèce, commune, date etc. Les données issues de l'application peuvent être exportées par les agents ce qui en fait un outil puissant d'aide à la décision. L'application a été déployée dans les parcs nationaux du Mercantour, de la Vanoise et de Guyane.

■ Données naturalistes et dynamiques de l'open-data

Longtemps certaines données faune-flore étaient considérées comme sensibles et les naturalistes n'étaient pas toujours enclin à les partager ou les rendre publiques. Aujourd'hui, la mouvance de l'open-data a

quelque peu fait changer les mentalités et une dynamique inverse s'opère. Cette dynamique est actuellement renforcée par des contraintes réglementaires : des directives (notamment Inspire) obligent les organismes publics à diffuser leurs données et fournir des catalogues de métadonnées.

Les données du parc national sont actuellement disponibles sur différentes plates-formes internet notamment SILENE (Système d'Information et de Localisation des Espèces Naturels), déclinaison en région PACA du SINP (<http://flore.silene.eu/index.php?cont=application&event=init>). Sur ce site, les utilisateurs peuvent rechercher une espèce (faune ou flore) et observer où celle-ci a été vue dans la région PACA (figure 4). Les données sont également accessibles au téléchargement sur demande pour les bureaux d'étude, les étudiants ou encore les chercheurs.



Figure 4: Capture d'écran de l'application SILENE

Cet outil permet au parc de répondre à ses obligations législatives mais a néanmoins quelques limites :

- la visibilité du parc et de son travail apparaît très faible et ses données sont noyées autour de centaines d'autres.
- les données sont agrégées par maille et sont peu précises.
- l'outil est clairement orienté pour les spécialistes ou les professionnels et est peu compréhensible pour le grand public.

Le parc souhaite donc aujourd'hui développer une application qui mette réellement en lumière le patrimoine naturel du territoire des Écrins et que ce contenu soit accessible au grand public.

■ Contenu de l'atlas

A mon arrivée en stage, le contenu global de l'application avait déjà été discuté. L'objectif est de réaliser un « atlas » de la faune et de la flore du PNE. L'atlas comporterait une fiche descriptive par espèce présente dans le parc. Chaque fiche serait composée de contenu généré automatiquement depuis la base de donnée «synthèse faune-flore» et du contenu complété manuellement espèce par espèce, par les spécialistes et les agents du parc.

Voici les contenus prévus à mon arrivée :

Contenu généré automatiquement pour chaque espèce :

- Nom de l'espèce et taxonomie¹ (famille de l'espèce)
- Carte de répartition des observations
- Graphiques altitudinaux
- Répartitions communales
- Répartitions par massif/secteur/unité géographique
- Graphiques de phénologies (période d'observations)
- Statuts de protection de l'espèce
- Observateurs

Infos saisies manuellement sur chaque espèce:

- Description de l'espèce
- Commentaire libre agent
- Répartition / Distribution (régions)
- Habitats/Milieus (liste prédéfinie)
- Photo de l'espèce

Ces informations seront saisies à partir d'une application interne du parc: TaxHub (<https://github.com/PnX-SI/TaxHub>). Cette application permet de gérer pour chaque taxon (espèce) tout une série de renseignement, tel que le statut de protection, la patrimonialité, une description etc. Elle s'appuie sur une base de donnée nationale de référence nommée Taxref (<https://inpn.mnhn.fr/programme/referentiel-taxonomique-taxref>) administrée par le Muséum d'Histoire Naturel, qui centralise toutes les espèces existantes sur le territoire français. Dans la base, chaque espèce est identifiée par une clé unique : le « cd_nom », ce qui permet de grandement faciliter l'échange des données naturalistes entres les structures.

Enfin, une page d'accueil comportera elle du contenu dynamique sur l'actualité des observations de la faune et de la flore: par exemple les 10 dernières espèces observées, les espèces les plus observées ou encore

¹ La taxonomie est la science de la classification hiérarchique des être vivants

les espèces patrimoniales etc... On y retrouvera une entrée « grand public » (géographique) ou les utilisateurs pourront rechercher les espèces par commune, ou par secteur du parc dans lequel ils se trouvent, une entrée scientifique avec une recherche par la taxonomie (nom latin, genre, groupe de l'espèce), et une entrée par l'actualité des prospections naturalistes (dernières espèces observés dans le parc etc).

Malgré cette idée assez claire du contenu de l'atlas, plusieurs points faisaient encore débat et nécessitaient d'être discutés avec les thématiciens faune et flore et avec le service communication. Cette partie du travail sera abordée dans le paragraphe « gestion du projet » de la troisième partie.

En plus de ces aspects thématiques, le PNE avait également des attentes techniques sur le développement de l'atlas. Le parc, s'est engagé depuis le début de ses développements dans une démarche open-source. Les diverses applications développées sont amenées à être déployées dans d'autres parcs naturels ou dans d'autres structures dont les besoins pourraient être similaires. Chaque application est donc développée dans un souci de généricité afin qu'elles puissent être facilement partagée. Un effort important est fait sur la documentation de l'installation et du code, et la totalité des codes-sources sont publiés sous licence libre sur Github (<https://github.com/PnEcrins/> et <https://github.com/PnX-SI>).

L'ensemble de ces points ont donc dû être pris en compte lors du développement.

3. Travaux réalisés

3.1 Gestion de projet et définition du contenu de l'atlas

▪ Revue des atlas faune-flore existant

Durant les premières semaines du stage, une partie du travail a consisté à faire une revue des applications et ouvrages existants pouvant se rapprocher de l'atlas souhaitant être développé. En analysant les différents contenus et fonctionnalités, nous avons tenté de faire une synthèse afin de mieux définir nos besoins. Nous avons également essayé d'identifier les points forts et faibles de ces outils et les points sur lesquels notre atlas pourrait se démarquer.

Par des recherches sur internet et dans les ouvrages du parc, nous avons identifié une dizaine d'applications ou livres se rapprochant d'un atlas faune-flore. Pour chacun d'entre-eux j'ai identifié et analysé plusieurs points :

- le public cible (grand public, scientifique)
- Le contenu des fiches descriptives des espèces (graphique, texte descriptif, photographie etc..)
- la présence d'une cartographie (si oui, avec quelle échelle et quelle précision des observations)
- la possibilité de télécharger les données
- le design

■ Réunions de coordination du projet

Le travail préparatoire décrit ci-dessus a permis de mieux aborder les futures réunions de définition du projet en relation avec toutes les parties prenantes, à savoir le pôle connaissance du service scientifique (notamment les thématiciens faune et flore), le service communication, et le pôle système d'informations.

Lors de ces réunions plusieurs points encore en suspens concernant l'atlas ont été discutés.

Un des points importants portait sur la définition du public cible de notre atlas. Bien qu'il était acté que l'application devait être accessible pour le grand public, il était important de définir clairement quel serait les cibles de l'atlas afin d'en adapter le contenu et la maquette.

Il a été décidé que l'atlas se devait de toucher deux grandes catégories de personnes : le grand public (touristes, curieux ou habitants du territoire) et les naturalistes (scientifiques, étudiants, curieux sur une thématique). Ce choix nous a donc amené à définir deux entrées distinctes pour l'atlas : l'entrée « grand public » se verra géographique et didactique : recherche des espèces par la commune, le secteur du parc, panel des dernières observations, afin que l'utilisateur puisse visualiser les espèces qui se trouvent autour de lui ; et une entrée par la taxonomie : recherche par le nom scientifique ou l'ordre, la famille du taxon.

La sensibilité des données était également un point important abordé lors de ces réunions. La question était de savoir si le parc ouvrait ou non au grand public la localisation exacte de toutes les observations faites par les agents du parc, notamment dans le cas d'espèces rares ou protégées. Dans les différents atlas investigués, les observations sont dégradées à l'échelle de mailles (de 500 mètre à un kilomètre de côté).

Le choix du parc a finalement été celui d'impulser une dynamique d'ouverture des données : chaque observation d'espèce pourrait donc être localisée sur une carte. Dans un souci de généricité, une option d'affichage par maille sera également développée pour permettre à d'autres structures de faire ses propres choix d'affichage des données.

Enfin, d'autres points ont été discutés, notamment sur le contenu exact de chaque fiche espèce, la possibilité de télécharger les données d'observation, ou encore le pilotage du projet en lien avec les autres services impliqués.

Une réunion finale qui s'est déroulée le 6 juin (voir annexe 3 du compte-rendu de la réunion) a permis de fixer les dernières orientations et de commencer le développement.

3.2 Dimension technique du travail

3.2.1 Mise en place de l'environnement de travail

L'ensemble des bases de données ainsi que des environnements de production et de développement sont

hébergés sur des serveurs virtualisés chez OVH sur deux serveurs dédiés. La première tâche a donc été d'installer mon environnement de travail. J'ai dans un premier temps créé une machine virtuelle de développement sous Debian 8 sur les serveurs d'OVH. J'ai installé Win-SCP et Putty (des clients SSH respectivement graphique et terminal) pour travailler sur mon environnement de développement.

3.2.2 Découverte des applications et des bases de données du parc national

Durant la première semaine je me suis approprié les différentes applications du parc national. J'ai installé sur mon environnement de développement les applications « GeoSites » (<https://github.com/PnEcrins/GeoSites>) (un outil de visualisation cartographique du patrimoine géologique) et « GeoNature » (présenté plus haut), afin de pouvoir m'approprier le code. Le côté back-office de ces deux applications est codé en PHP avec le framework Symfony. L'idée de départ de l'équipe était de faire le développement de l'atlas sur Symfony et AngularJS. Je me suis donc formé la première semaine à Symfony tout en continuant à me documenter sur AngularJS que je connaissais.

N'ayant jamais utilisé de framework « full-stack » tel que Symfony, j'ai dû apprendre les bases de son fonctionnement. Je me suis familiarisé avec la structure et les différentes composantes de celui-ci. J'ai principalement découvert trois composantes importantes du développement web que je n'avais pas vu cette année (<http://documentation-symfony.fr/>):

- **Le fonctionnement d'un ORM** (mapping object-relationnel). Les ORM permettent de créer des classes (PHP en occurrence) à partir de base de données puis d'interroger ou modifier directement sa base en mode objet à partir d'un langage de requête propre à l'ORM. Symfony utilise l'ORM Doctrine qui possède son « Query Language » PHP appelé Doctrine Query Language, ce qui permet de ne pas (ou peu) utiliser le langage SQL dans le développement. Dans Symfony, chaque table de la base donnée possède sa propre « entity » et son « repository ». Dans l'« entity » sont définis tous les attributs de la classe (qui correspondent aux champs de la table) ainsi que les « getter » et « setter » pour accéder ou modifier à ces attributs. Le « repository » contient les interrogations à la base de donnée écrit en PHP grâce au « query language » de Doctrine.

- **Le routing** : dans Symfony chaque « URL », ou route, est associé à un contrôleur et à une action. Les contrôleurs sont également des classes PHP et les actions sont des méthodes de cette classe. Pour chaque route, on définit le contrôleur qui va être appelé, ainsi que l'action qui va être déclenchée. Une action va généralement aller interroger une base de données pour ensuite retourner des fichiers (XML, Json) dans le cas d'une application REST, ou alors un template (une page HTML).

- **Les générateurs de templates** : Symfony dispose d'un moteur de template appelé Twig, qui permet de générer des pages HTML dynamiquement en fonction du contrôleur et de l'action appelée. Une action qui

retourne une page HTML, peut également envoyer à cette page des variables qui sont préalablement calculées depuis la base de données par exemple. Twig permet de pouvoir afficher et traiter ces variables directement dans la page HTML, un peu à la manière du JavaScript.

Bien que nous ayons par la suite décidé de nous éloigner de PHP, et donc de Symfony, la découverte de ces trois concepts propre aux architectures de type modèle-vue-contrôleur (MVC) qui m'ont été fortement utiles par la suite.

J'ai me suis parallèlement intéressé à la manipulation des bases de données du parc national sur le SGBD PostgreSQL (assimilation des modèles conceptuels de données des BDD et appropriation de PostgreSQL et de PostGIS) et à la compréhension des différents protocoles faune-flore de la structure. Le parc national possède en effet une architecture de BDD assez complexe qui manipule des concepts naturalistes. Je me suis notamment familiarisé avec la taxonomie, concept qui sera fortement utilisé dans l'atlas.

3.2.3 Choix de l'architecture web mise en place

A mon arrivée en stage, le choix des technologies pour le développement de l'atlas n'était pas arrêté. On m'a demandé de réfléchir aux technologies et à l'architecture adaptée aux problématiques de ce projet.

La plupart des applications développées par le SI utilisent PHP et Symfony pour le back-office et AngularJS pour le front-end. Cependant en développant récemment l'application « GeoSites », le parc n'était pas tout à fait satisfait du référencement (SEO) par les moteurs de recherche. Les sites développés avec AngularJS (de manière générale avec un framework JS), où l'intégralité des templates sont générés en Javascript côté client pose en effet des problèmes de référencement. Lorsque l'on ouvre le débogueur de son navigateur, on s'aperçoit en effet qu'en dehors des balises html, le contenu de la page est presque vide. Les moteurs de recherche ne peuvent donc pas « crawler » et donc référencer le résultat de ces pages. De plus, les applications développées avec AngularJS tendent à faire des « Single Page App » (SPA) davantage typé « application » que site web, ce qui paraît davantage adapté pour les besoins métiers et moins pour le grand public.

Je me suis donc intéressé aux solutions permettant de palier ce problème de référencement.

Plusieurs solutions existent :

- le polymorphisme : le back-end et le front-end sont tous les deux écrits en JavaScript, ce qui permet de générer du contenu HTML à la fois côté serveur que côté client (le framework ReactJs permet notamment cette pratique).

Appliqué à notre projet, l'architecture pourrait donc être la suivante : (figure 5). Le framework Javascript est

présent la fois côté serveur et côté client. La BDD est interrogée grâce à un ORM. On peut ensuite générer directement du contenu HTML à partir des résultats de la base côté serveur ou alors construire une API qui sera interrogée côté client. Le framework client génère ensuite les templates HTML.

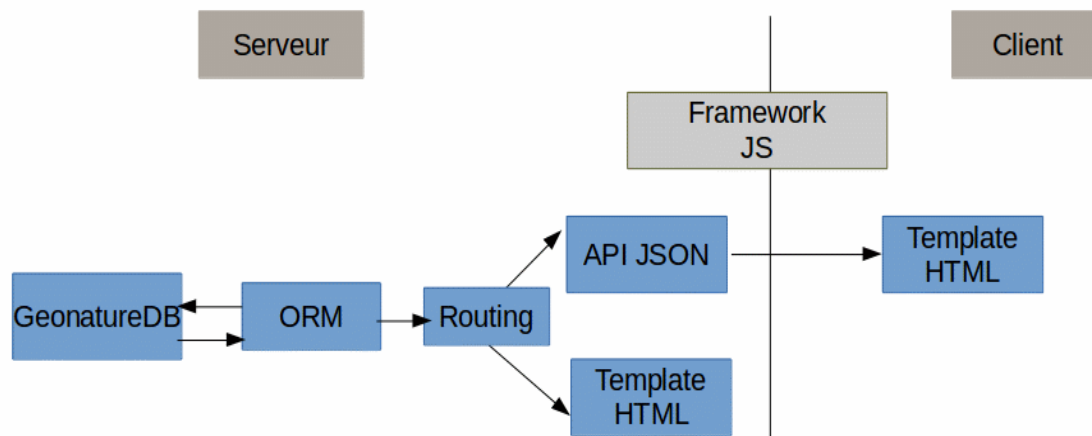


Figure 5: Schéma d'une application web isomorphe

- La seconde solution consiste à générer l'ensemble des templates côté serveur en utilisant un framework web full-stack (Symfony, Django), sans framework lourd côté client (figure 7)

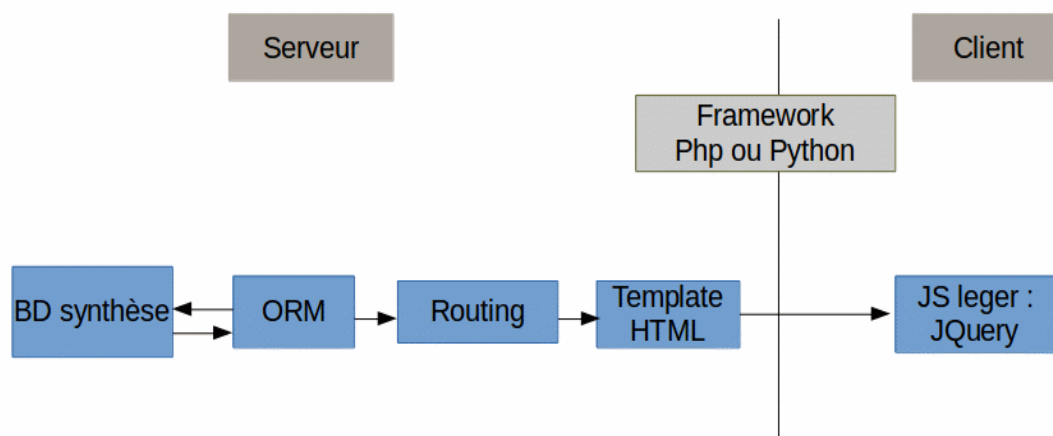


Figure 6: Schéma d'une application génération des templates côté serveur

Concernant le développement, le PNE travaille beaucoup avec les autres parc nationaux français. Comme mentionné plus haut, la logique open-source des parcs pousse à la mutualisation. Le choix d'une nouvelle

technologie est donc souvent discuté avec les équipes SI des autres Parcs nationaux. Nous les avons donc consultés afin de prendre une décision sur l'architecture des technologies de l'atlas. Après concertation et présentation de nos problématiques, nous avons décidé de partir sur une solution de génération de template côté serveur (voir annexe 3 : compte-rendu de la réunion SI inter-parc). Le choix de se former sur des framework relativement compliqués tel que ReactJS nous paraissait ambitieux et lourd. Nous avons également fait le choix de nous tourner vers Python pour plusieurs raisons :

- la souplesse et la simplicité de ce langage
- c'est le langage préférentiel de la communauté open-source SIG : ils se prête donc très bien aux problématiques spatiales (de nombreuses librairies spatiales sont disponibles, de plus l'ensemble des plug-in QGIS sont codés en Python).

Ces deux points ont fait pencher la balance en faveur de Python, collant bien aux problématiques métiers des parcs. Concernant, le framework nous avons fait le choix d'une solution légère : le micro-framework Flask, qui contrairement à Django a le mérite d'être souple et plus adapté au petit projet. 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-parc.

Pour finir, la figure 7 illustre l'ensemble des technologies utilisées pour le développement de l'atlas. Pour le côté back-end, les bases sont gérées avec PostgreSQL et PostGIS et interrogées grâce à l'ORM SQLAlchemy. Flask permettra de faire le routing ainsi que la génération de template grâce au module Jinja2. Côté front-end, nous utiliserons JQuery pour la gestion du DOM et les animations. Les librairies Leaflet et MorrisJS, serviront respectivement à la génération de la cartographie et des graphiques. Enfin, le framework Bootstrap gèrera le CSS.



Figure 7: Illustrations des technologies utilisées

3.2.4 La partie base de données

Finalement, nous avons réfléchi à l'architecture de la base de données.

L'outil GeoNature possède déjà une base qui correspond parfaitement au besoin de l'atlas : elle est construite en partie à partir des données « Synthèse-Faune-Flore », et d'une base de donnée « taxonomie » qui renseigne les caractéristiques de chaque espèce (basé sur le référentiel national Taxref). Cette base de données, appelée « geonaturedb » embarque cependant toute une série de tables qui ne sont pas nécessaires pour l'atlas (table de chaque protocole, table de gestion des utilisateurs etc.).

Notre objectif pour le développement de l'atlas était que n'importe quelle structure qui dispose d'une base de données des observations de ses espèces ainsi qu'une base des caractéristiques de ses taxons puisse installer l'application.

Ne voulant pas se connecter directement à la base de production de GeoNature ni créer deux bases jumelles ayant les mêmes fonctions, nous avons donc choisis d'utiliser le mécanisme des « foreign-data-wrapper » (FDW) (voir figure 8). Ce mécanisme consiste à créer une base miroir (base fille) qui se connecte à une base mère, la première pouvant se mettre à jour suivant les modifications de la base mère. La base fille ne contient elle aucune donnée : c'est simplement un miroir de la base mère.

Les foreign-data-wrapper ont notamment une utilité en terme de sécurité. On peut régler les permissions

d'accès à la base mère depuis la base «fille». Dans le cas de l'atlas, aucun besoin en écriture ne sont nécessaire, celui-ci à seulement besoin de récupérer les données. Notre base fille a donc seulement des droit en lecture sur « geonaturedb ».

Pour requêter sur des grosses bases de données, il est souvent utile de créer des « vues » qui permettent de remonter des résultats pré-calculées ou simplement de remonter qu'une partie de la base. Nous avons donc testé les performances des vues avec les FDW. Nous nous sommes rendus compte que celles-ci étaient largement moins bonnes avec les FDW. La solution trouvée a donc été de créer des vues matérialisées. Les vues matérialisées, reprennent le mécanisme des vues, sauf qu'à la différence de ces dernières, les données sont stockées en dur dans la base fille. Ce mécanisme nous permet donc de simuler un « cache » des données. En effet, toutes les nuits (grâce à un mécanisme linux de « cron »), un « refresh » est fait sur les vues matérialisées sont ré-calculées en interrogeant la base mère via la base fille : ce mécanisme permet de toujours avoir des données à jour qui sont stockées en dur dans la base de l'atlas.

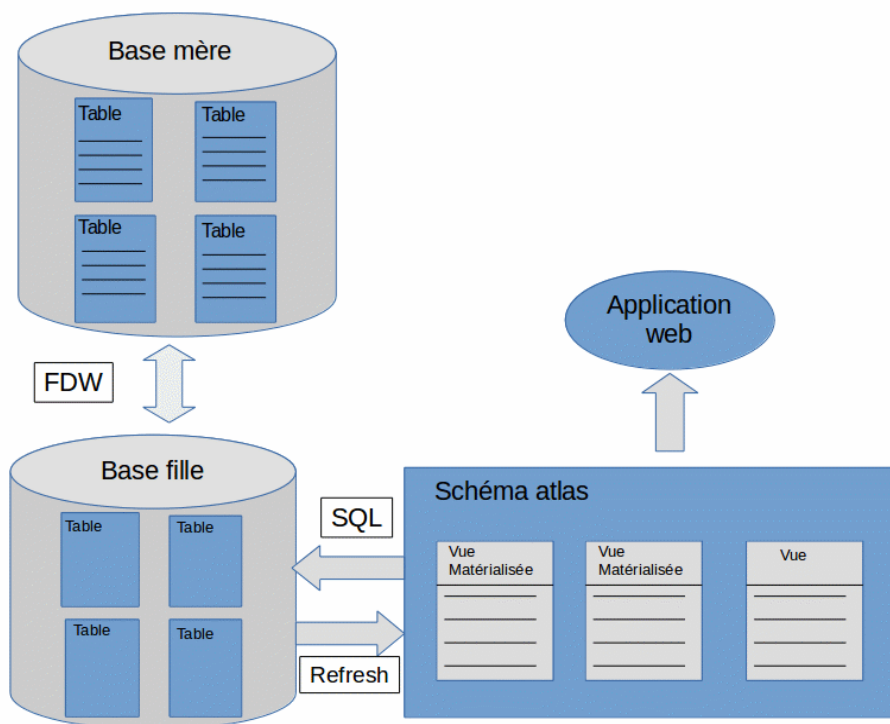


Figure 8: Schéma du mécanisme "foreign data wrapper" mis en place. La base mère est clonée en une base fille qui ne contient pas directement les données. Des vues matérialisées permettent de stocker en dur dans un schéma nommé « atlas », les données nécessaires . Chaque nuit, un mécanisme de « refresh » recrée le FDW et les vues matérialisées avec les nouvelles données de la base mère.

Toutes les réflexions menées durant le premier mois de stage ont donc permis d'avoir une vision plus claire

des possibilités liées au projet de l'atlas, aussi bien en terme de contenu thématique que concernant la partie technique du développement web. Les réunions thématiques ont permis de bien cerner les différentes attentes du service scientifique et les réunions techniques ont permis de faire des choix marquant en terme de technologies. A la lumière de ces réflexions, nous nous sommes fixé comme objectif de livrer à la fin de mon stage une première version de l'atlas contenant : une fiche par espèce (fiche d'identité, cartographie des observations, graphiques descriptifs), une page d'accueil à la fois didactique et technique permettant de percevoir l'actualité des observations naturalistes du parc, et une page par commune du parc (liste des espèces, cartographie, illustrations).

3.4 La phase de développement

■ Planning et gestion du développement

La phase de développement de l'atlas a commencé la semaine de 30 mai. Durant cette phase nous avons travaillé à deux avec le chargé de mission développement Web et base de données : lui s'occupant davantage de la partie back-office (base de données, génération des vues, gestion des performances) et moi de la partie front-end et Flask (interrogations des vues grâce à l'ORM, génération des routes, JavaScript, HTML et CSS).

Une première étape a été une phase de formation au développement web sous Python ainsi qu'une familiarisation avec le framework Flask (<http://flask.pocoo.org/docs/0.11/>). L'objectif était dans un premier temps d'installer une application Flask simple sur mon serveur de développement: configuration du serveur Apache, connexion et interrogation de la base de données, génération simple d'une route et d'un template HTML.

Bien que les technologies utilisées soit nouvelles, le travail de documentation réalisé sur Symfony m'a permis d'avancer assez rapidement, en effet les concepts d'ORM, de routing et de génération de templates sont relativement similaires.

Concernant la stratégie de développement, nous avons travaillé avec les méthodes agiles : à savoir, une livraison régulière (toutes les semaines ou deux semaines) d'un rendu aux parties prenantes. Nous avons construit l'application brique par brique en tenant régulièrement en compte des remarques et observations faites par les thématiciens.

J'ai réalisé au début du développement un diagramme de Gant planifiant les différentes étapes (figure 9).

Diagramme gant - developpement

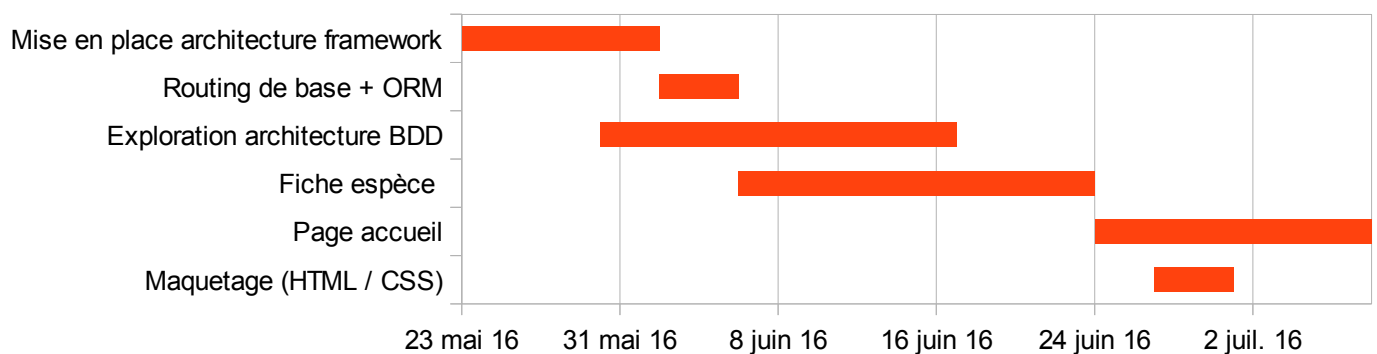


Figure 9: Diagramme de gant de la partie développement

Dans un premier temps, nous nous sommes concentré sur la « fiche espèce » : génération d'un outil de recherche des espèces par le nom latin et le nom français, affichage d'une carte des observations, affichage de graphiques des altitudes et des dates d'observations. Dans un second temps, le planning prévoit de développer la page d'accueil de l'atlas : cartographie des dernières observations, développement d'un outil de recherche taxonomique avancé etc.

Nous avons également utilisé GitHub comme outil de gestion et de suivi du projet. L'ensemble du code est régulièrement poussé sur la plate-forme GitHub, ce qui permet de faciliter le travail collaboratif. Chaque développeur possède en effet son propre serveur de développement. Cet outil participe notamment à la montée en compétence: chaque « commit » peut être visualisé par l'ensemble de l'équipe, tout le monde profite ainsi des avancées et des compétences des autres.

Le responsable du service ainsi que mon collègue de développement ont également rédigé sur cet outil des « tickets » décrivant les différentes tâches à réaliser. Pour chacun des ces tickets, nous pouvons ainsi discuter des éventuels problèmes ou bugs, suivre les « commits » relatif à cette tâche, puis finalement fermer le ticket lorsque la tâche est réalisée (<https://github.com/PnEcrins/GeoNature-atlas/issues/8>).

- **Architecture et structuration du code**

A l'installation de Flask, le framework ne livre pas une arborescence de dossier déjà préparé. Chacun est libre de structurer son code comme il l'entend. M'étant documenter sur Symfony, j'ai choisi de reproduire son architecture en adoptant un « pattern MVC » (Modèle-Vue-Contrôleur). Cette structure de séparation du code permet en effet de développer de manière claire et structurée, tout en facilitant la maintenabilité de celui-ci.

- La partie modèle :

Flask laisse le choix au développeur d'utiliser ou non un ORM, celui-ci n'est pas fourni avec le framework. Nous avons choisi d'utiliser SQLAlchemy (<http://www.sqlalchemy.org/>) qui est l'ORM Python conseillé avec l'utilisation de Flask.

La partie modèle concerne toutes les questions liées à la base de données (la logique métier). On y retrouve dans notre projet, deux dossiers différents :

- le dossier « entités » qui comporte des classes représentant les tables de la base de données. Le nom de la classe est le nom de la table ou de la vue de la base, et les attributs de la classe sont les champs des tables, on appelle ce principe le « mapping » (figure 10). Dans notre projet, Chaque entité est segmentée dans un fichier séparé.

```
class VmObservations(Base):
    __table__ = Table(
        'vm_observations', metadata,
        Column('id_synthese', Integer, primary_key=True, unique=True),
        Column('id_source', Integer),
        Column('id_fiche_source', String(50)),
        Column('code_fiche_source', String(50)),
        Column('id_protocole', Integer),
        Column('id_precision', Integer),
        Column('insee', String(5), index=True),
        Column('dateobs', Date, index=True),
        Column('observateurs', String(255)),
        Column('determineur', String(255)),
        Column('altitude_retenue', Integer, index=True),
        Column('remarques', Text),
        Column('date_insert', DateTime),
        Column('date_update', DateTime),
        Column('derniere_action', String(1)),
        Column('the_geom_point', Geometry, index=True),
        Column('id_lot', Integer),
        Column('id_critere_synthese', Integer),
        Column('effectif_total', Integer),
        Column('cd_ref', Integer, index=True),
        Column('geojson_point', Text),
        schema='atlas', autoload=True, autoload_with=engine
    )
```

Figure 10: Exemple de mapping de la vue vmObservation

Pour générer automatiquement les entités, nous avons utilisé un générateur de code appelé « sqlacodegen » (<https://pypi.python.org/pypi/sqlacodegen>). Celui-ci génère automatiquement les clés primaires, les clés étrangères et le type de chaque champs d'une table.

- les « repositories » : chaque entité possède son « repository » dans lequel on retrouve les interrogations à la base. Chaque interrogation est défini par une fonction qui renvoie un objet ou un tableau d'objets contenant les informations requêtes. La convention de nommage est la suivante : nom de l'entité + « repository ». Exemple : VmtaxonRepository, VbobservationsRepository.

- La partie contrôleur :

Il s'agit ici de la génération des routes (des URL) de l'application (figure 11). Chaque URL est associée à une fonction. Sur la figure 11, la fonction « index » est associée à la route « / » : la racine du projet. Cette fonction va faire appel aux méthodes définies dans les « repositories » pour récupérer les informations dans la base. Cette fonction va ensuite retourner un template et lui passer l'ensemble des variables voulues.

```
@main.route('/', methods=['GET', 'POST'])
def index():
    listeTaxons = vmSearchTaxonRepository.listeTaxons()
    observations = vmObservationsRepository.lastObservations(100)
    return render_template('index.html', listeTaxons=listeTaxons,
                           observations=observations)
```

Figure 11: Exemple de routing avec Flask

Nous avons à l'heure actuelle un seul fichier (/main/atlasPNE.py) qui définit deux routes principales : une route pour la page d'accueil à la racine de notre serveur (« / »), et une route pour la fiche espèce (« /atlas/nomEspece »).

- La partie « vue »

On retrouve dans cette partie les templates HTML (dans le dossier « template »), et les fichiers JavaScript et CSS (dans le dossier « static »). Le moteur de template intégré dans Flask est nommé Jinja2 (<http://jinja.pocoo.org/>), il permet d'utiliser les variables passées dans le template puis des les gérer grâce des fonctionnalités similaire à JavaScript (boucle, condition, filtre etc.). Jinja2 permet aussi de factoriser des « briques » HTML qui sont souvent réutilisées.

Nous avons donc à l'heure actuelle deux templates principaux : un pour la page d'accueil et un pour la fiche espèce. Une brique HTML à part génère la bar de navigation est insérée dans chacune de ces pages. A terme d'autres briques génériques seront développées pour davantage factoriser les templates.

Le code JavaScript est lui segmenté par fonctionnalité. On dispose d'un fichier qui gère la logique de la cartographie, un pour gérer les graphiques etc...

Enfin, la partie « static » contient également toute les librairies JavaScript et CSS utilisées dans le projet (Leaflet, D3, MorrisJS, Bootstrap).

▪ Effort de généricité du code

Comme je l'ai déjà mentionné à plusieurs reprises, une place importante est accordée à la généricité dans le travail afin de pouvoir partager facilement l'application.

Dans le développement, cela se traduit par plusieurs mécanismes simples :

L'ensemble des briques qui sont spécifiques au PNE sont soit passé en paramètre soit mis dans des variables. Ces variables sont regroupées dans des fichiers de configuration qui seront livrés avec l'application.

On dispose à l'heure actuelle de deux fichiers de configuration : un pour la base de données qui contient l'ensemble des codes de connexions au serveur (mot de passes, IP, port etc.) ; et un pour la partie JavaScript et HTML: on y retrouve par exemple les configurations de la cartographie (choix des fonds de cartes, du nombre d'observations à afficher) et le texte HTML spécifique au PNE etc.

Concernant l'affichage CSS, il est prévu de livrer un fichier de configuration de base et un fichier de surcouchage pour que chaque structure puisse personnaliser le style de l'atlas.

Enfin, l'effort de généricité se traduit par la rédaction régulière de documentation. Une documentation (encore en construction : <https://github.com/PnEcrins/GeoNature-atlas/tree/master/docs>), d'installation complète sera publiée sur le gitHub de l'application.

▪ Aperçu de l'application

Bien que le maquettage final de l'application n'ait pas encore été discuté avec la chargée de design du parc, nous avons commencé à afficher et coder les fonctionnalités principales de l'atlas. Les considérations d'esthétisme, d'ergonomie et de design ne sont donc pas encore à prendre en compte à ce stade du développement.

-La page d'accueil : <http://188.165.118.87/atlas/>

Comme décidé en réunion de préparation, la page d'accueil se vaudra dynamique et didactique. Elle devra présenter une sorte « d'almanach » des observations faune-flore du parc.

Durant ce premier mois de développement, nous nous sommes surtout concentré sur la fiche espèce. La page d'accueil est donc davantage une démo, puisqu'à l'exception de la barre de recherche des espèces, le reste est statique et non connecté à la base de données.

➤ L'outil de recherche et l'autocomplétion

Dans la barre de navigation située en haut de l'application se trouve une barre de recherche. L'utilisateur peut ainsi rechercher une espèce, ce qui lance une fonction d'auto-complétion qui l'aide dans sa saisie. La recherche peut être faite par le nom latin et le nom français pour pouvoir être accessible de tous.

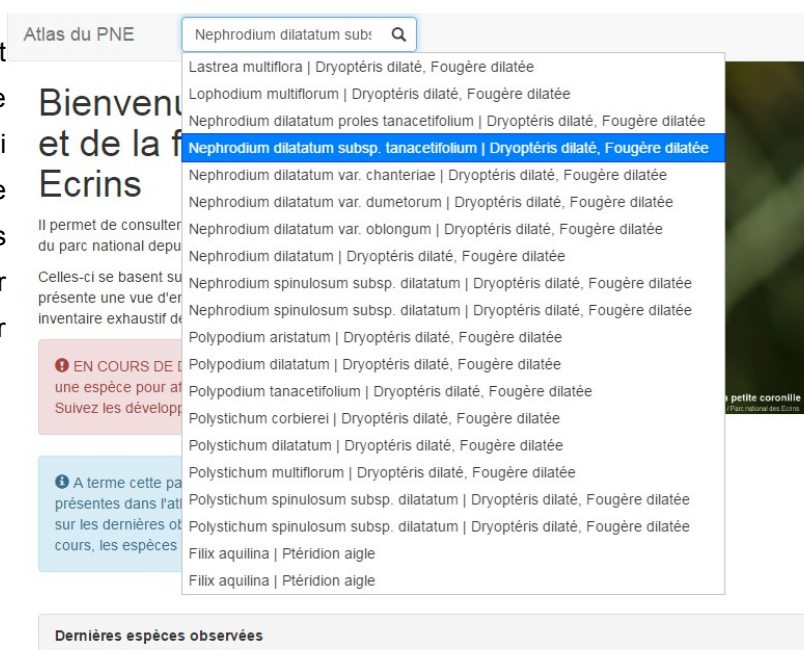



Figure 12: Le module de recherche avec autocomplétion

➤ Les parties statiques

L'atlas propose deux modules qui affichent les dernières espèces observées ainsi que les espèces les plus observées durant la période. En cliquant sur la photo ou sur « fiche détail », on est redirigé vers la fiche de l'espèce.


Dernières espèces observées



Saxifrage à feuilles opposées

Observé le 22 juin 2016 dans la commune XXX


Fiche détail



Lièvre variable

Observé le 22 juin 2016 dans la commune XXX


Fiche détail



Buxbaumie verte

Observé le 21 juin 2016 dans la commune XXX

Fiche détail





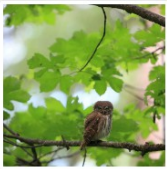



Hermine

Observé le 19 juin 2016 dans la commune XXX

Fiche détail

Espèces les plus observées à cette période














Figure 13: Les modules d'affichage des espèces en cours d'observations

- La fiche espèce



Figure 14: Fiche d'identité d'une espèce

Chaque fiche espèce contient d'abord une fiche d'identité. Elle renseigne :

- le nom (latin et français)
 - les informations sur la taxonomie
 - le nombre d'observation effectuée sur la parc
 - un lien extérieur renvoyant vers la fiche espèce du site de l'inventaire national du patrimoine naturel (<https://inpn.mnhn.fr/>)
 - Une photo (qui est pour l'instant statique)
- La carte des observations

Nous avons utilisé la librairie open-source Leaflet pour l'affichage de la cartographie des observations.

Pour chaque espèce, on génère depuis la vue matérialisée « vmObservation » un geoJson (format Json comportant des coordonnées géographiques), pour afficher un point par observation (figure 15).

Par défaut, l'affichage est fait sous forme de clusters (figure 15 gauche) : cette fonction permet de regrouper les observations et d'éviter la superposition. En cliquant sur un point le niveau de zoom augmente et une nouvelle segmentation en cluster est calculée. Ce mécanisme est fourni par un module additionnel de la

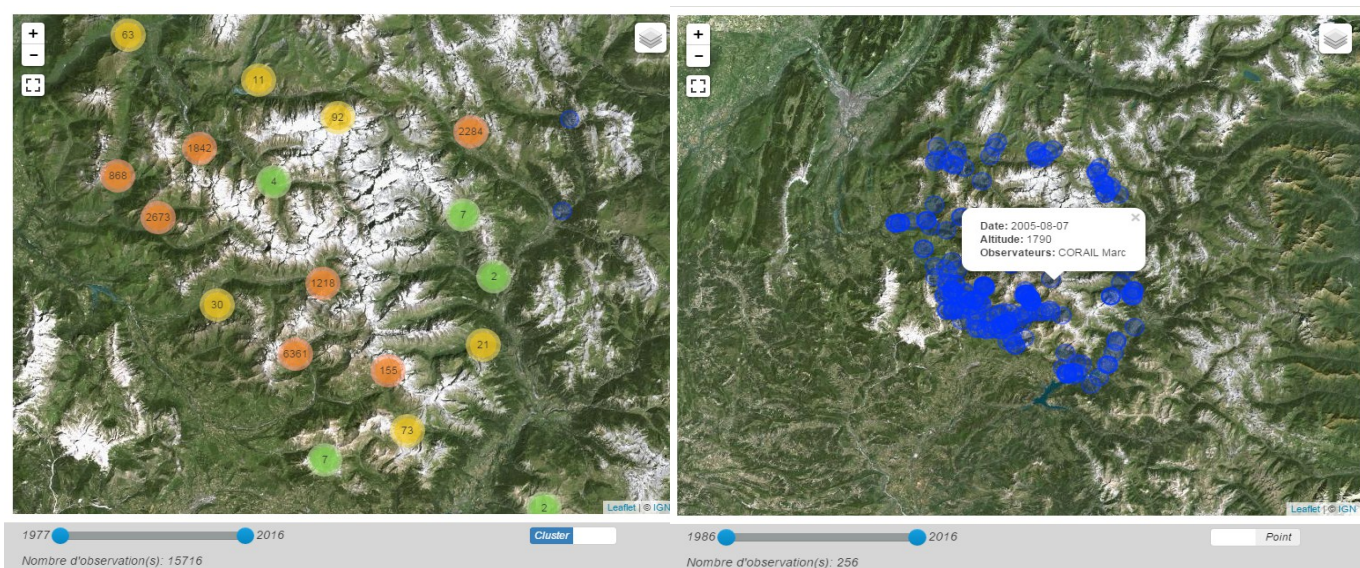


Figure 15: La carte des observation - A gauche en cluster - A droite: tous les points

librairie Leaflet.

L'utilisateur peut également choisir de désactiver le clustering pour avoir une vue d'ensemble des observations (un cercle bleu = une observation : figure 15, droite). Au clic, chaque point est associé à une popup qui renseigne sur la date d'observation, son altitude, le nom de l'agent qui l'a effectué et éventuellement l'effectif.

Le « slider » situé en bas de la carte permet de filtrer les observations par la date. Les dates minimum et maximum sont calculées dynamiquement (la date de l'observation la plus ancienne de l'espèce pour la date minimum et la date de l'année en cours pour la date maximum).

Enfin la carte possède des fonctionnalités annexes : on peut changer le fond de carte parmi trois options : le fond OpenStreetMap, la carte topographique IGN, et les orthophotographies (sur la figure 15). Une option permet également de passer en plein écran.

- les graphiques

La page de l'espèce contient à l'heure actuelle deux graphiques : un pour les altitudes d'observations, et un pour la mensualité.

Ces graphiques ont été construits avec la librairie MorrisJS et D3 qui permettent de générer des balises SVG dans la page HTML.

Concernant le graphique d'altitude, par souci de généralité, les classes d'altitudes qui sont amenées à être largement différentes en fonction des structures sont générées depuis la base de données. Aucun code, en « dur », ne mentionne les altitudes dans la partie JavaScript.

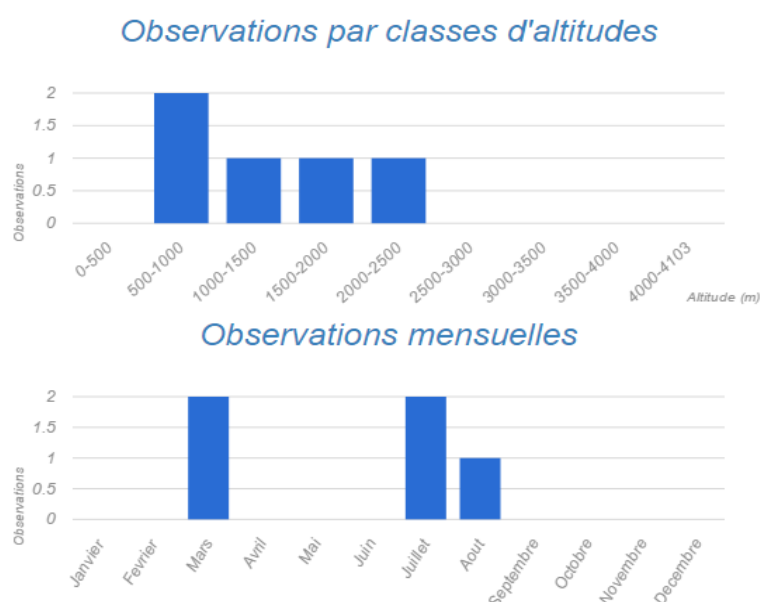


Figure 16: Graphiques des observation mensuels et altitudinales

Conclusion et perspectives

Plusieurs perspectives s'annoncent pour la suite du stage. L'objectif fixé est de livrer pour le 30 septembre une première version terminée de l'atlas. Plusieurs grands chantiers s'annoncent :

Le premier concerne la page d'accueil de l'atlas sur laquelle nous n'avons pour l'instant presque pas travaillé. Le second concerne la partie « manuelle » de l'atlas : à savoir l'affichage des photographies, de la description de l'espèce etc. L'outil TaxHub, en cours de développement servira de plate-forme de centralisation de ces informations. Un travail de long haleine s'annonce pour que progressivement, les agents renseignent ces informations pour chacune des 5000 espèces du parc. De notre côté nous devrons connecter l'outil TaxHub à l'atlas ; nous avons d'ailleurs récemment fait la modélisation de la base de données « taxonomie » de TaxHub qui accueillera toutes ces informations.

Un travail important de maquettage et d'ergonomie devra également être fait. Enfin, une phase de test devra être menée et ce à chaque grande fonctionnalité développée.

D'un point de vue plus personnel, j'ai acquis au cours de ces deux premiers mois de stage des compétences qui seront à consolider (développement web en Python et Flask et JavaScript, architecture web : MVC, REST, modélisation de base de données, connaissance naturalistes). J'aimerais durant les trois mois qui restent davantage toucher à la partie « base de données » de l'atlas : manipuler PostgreSQL et PostGIS pour avoir une vue d'ensemble de l'application que nous avons développée et ainsi avoir un profil plus généraliste.

Index des illustrations

Figure 1: Carte de localisation du PNE et de ses secteurs.....	4
Figure 2: La chaîne de traitement du SI : du recueil à la consultation des données (document interne). On distingue en vert la chaîne de traitement actuelle, et en blanc l'ancienne.....	5
Figure 3: Illustration 2: Architecture de la BDD faune du PNE (document interne).....	6
Figure 4: Capture d'écran de l'application SILENE.....	8
Figure 5: Schéma d'une application web isomorphe.....	14
Figure 6: Schéma d'une application génération des templates côté serveur.....	14
Figure 7: Illustrations des technologies utilisées.....	15
Figure 8: Schéma du mécanisme "foreign data wrapper" mis en place. La base mère est clonée en une base fille qui ne contient pas directement les données. Des vues matérialisées permettent de stocker en dur dans un schéma nommé « atlas », les données nécessaires . Chaque nuit, un mécanisme de « refresh » recrée le FDW et les vues matérialisées avec les nouvelles données de la base mère.....	17
Figure 9: Diagramme de gant de la partie développement.....	18
Figure 10: Exemple de mapping de la vue vmObservation.....	19
Figure 11: Exemple de mapping de la vue vmObservation.....	19
Figure 12: Exemple d'un routing Flask.....	20
Figure 13: Le module de recherche avec autocomplétion.....	21
Figure 14: Les modules d'affichage des espèces en cours d'observations.....	22
Figure 15: Fiche d'identité d'une espèce.....	23
Figure 16: La carte des observations - A gauche en cluster - A droite: tous les points.....	23
Figure 17: Graphiques des observations mensuels et altitudinales.....	24

Annexe 1 : Fiche de proposition de stage

CONTEXTE :

Le Parc national des Ecrins est un établissement public de l'Etat. Il comprend une centaine d'agents, répartis sur 8 sites : le siège à Gap (05), 2 secteurs dans le département de l'Isère et 5 secteurs des Hautes-Alpes.

Le stage est placé au sein du pôle Système d'informations, lui même rattaché au service scientifique.

Le service a pour mission de définir la politique scientifique de l'établissement et de conduire les actions du Parc national en matière d'ingénierie écologique. A ce titre, il est en charge de l'élaboration des protocoles, de l'administration des données, de la restitution et de la diffusion des informations.

Le pôle Système d'informations est composé d'un géomaticien, chef du pôle SI, d'un administrateur systèmes et réseaux et d'un développeur web/bases de données.

Le pôle SI a développé plusieurs bases de données et applications métier pour la gestion des données relatives aux principales missions du parc national (inventaires et suivis faune et flore, inventaire des patrimoines, programme d'animation, gestion et valorisation des sentiers...).

La plupart des applications ont été libérés sous licence libre : <https://github.com/PnEcrins>, <https://github.com/makinacorp/Geotrek>, <https://github.com/PnX-SI/TaxHub>.

Certaines applications sont déjà destinées au grand public : <http://rando.ecrins-parcnational.fr>, <http://bouquetins.ecrins-parcnational.fr>.

Le Parc national des Ecrins souhaite étendre cette mise à disposition des données en mettant en ligne un atlas de la faune et de la flore.

Celui-ci s'appuiera sur les données présentes dans l'application GeoNature : <https://github.com/PnEcrins/GeoNature>.

Il sera publié sous licence libre, générique et documenté pour pouvoir être déployé par les autres utilisateurs de GeoNature (parcs nationaux, conservatoires, associations...).

THEME :

Développement web d'un atlas dynamique de la flore et de la faune du Parc national des Ecrins.

ENCADREMENT :

Camille MONCHICOURT (Parc national des Ecrins) / Responsable du pôle Système d'Informations / Géomaticien

Le parc national accueille chaque année quelques stagiaires. Ces stages doivent avoir un caractère obligatoire dans le cursus universitaire. Ils donnent lieu à une convention tripartite entre l'établissement d'enseignement, le stagiaire et le parc national. Ils doivent obligatoirement faire l'objet d'un rapport de stage dont le contenu sera précisé au début et en cours de stage.

Il est indispensable qu'un enseignant apporte son concours au suivi du stagiaire avant et durant le stage.

OBJECTIFS DU STAGE :

Principaux :

Dans le cadre de la mise à disposition des données et du partage des connaissances acquises par le parc national depuis sa création en 1973, les observations faune et flore constituent une importante partie de ces données (plus de 500,000 observations).

L'objectif du stage est de développer en web et de déployer un atlas de la faune et de la flore interrogeant dynamiquement les données stockées dans l'outil métier GeoNature.

Le stagiaire réalisera une

- Analyse des solutions existantes
- Définition des besoins et des fonctionnalités
- Proposition d'architecture
- Maquettage et ergonomie
- Développement
- Documentation et publication sur Github

Secondaires :

- Le stagiaire participera à la définition de l'architecture globale du système d'informations du parc national et de sa stratégie de développement.
- Le stagiaire pourra être amené à travailler sur d'autres projets de développement WEB.

LIEU :

La résidence administrative du stagiaire est fixée au siège du Parc national des Ecrins (Gap).

PROFIL et COMPETENCES REQUISES :

**Formation en développement WEB ou géomatique.
Bac +3 ou bac +5**

Compétences et connaissances requises :

- Maîtrise des langages HTML, PHP, javascript, CSS et SQL
- Maîtrise en ergonomie et architecture web
- Maîtrise des concepts du développement
- Connaissance de Bootstrap, AngularJS, Git
- Connaissance d'un framework PHP (Symfony ou autre)
- Connaissance de l'architecture REST
- Connaissance en base de données (PostgreSQL/PostGIS de préférence)
- Connaissance en administration et sécurité de serveurs web linux
- Connaissance appréciée en SIG (QGIS ou autre)
- Intérêt pour les thématiques environnementales et naturalistes
- Rigueur
- Aptitude à travailler en autonomie et en équipe.

INFORMATIONS PRATIQUES :

Indemnité de stage : Selon le barème légal en vigueur.

Durée : 5 ou 6 mois à partir de février 2016.

Annexe 2 : Compte rendu réunion atlas - thématique

Date, lieu : 06/06/2016, Charance

Objet : L'atlas web faune et flore du PNE : Quel public, quel contenu, quel ressources ?

***** PUBLIC CIBLE *****

Après rappel du contexte et des objectifs du projet (SIT, outils existants, SINP, CS, CA, CODIR), consensus sur 2 publics cibles : grand public (curieux / visiteurs) et naturalistes
Ces deux publics impliquent 2 voire 3 entrées:

- une entrée botanique/thématique: recherche par la taxonomie
- une entrée plus géographique / territoriale: recherche par commune / secteur
- une entrée par l'image.

Les approches par détermination (couleur, forme...) pourront être envisagées dans un second temps (pas réalisable à l'heure actuelle).

Pour les autres publics : rediriger vers d'autres sites qui proposent de télécharger ou de participer (SILENE, Bdflore 05, LPO etc.)

**** CONTENU DE L'ATLAS*****

Rappel des contenus générés automatiquement et manuellement pour chaque espèce

Automatiquement :

- Noms et taxonomie
 - Liens vers INPN (statuts de protection etc.)
 - Graphique altitude et phénologie (penser à un seuil de pertinence du nombre d'obs: fiches avec peu d'observation : données potentiellement erronées)
 - Carte de répartition : consensus sur le fait de faire une carte par maille ou de chaleur à petite échelle et à l'observation à grande échelle (voir au cas par cas pour les espèces sensibles)
- => Important : pour les deux derniers points, rappeler que ces données sont des observations et non pas des inventaires exhaustifs.

Pour la génération automatique : abandon de l'approche par milieu et de la carte de probabilité de présence: nous ne disposons pas des données nécessaires pour que ce soit pertinent.

Manuellement :

- Description : courte et simple
- Champs commentaire / contextualisation
- Corrologie à une échelle plus large (plante méditerranéenne, alpine etc.) à dire d'expert (travail de Baptiste notamment). Possibilité d'afficher une petite carte.
- Habitat / milieu à dire d'expert, basé sur une liste prédéfinie
- Photos : une photo fixe pour l'espèce + une galerie de photos

Autres points sur le contenu :

- Gestion des photos dans la photothèque Arjaris ? Ajouter le cd_nom dans la photothèque
Faire un inventaire des photos existantes dans la photothèque ou repartir de zéro ?
- Page d'accueil : tableau de bord de l'actualité des observations : almanach, traitements statistiques intéressants pour le grand public
- Se concentrer sur une famille de fiches espèces « exemple » (ex : arbre, fougère) pour générer une dynamique concernant le contenu.
- Réunion maquettage du site 27/06

Rédacteur : Theo Lechemia

Diffusion : SSC, SCOM, DIR

Annexe 3 : Compte rendu réunion interparc atlas-technologie

Date, lieu : Charance, 23/05/2016, conférence téléphonique

Objet : Architecture et technologiques de GeoNature-atlas

Ordre du jour :

- Infrastructure de la BD
- Architecture
- Technologies web

- **Base de données**

- Toutes les nouvelles informations relatives à chaque espèce seront saisies dans TaxHub. On ne crée pas une autre base atlas avec sa propre administration. La partie atlas de TaxHub (photo, description) sera saisie directement dans le formulaire taxon (avec droits spécifiques si besoin).

- Duplication de la base GeoNature pour l'atlas (schéma synthèse et taxonomie) ou création d'un schéma « atlas » dans GeoNature ?

Solution envisagée : duplication de la BDD de GeoNature sur un autre serveur (sous réserve de performance : foreign data wrapper). Uniquement les schémas et tables utilisés par l'atlas (synthèse et taxonomie, à affiner selon les fonctionnalités et besoins de l'atlas). Dans cette base fille intégrée à l'atlas, des champs et des vues supplémentaires pourront être ajoutés pour optimiser les calculs (répartition phréologiques, altitudinale).

- Dans TaxHub : est-ce que l'on utilise la bib_attributs ou est ce que l'on crée une table spécifique aux champs de l'atlas (sachant que ceux-ci seront nécessaire pour que l'atlas fonctionne) ? Dans tous les cas il est acté que l'atlas interrogera des vues et pas directement les tables pour que chacun ait de la souplesse. Le groupe semble préférer de garder la généricité de TaxHub en utilisant la table bib_attributs.

Interrogations : Comment l'atlas saura dans quel attribut il retrouve la description ? Comment l'atlas pourra ressortir tous les taxons d'un milieu, s'il n'a pas une bib_milieux ? Comment gérer des droits spécifiques pour les champs de l'atlas ?

- Solution de générer des vues pour les besoins de l'atlas : l'API se servira directement dans ces vues. Avantage : chacun peut « personnaliser » ses vues en fonction des besoins et des différences dans les tables. L'ensemble des vues seront regroupées dans un schéma spécifique « atlas », ce qui laisse la possibilité à terme de les remplir avec autre chose que GeoNature.

- Question de la sensibilité des données : est ce qu'on le gère au niveau du taxon ou au niveau de l'observation ? Le PNC souligne que juridiquement, c'est plutôt au niveau de l'observation, mais pour autant ce n'est pas à l'observateur de le décider au moment de la saisie. La solution serait de rajouter un champs dans la synthèse de GeoNature qui est rempli par l'administrateur. Cette information permettra à l'atlas de totalement masquer une observation ou de la dégrader.

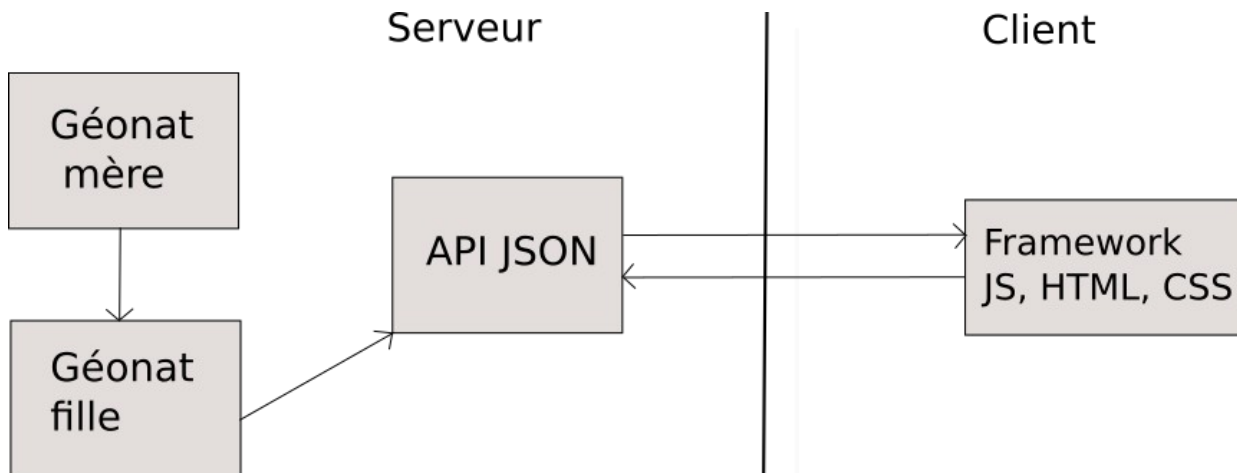
- Faire un point sur les fonctionnalités de l'atlas pour préciser les contenus de l'API.

- **Architecture**

Point abordé rapidement durant la réunion.

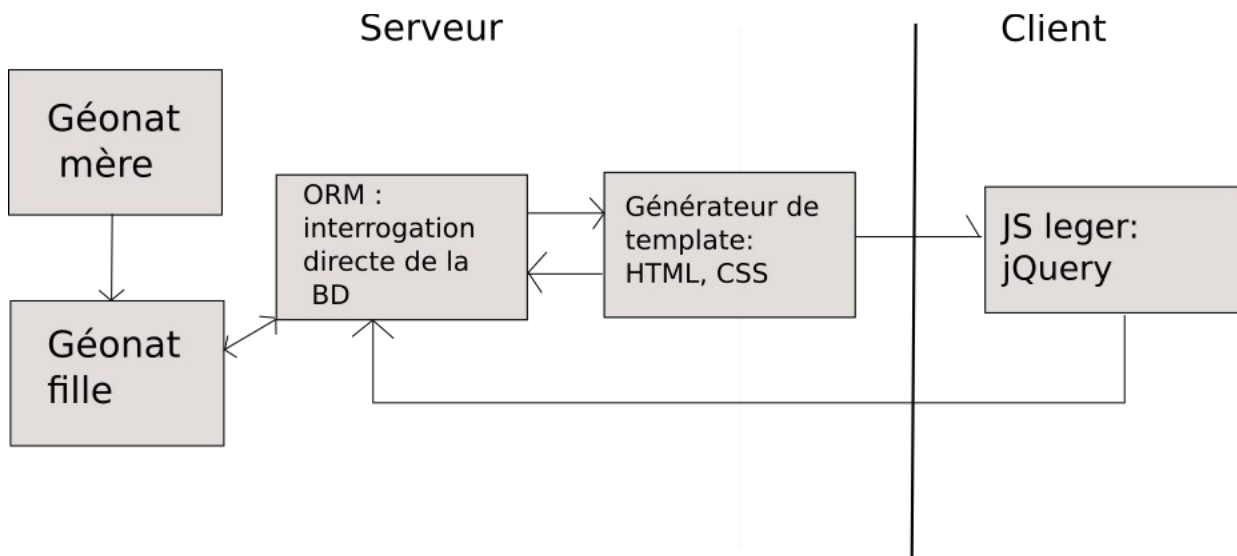
Trois solutions d'architectures sont envisagées :

- 1ère solution : Architecture de TaxHub ou GeoTrek



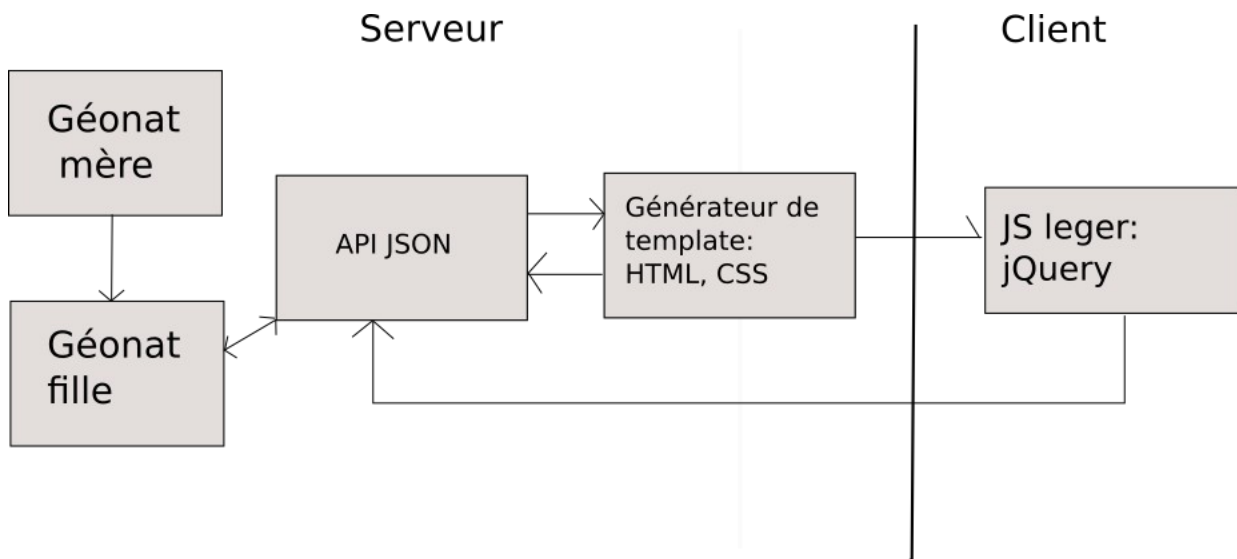
L'API se sert dans la base de données fille et génère des fichiers Json. Pour gagner en performance, ces fichiers peuvent être générés chaque nuit en dur. Le côté client, appelle cet API pour générer les pages HTML du côté front. Nécessite un framework JS lourd comme AngularJS ou React.

- 2ème solution :



Le framework serveur se sert directement dans la BD (ORM). A chaque requête du client, la BD est interrogée, et les pages HTML sont générées par le moteur de templates côté serveur. Possibilité de gérer les réponses des requêtes par de la mise en cache. Côté client, du JS léger du type JQuery pour les animations du site.

- 3ème solution :



Architecture semblable à la 2ème solution, avec génération des pages HTML côté serveur. Différence : le moteur de templates sollicite une API qui renvoie des fichiers en JSON en fonction de la requête. Possibilité de stocker chaque nuit en dur les réponses de l'API pour des questions de performance. Même solution légère côté client JS.

- **Technologies web**

Retour sur le problème de référencement des applications web rendues côté client, du type AngularJS (faisable mais compliqué car ce sont des frameworks fait pour générer des applications web singlepage).

De plus ce type de framework JS, sont davantage appropriés à un des « applications web » avec de grosses interactions côté client, ce qui n'est pas notre cas.

Pour le projet, une solution de génération des pages HTML côté serveur paraît plus adaptée. Idée initiale d'utiliser Symfony 2 pour cela.

Amandine, de son côté ne souhaite pas développer de nouvelle application avec Symfony : trop lourd pour nos besoins. Elle propose de migrer vers Python : plus simple et plus adapté aux fonctionnalités « spatiales ». Elle propose le « micro-framework » Flask. Ce n'est pas un framework complet du type Django : il est plus souple et plus modulaire, avec une communauté et un support important.

Nécessite de se former à Python et à ce framework dans la durée du stage. Pose la question maintenabilité du projet si Théo se forme tout seul sur Python.

Rédacteur : Théo Lechémi

Diffusion : SI interparcs