



Master 2Pro Systèmes, Territoires, Environnement, Patrimoines

Option Systèmes d'Information Géographique

Parcours Professionnel

Université Jean Monnet de Saint-Etienne

Khadim MBACKE

**< / Développement d'une application de cartographie
interactive sur internet >**

Rapport de stage de fin d'études.

Stage effectué du 7 avril au 6 octobre 2015 au siège du Parc national de la
Vanoise à Chambéry.

Sous la direction de :

Mr. Thierry JOLIVEAU

Directeur universitaire

Mr. Christophe CHILLET

Maître de stage

Année universitaire 2014-2015

Remerciements

Je tiens à remercier dans ces quelques lignes, toutes les personnes qui ont participé de près ou de loin à la réalisation de cette mission.

Mes remerciements vont premièrement à l'endroit du Très Haut : Dieu tout-puissant, à qui nous tenons grande reconnaissance pour toutes les merveilles qu'il a fait pour nous jusqu'ici.

Je remercie ma mère et toute la famille pour leur soutien pendant toute cette période estudiantine.

Je remercie toute la Direction du Parc national de la Vanoise et je tiens à remercier tout particulièrement et à témoigner toute ma reconnaissance aux personnes suivantes, pour leur disponibilité, leur expérience enrichissante et pleine d'intérêt qu'elles m'ont donné durant ces 6 mois au siège du Parc.

Monsieur Christophe Chillet, chef de l'Unité Système d'Information, mon maître de stage pour l'accueil et la confiance qu'il m'a accordé et apporté depuis le début et pour m'avoir intégré rapidement au sein de l'établissement; pour m'avoir aussi donné le temps et les réponses à toutes mes questions, sans oublier sa participation au cheminement de ce rapport.

Madame Claire Lagaye, géomaticienne, pour son soutien, son écoute et sa disponibilité pendant cette période.

Madame Amandine SAHL et Monsieur Frédéric FIDON du Parc national des Cévennes pour leur soutien pendant ces moments de développement et pour avoir partagé leur code source et remarques pour la réussite de cette mission.

Je remercie aussi toute l'équipe pédagogique de l'Université Jean Monnet de Saint-Etienne et les intervenants professionnels responsables de notre formation Systèmes, Territoires, Environnement, Patrimoines SIG.

Je n'oublie pas toutes les personnes qui se sont intéressées au sujet au stage comme Patrick Folliet et Benoit Martineau, Agnès Coindre et tous les agents du siège.

Résumé

Les câbles électriques ERDF/RTE ou câbles de remontées mécaniques installés peuvent être à l'origine de cas de mortalité chez certaines espèces oiseaux comme le *Gypaète barbu*. Un projet dénommé de *LIFE GypHelp* (projet avec subvention européenne) a été lancé et un de ses objectifs est la réduction des risques de percussioin et d'électrocution des oiseaux (grands rapaces) sur ces câbles aériens identifiés comme dangereux.

D'un point de vue métier, depuis plusieurs années, le *Parc national de la Vanoise* s'est investi dans un projet de recensement de la *dangerosité de ces câbles et pylônes* pour l'avifaune et d'inventaire et de suivi des équipements de neutralisation mis en place sur ces éléments dangereux.

D'un point de vue technique, une application de *cartographie dynamique* sur internet est actuellement en place. Elle a été réalisée depuis la solution de développement propriétaire AIGLE™.

Ce projet a pour but de redéployer cette *application dynamique web cartographique* sur un environnement de développement *open source* (*PostgreSQL/PostGIS*, *Symfony2*, *AngularJS*, *Leaflet*) et potentiellement faire évoluer ses fonctionnalités.

Mots-clés : *Gypaète barbu, LIFE GypHelp, Parc national de la Vanoise, dangerous electric cables, application web, internet, WebSIG, open source, PostgreSQL/PostGIS, Symfony2, AngularJS, Leaflet*

Abstract

Electrical cables ERDF / TEN or cable lifts installed may be the cause of mortalities in some bird species such as the *Bearded vulture*. A project named LIFE GypHelp (European project with grant) was created and one of its objectives is the reduction of risk of percussive and electric birds (birds of prey) on these overhead cables identified as dangerous.

From a business perspective, for several years, the *Vanoise national Park* has invested in a census project of the dangerousness of these cables and posts for birds and inventory tracking and neutralizing equipment placed on these dangerous elements.

From a technical point of view, an *application of dynamic mapping on the internet* is currently in place. It was conducted from an owner development: AIGLE™ solution.

This project aims to redeploy the web mapping application on a dynamic open source development environment (*PostgreSQL / PostGIS, Symfony2, angularjs, Leaflet*) and potentially evolve its features.

Keywords: *Bearded, LIFE GypHelp, the Vanoise national Park, dangerous electric cables, web application, WebGIS, open source, PostgreSQL / PostGIS, Symfony2, angularjs, Leaflet*

Sommaire

REMERCIEMENTS	2
RESUME	3
ABSTRACT	4
SOMMAIRE.....	5
TABLE DES ILLUSTRATIONS.....	6
GLOSSAIRE	7
INTRODUCTION.....	9
CHAPITRE I - PRESENTATION ET CONTEXTE DU STAGE.....	11
I. LE PARC NATIONAL DE LA VANOISE	11
1. Histoire du Parc	12
2. Missions du PNV.....	12
3. Siège du Parc national de la vanoise	13
II. LE SYSTEME D'INFORMATION GEOGRAPHIQUE DANS LES PARCS NATIONAUX	14
III. APPLICATION CABLES ET OBJECTIFS DU STAGE	16
1. Protocole câbles.....	16
2. Objectif du stage	18
IV. LES OUTILS MOBILISES	18
1. Le côté serveur	19
2. Le coté Client	20
CHAPITRE II - DEVELOPPEMENT DE L'APPLICATION	24
I. LE COTE SERVEUR.....	26
1. La base de données (BDD)	26
2. Symfony2 et Doctrine	27
3. Requêtes et réponses HTTP	28
II. LE COTE CLIENT.....	29
1. Une application avec AngularJS	29
2. La partie cartographie: Leaflet et Angular ?	30
3. L'ergonomie de l'application : Bootstrap et CSS	31
CONCLUSION	32
ANNEXES.....	34
BIBLIOGRAPHIE	46
WEBOGRAPHIE.....	46

Table des illustrations

Figure 1: carte de localisation du Parc National de la Vanoise	11
Figure 2: exemple définition des SIG.....	14
Figure 3: base de données géographiques avec PostgreSQL et PostGIS	19
Figure 4: exemple d'architecture d'une application AngularJS.....	20
Figure 5: carte basique Leaflet avec trois formes géométriques différentes	21
Figure 6: exemples de quelques fonctionnalités Bootstrap.....	21
Figure 7: carte basique Leaflet avec trois formes géométriques différentes	22
Figure 8: capture d'écran de Sublime Text 3.....	23
Figure 9: architecture de l'application Câbles.....	25

Glossaire

API : Application Programming Interface (Interface de programmation d'Application), c'est ce qui permet à deux systèmes informatiques totalement indépendants de se parler de façon automatique.

Bundle: c'est un répertoire qui a une structure bien définie et qui peut héberger à peu près tout, des classes aux contrôleurs en passant par les ressources web.

CSS: Cascading Style Sheets, c'est un langage informatique utilisé sur l'internet pour mettre en forme les fichiers HTML ou XML.

DOM : Document Object Model, c'est une interface de programmation d'application (API) pour des documents HTML et XML bien formés. Il définit la structure logique des documents et la façon dont un document est accessible et manipulé.

EPSG: European Petroleum Survey Group (devenue depuis l'Oil and Gas Producers Surveying and Positioning Committee -OGP-), il regroupe sous un code unique à 5 chiffres les systèmes de référence spatiale existant dans le monde.

Framework: c'est un ensemble structurel de composants logiciels qui permet de créer et de modeler l'architecture des applications (applicatif, web...).

Géomatique: elle regroupe l'ensemble des outils et méthodes permettant d'acquérir, de représenter, d'analyser et d'intégrer des données géographiques. La géomatique étant liée à l'informatique, son application passe par l'utilisation d'outils informatiques que l'on nomme les SIG.

HTML: HyperText Markup Language est un langage informatique utilisé pour créer des pages web.

HTTP: HyperText Transfer Protocol, « protocole de transfert hypertexte », c'est un protocole de communication client-serveur développé pour le World Wide Web (www).

IGN: Institut national de l'information géographique et forestière, c'est un établissement public à caractère administratif ayant pour mission d'assurer la production, l'entretien et la diffusion de l'information géographique de référence en France.

JSON : JavaScript Object Notation, c'est un format de données textuelles qui permet de représenter de l'information structurée comme le XML.

Linux: c'est un système d'exploitation complet et libre qui peut être utilisé en lieu et place de systèmes d'exploitation commercialisés, tels que Windows avec de nombreux logiciels libres complémentaires, offrant un système complet aux utilisateurs.

MVC: Modèle Vue Contrôleur, c'est un modèle d'architecture destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants (base de données, côté serveur, et côté client) au sein de leur architecture respective.

Open Source:

ORM : Object-relational Mapping, c'est une technique de programmation informatique qui crée l'illusion d'une base de données orientée objet.

PHP: Hypertext Preprocessor, c'est un langage de programmation libre pour faire des pages web dynamiques.

Projection : c'est un système de coordonnées spatiales résultant de l'application de formules mathématiques destinées à traduire des coordonnées géographiques en coordonnées planes (carte).

SFTP: Secure File Transfer Protocol, c'est une version sécurisée du protocole de transfert de fichiers (FTP), qui facilite l'accès aux données et le transfert de données sur un Shell (SSH) flux de données sécurisé.

SGBD : Système de Gestion de Base de Données, c' est un logiciel système destiné à stocker et à partager des informations dans une base de données, en garantissant la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations.

SIG: Système d'Information Géographique, c'est un système qui a pour but d'informer sur la géographie d'un espace donné en s'appuyant sur un certain nombre de bases de données géographiques, qu'il permet d'intégrer, de gérer, de traiter et de représenter sous forme de cartes.

SQL: Structured Query Language (en français langage de requête structurée), c'est un langage informatique normalisé servant à exploiter des bases de données relationnelles.

URL : Uniform Resource Locator, littéralement « localisateur uniforme de ressource »), c'est une chaîne de caractères utilisée pour adresser les ressources du World Wide Web.

Webmapping: c'est méthode qui regroupe l'ensemble des technologies permettant d'afficher une carte par internet (navigateur web ou mobile).

WGS : World Geodesic System, le système géodésique mondial. Il définit une représentation du géoïde terrestre (différent de projection).

WMS : le Web Map Service est un protocole de communication standard qui permet d'obtenir des cartes de données géo-référencées à partir de différents serveurs de données.

XML: l'eXtensible Markup Language est un langage informatique qui sert à enregistrer des données textuelles.

YAML: Ain't Markup Language, est un format de représentation de données par sérialisation Unicode. Il reprend des concepts d'autres langages comme XML

Introduction

Les parcs nationaux sont des structures créées par l'État. Ils ont pour missions la préservation des ressources naturelles, la protection de la biodiversité, la bonne gouvernance, la gestion du patrimoine culturel mais aussi l'accueil des publics. La réussite de ces missions s'appuie sur un ensemble d'outils et de méthodes.

Plusieurs définitions du **SIG*** (Système d'Information Géographie) ont été données. Mais on note que le SIG peut être défini comme l'ensemble des structures, des méthodes, des outils et des données constitué pour rendre compte de phénomènes localisés dans un espace spécifique et faciliter les décisions à prendre sur cet espace (Thierry Joliveau, 1996, p. 111). Comme tout système d'Information, il combine et articule des données, du matériel, des logiciels, des structures organisationnelles et des méthodes pour représenter les objets géographiques nécessaires à un projet d'action et de connaissance de l'organisation¹. Les parcs nationaux disposent de services SIG qui participent aux missions qui leur sont confiées.

Nous assistons avec la modernité et l'avancée de la technologie, à une orientation du SIG vers le Web, ce que nous appelons le **webmapping*** ou la cartographie en ligne. Cette nouvelle méthode a pour but de produire, de traiter et de publier des données géographiques géo-référenciées via une interface web.

Le Parc national de la Vanoise (PNV) s'est investi dans un projet de recensement de la dangerosité des câbles et pylônes (lignes électriques et remontées mécaniques) pour l'avifaune et de l'inventaire mais aussi le suivi des équipements de neutralisation mis en place sur ces éléments dangereux. Ainsi, une application de cartographie dynamique sur internet a été créée et est toujours en place. Elle a été réalisée depuis une solution de développement propriétaire, la solution AIGLE™. Le PNV a pris la décision d'arrêter d'utiliser la solution AIGLE et de redévelopper cette application en utilisant les technologies open source.

Dans le cadre de ma formation Master II PRO Systèmes, Territoires, Environnement, Patrimoines (STEP), spécialité SIG à l'Université Jean Monnet de Saint – Etienne, un stage d'une durée minimale de 5 mois est obligatoire pour compléter la formation. J'ai tenu à

¹ Glossaire de la géomatique, des SIG et du géoweb du Master SIG de Saint-Etienne.

effectuer mon stage au Parc de la Vanoise avec comme mission la participation au redéploiement et à l'évolution fonctionnelle de cette application.

Du 07 avril au 06 octobre 2015, j'ai effectué ce stage au sein de l'Unité Système d'Information du Parc national de la Vanoise sous la direction de Monsieur Christophe CHILLET, responsable de l'unité. Ce rapport décrit ainsi les missions effectuées pendant le stage.

Pour mieux présenter mon travail, je présenterais dans une première partie, la structure d'accueil, à savoir le Parc national de la Vanoise, dans laquelle le stage s'est déroulé pour mieux contextualiser le sujet et montrer le rôle des SIG dans les parcs nationaux. Dans un second temps, je présenterais les missions au cours de ce stage avant de faire un bilan de celui-ci du point de vue personnel et professionnel.

Chapitre I -Présentation et contexte du stage

I. Le Parc national de la Vanoise

L'établissement public en charge du Parc national de la Vanoise (PNV) est placé sous tutelle du ministère de l'écologie, du développement durable et de l'énergie. Il se situe en région de Rhône-Alpes dans le département de la Savoie. Il s'étend entre la vallée de la Tarentaise au Nord et celle de la Maurienne au Sud. Il est constitué d'un cœur (zone réglementée de 529 km²) et de l'aire optimale d'adhésion, sans réglementation spécifique (1465 km²).

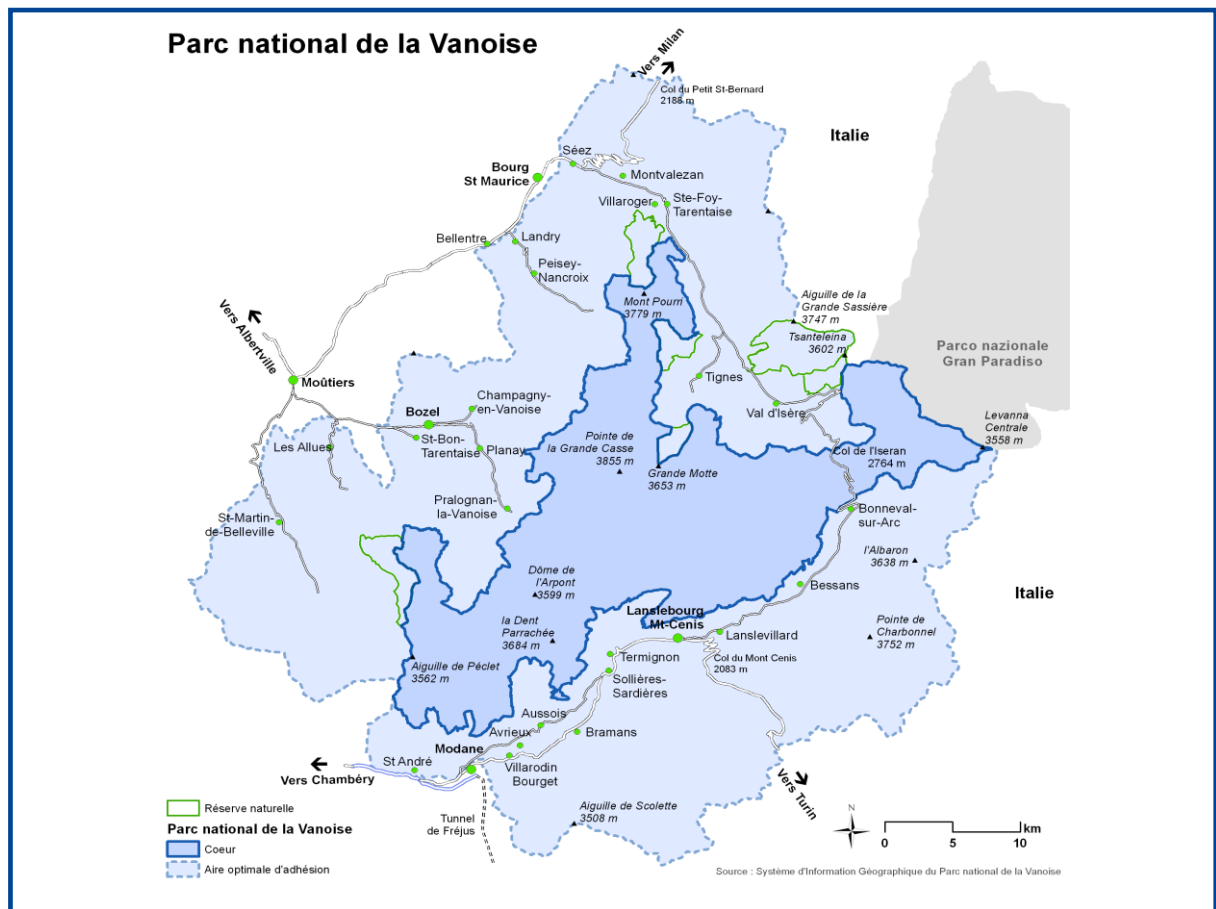


Fig.01: carte de localisation du Parc national de la Vanoise

Le PNV est jumelé avec le Parc national Italien du Grand-Paradis depuis 1972. C'est ainsi que le cœur du Parc national de la Vanoise et le Parc Italien du Grand-Paradis forment ensemble 1250 km², qui de ce fait représente l'une des surfaces protégées les plus étendues d'Europe Occidentale.

1. Histoire du Parc

En 1943, le docteur Couturier milite pour un "parc national à bouquetins" en Vanoise pour la protection des bouquetins, toujours menacés par la chasse excessive. Ceci a participé à la réflexion sur leur protection ainsi qu'à la mise en place d'une réserve royale en Italie par le roi Victor Emmanuel, qui devint le Parc national Italien du Grand Paradis. A partir de 1955, Gilbert André, souhaitant valoriser la culture parle d'une mise en place d'un parc culturel des Alpes, symbiose de l'homme et de la montagne. Le Club Alpin Français propose un parc de 13 000 hectares, axé sur la protection et le tourisme hivernal. La synthèse de ces idées aboutit en 1960 à la loi sur la création des parcs nationaux.

Ce sont des espaces protégés soumis à une réglementation spécifique (articles L331 et R331 du code de l'environnement) qui assure la sauvegarde de leur patrimoine naturel et culturel reconnu comme exceptionnel. Le Parc national de la Vanoise est le premier parc national français et la France en compte actuellement 10.

2. Missions du PNV

Le Parc national de la Vanoise ainsi que les autres parcs nationaux a pour missions:

- la protection et la préservation des espèces, des habitants et des ressources naturelles comme l'organisation des travaux ou même la restauration des milieux naturels.
- la connaissance dans la mesure où il favorise les initiatives dont le but est la connaissance et le suivi du patrimoine naturel, culturel et paysager tout en assurant lui-même les actions de suivi. En partenariat avec les collectivités locales et les grandes associations de montagne, il a développé un puissant réseau d'information pour le public.
- la sensibilisation et l'éducation à l'environnement dans lequel il réalise des supports de communication et d'information, organise des sorties de découverte pour les visiteurs. Il intervient dans les programmes pédagogiques des écoles environnantes.
- la participation au développement local et durable, sentier technique en matière de préservation des espèces naturelles aux collectivités territoriales. Il joue aussi un rôle important dans la réalisation d'aménagements sur le patrimoine naturel, paysager et culturel.

3. Siège du Parc national de la Vanoise

Le Parc est composé d'une équipe d'agents sous l'autorité du Directeur. Son siège se trouve à Chambéry et héberge la direction, les services administratifs, le pôle patrimoine, le pôle communication, le pôle développement durable (voir l'organigramme en Annexe I).

Sur le territoire propre, il est aussi composé de 4 secteurs répartis sur les vallées de la Maurienne et de la Tarentaise. Chaque secteur est composé d'un chef de secteur (qui dépend du directeur du parc), d'un technicien, de 3 à 7 gardes-moniteurs et d'une secrétaire. Les secteurs sont :

- Secteur de Pralognan-La-Vanoise
- Secteur Haute Tarentaise
- Secteur de Modane
- Secteur haute Maurienne

Le Parc compte 75 agents répartis entre le siège et les secteurs. Son équipe, avec des compétences multiples, lui permet de mettre en œuvre les missions :

- de surveillance du territoire ;
- de connaissance, de suivi et de gestion du patrimoine naturel, culturel et paysager ;
- de participation à la recherche scientifique ;
- de conseil, d'étude et d'ingénierie auprès des communes et des acteurs locaux pour un développement durable ;
- d'accueil du public en milieu naturel ;
- de création de supports de communication et la réalisation d'animations pédagogiques.

Le budget annuel est de l'ordre de 6 millions d'euros en fonctionnement et investissement (en majorité subventions du Ministère chargé de l'environnement).

II. Le Système d'Information Géographique dans les parcs nationaux

La société française de photogrammétrie et de télédétection définit le SIG comme « un système d'informatique permettant, à partir de **diverses sources**, de **rassembler** et d'**organiser**, de **gérer**, d'**analyser** et de **combiner**, d'**élaborer** et de **présenter** des **informations localisées géographiquement**, contribuant notamment à la **gestion de l'espace**. » Comme tout système de ce type il s'articule autour de trois composants principaux :

- le **matériel** (logiciel, machine, serveur, réseau...) ;
- la **donnée** (référentielle, thématique, à grande ou à petite échelle) ;
- la **ressource humaine** (géomaticien, décideur, commanditaire, usagers).

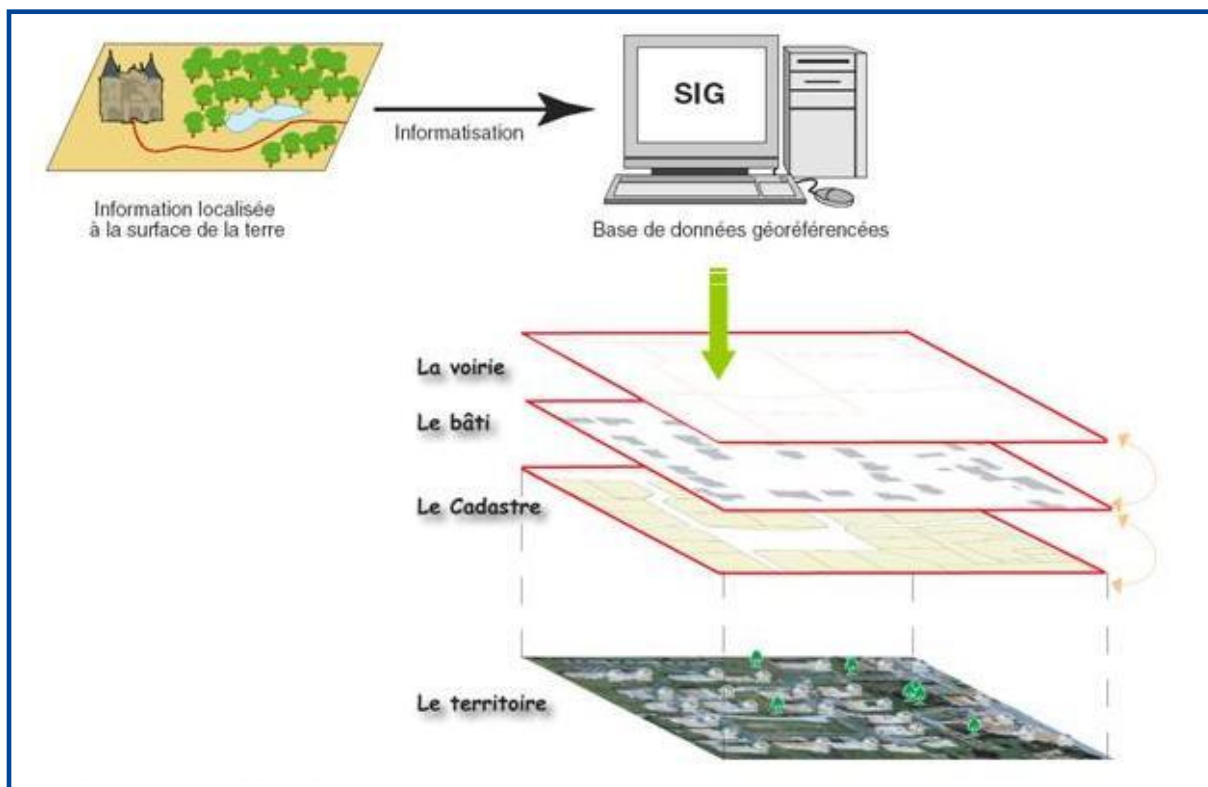


Fig.02: exemple de définition des SIG

La mise en place d'un SIG est très importante dans une structure comme les parcs nationaux. Ils sont utilisés depuis de nombreuses années dans les missions des parcs pour participer à leur accomplissement.

Avec le développement de la technologie, le monde du SIG a beaucoup évolué. La gestion des SIG est toujours une affaire de spécialiste pour l'administration et la validation des données, mais maintenant en faveur de sa diffusion auprès de tout public. Ceci suit les

préconisations européennes qui sont mises en place par les différents États, comme la **Directive Inspire**. L'avancée technologique dans les SIG permet de faire de la cartographie interactive sur internet (webmapping, WebSIG). Ceci s'appuie sur de technologies propriétaires ou *open source**. Le SIG en ligne est considéré comme l'avenir du SIG, différent du « SIG classique » (logiciel propriétaire ou open source) qui nécessite obligatoirement une installation du produit sur des postes locaux. Le SIG en ligne nécessite seulement d'avoir un navigateur internet qui permet de dialoguer avec l'environnement applicatif déployé sur un serveur, lui-même se basant sur 3 composants : le côté *client*, le côté *serveur*, les *données* (base de données).

En prenant en compte tout ce contexte, les parcs nationaux en tant qu'établissement public, donc ayant l'obligation de diffuser leurs données ont entrepris depuis plusieurs années, dans le domaine des SIG, un processus de mutualisation des outils, des compétences, des méthodes, des référentiels... pour répondre à cette évolution. Ainsi pour la réalisation d'applications de cartographie dynamique, les parcs nationaux ont fait le choix de technologie de développement open source.

Au PNV, le service SIG gère toutes les données géographiques de l'établissement. Il participe à la mise en œuvre de toutes les missions du Parc dans lesquelles l'informatique entre en jeu, apportant son soutien dans toutes les phases des projets. Il a donc une mission transversale. Actuellement, les logiciels QGIS et ArcGIS sont utilisés pour les traitements de données SIG. Le Parc est aussi doté d'un Intranet SIG (sur internet et appareil mobile) mettant à disposition des applications de cartographie interactive articulées autour de bases de données « métiers » utilisées par les chargés de mission, les agents de terrain et aussi par les partenaires du territoire et institutionnels. C'est dans ce contexte de fonctionnement interne et inter-parc que l'application sur la gestion des câbles dangereux pour la protection de l'avifaune a été créée, qui est l'objet de mon stage.

III. Application câbles et objectifs du stage

1. Protocole câbles

Cette application web dynamique se base sur un protocole d'inventaire et de visualisation des câbles aériens dangereux ou potentiellement dangereux pour les grands rapaces, le gypaète en particulier. Ce protocole peut être divisé en quatre parties :

a. La désignation des zones sensibles

L'objectif est d'inventorier les zones de présence de Gypaète barbu et des espèces associées ciblées. Cela permet de cibler et d'orienter les actions de réduction des risques de percussioin et d'électrocution dus aux câbles aériens dangereux ainsi que celles visant à réduire les risques de dérangements anthropiques. Cette cartographie permet de travailler localement pour les actions de conservation avec les partenaires en se basant sur des données fiables et précises.

Le résultat attendu est surtout la localisation des zones sensibles pour le gypaète barbu et les autres grands rapaces sur l'ensemble de la zone du projet et de réaliser la cartographie des câbles aériens dangereux et potentiellement dangereux.

La représentation graphique se fait en fonction des espèces prises en compte : le Gypaète barbu, l'Aigle royal, le Grand-Duc d'Europe, le Faucon pèlerin, le Vautour fauve, le Circaète Jean-Leblanc et le Tétraz-lyre.

b. Inventaire des lignes électriques

L'inventaire des lignes passe par une identification des gestionnaires concernés, comme les régies électriques privées ou certains services de remontées mécaniques et le Réseau de Transport d'Electricité (RTE). Quatre étapes sont ainsi notées :

- **Etape 1** : Hiérarchisation des zones sensibles dont les zones à cas avérés de mortalité dus aux câbles, les zones de priorité "espèces" par l'attribution de points aux espèces en fonction de leur statut sur le site et leur degré de vulnérabilité, les zones "travaux de neutralisation" pour prendre en compte des zones sur lesquelles des travaux d'équipements de câbles (remontées mécaniques, autres types de lignes électriques) ont été déjà réalisés.

- **Etape 2** : Délimitation Périmètre d'inventaire (zones à cas avérés de mortalité, zones de priorité " Espèces").
- **Etape 3** : Inventaire des armements où deux types de risque existent vis-à-vis des lignes électriques : le risque de percussion et le risque d'électrocution.
- **Etape 4** : Neutralisation des armements pour éviter l'électrocution, par une isolation ou incitation, et pour éviter la percussion, par l'installation de Firefly Alpine ou Avisphère².

c. Inventaire des câbles de remontées mécaniques

Les infrastructures implantées en montagne pour la pratique du ski alpin peuvent être à l'origine d'une mortalité importante chez certaines espèces. La cartographie des tronçons potentiellement dangereux vient donc en complément de l'inventaire des câbles aériens dangereux (cas avérés de mortalité).

d. Promotion de la visualisation

Il convient d'élaborer des plans de visualisation avec chaque gestionnaire concerné. L'objectif étant de planifier, en concertation avec chaque gestionnaire concerné, la visualisation des infrastructures identifiées comme dangereuses et potentiellement dangereuses, en place et en projet. L'objectif étant que les services puissent intégrer et prévoir dans leur planning, le coût ainsi que la programmation des travaux.

Ainsi, dans le cadre de ce protocole, le nombre de données commençait à prendre de l'ampleur et devenait de plus en plus difficile à gérer pour la personne en charge du projet.

Par la suite, des besoins de consultation, de saisie et de restitution par les partenaires se sont ajoutés et ont incité à la réflexion de la mise en place d'une application de cartographie interactive sur internet pour gérer toutes ses données.

L'application au début, uniquement destinée à une utilisation sur le département de la Savoie est maintenant étendue sur tous les départements de la région Rhône Alpes. Ceci a impliqué

² Balises Avifaune installées sur les conducteurs identifiés dangereux pour les rendre plus visibles et ainsi éviter que les oiseaux ne les percutent en plein vol, notamment par temps de brouillard.

une évolution au niveau fonctionnel, particulièrement pour la gestion de la diffusion des données et des profils utilisateurs (augmentation des partenaires).

Cette application a été réalisée au PNV à l'aide des solutions open source choisies en inter-parcs. De plus, elle s'appuie sur des briques fonctionnels de développement générique développé par le Parc national des Cévennes.

2. Objectif du stage

Le contenu de mon stage est de participer à la définition de ces nouvelles fonctionnalités ; de développer cette nouvelle application avec les outils open source choisis (décrits dans le chapitre suivant) ; de participer à son animation technique et fonctionnelle (présentation du contexte technique, de l'application si finalisée).

Quelques fonctionnalités de base sont à noter dans cette application :

- la consultation, la saisie, la modification, la suppression géographique et attributaire uniquement de points (polyônes, dangerosité/neutralisation) et de segments (tout ou partie de câbles, dangerosité/neutralisation) ;
- l'exportation/reporting des données géographiques et attributaires (couches, tables...).

Dans un premier temps, ces fonctionnalités de bases seront développées pour garder le même fonctionnement que l'application existante et puis d'autres fonctionnalités seront ajoutées à celles-ci pour faire évoluer l'application.

IV. Les outils mobilisés

L'application sera développée dans un environnement Linux (Ubuntu) sur un serveur hébergé. Le choix du fournisseur d'hébergement a aussi été fait dans le cadre de la mutualisation afin d'avoir des environnements de travail cohérents en inter-parcs.

Pour ce projet et pour de nombreux projet inter-parcs, aucun serveur cartographique n'est utilisé (*MapServer* ou *GeoServer*) pour générer des fonds dynamiques. Ceci est remplacé par l'API IGN, mise à disposition gratuitement pour les établissements publics.

Les outils mobilisés dans ce projet peuvent être classés en trois catégories : **serveur**, **client**, **autres**.

1. Le côté serveur

a) PostgreSQL/PostGIS

PostgreSQL est un outil de gestion de bases de données qui s'appuie sur des modèles de données relationnels. Pour gérer des objets géographiques par l'extension PostGIS. Toujours dans cette logique de mutualisation, PostgreSQL et PostGIS est l'outil de référence que les parcs nationaux utilisent depuis plus de 10 ans. Cet outil était alors déjà en place sur le projet existant.

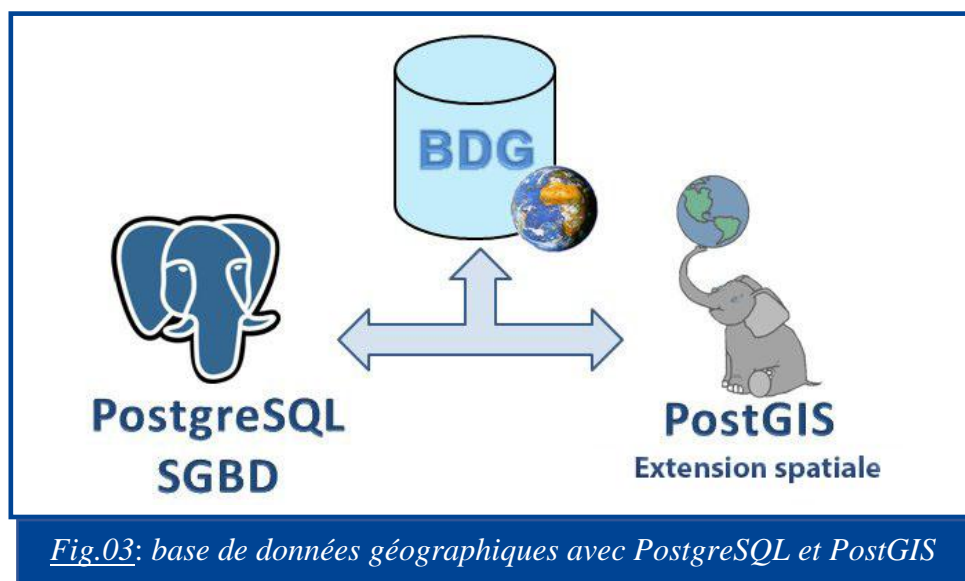


Fig.03: base de données géographiques avec PostgreSQL et PostGIS

b) Symfony2

C'est un Framework **PHP*** développé pour créer et structurer des applications web. Symfony2 est organisé autour du modèle **MVC** (Modèle Vue Contrôleur) qui permet de cloisonner les éléments tout en simplifiant l'architecture. Il a utilisé dans ce projet pour gérer une partie du côté serveur permettant de faire le lien entre la base de données et le côté client. C'est un outil qui demande une certaine maîtrise des langages de programmation orientée objet.

2. Le coté Client

a) AngularJS

AngularJS est un Framework JavaScript³ open source pour faire des applications web dynamiques. Il étend le HTML pour le rendre plus dynamique et permet de développer ses propres balises et attributs HTML. Il a été inventé par des développeurs Google dans le cadre d'un projet pour simplifier et factoriser leur ligne de codes. Le PNV utilisait la bibliothèque JavaScript **ExtJS** qui est devenu payante dans sa dernière version, poussant le PNV à choisir AngularJS qui utilise le même modèle d'architecture que Symfony2, le modèle **MVC***. Ce choix a été fait parce qu'aujourd'hui il est très utilisé dans la communauté des développeurs web.

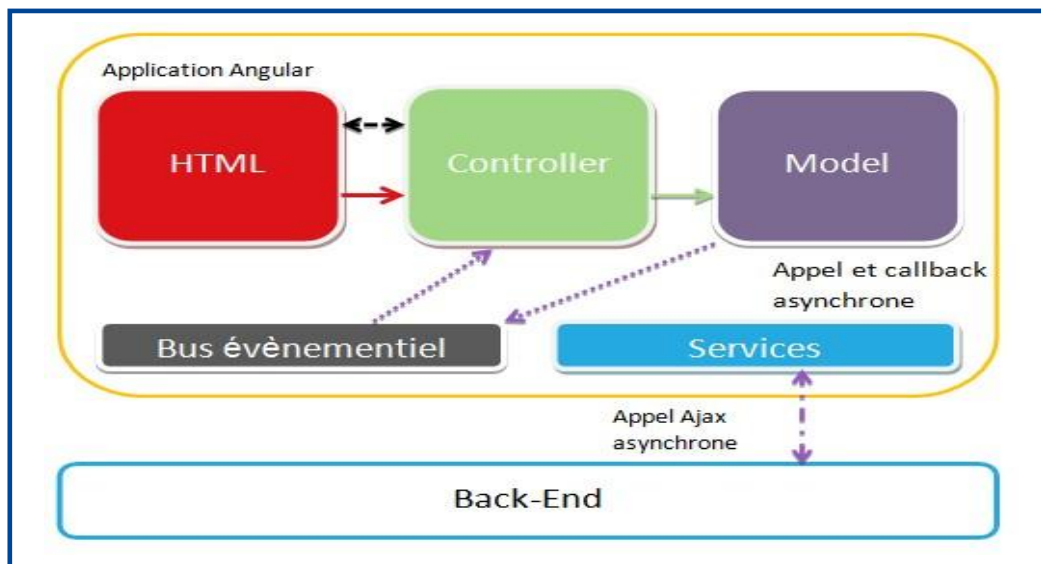


Fig.04: exemple d'architecture d'une application AngularJS

b) Leaflet

Leaflet est une bibliothèque open source JavaScript moderne qui a toutes les fonctionnalités pour faire des cartes interactives. Elle est très légère (environ 30 ko de codes), facile d'utilisation, bien documentée et la plus populaire aujourd'hui de ces bibliothèques. Elle permet entre autre de gérer des couches tuilées de base, d'afficher des données géo-spatiales.

³ Langage de programmation de script utilisé dans les pages web interactives.

Elle dispose aussi des outils basiques de navigation. Le développement d'extensions par la communauté de développeur apporte de fonctionnalités supplémentaires.

Il faut rappeler qu'au début du projet, le choix s'était porté sur OpenLayer3 mais Leaflet a été choisie pour deux raisons. Comme pour AngularJS, Leaflet est aujourd'hui très utilisée dans la communauté des développeurs web. De plus, elle était déjà utilisée par le Parc national des Cévennes.

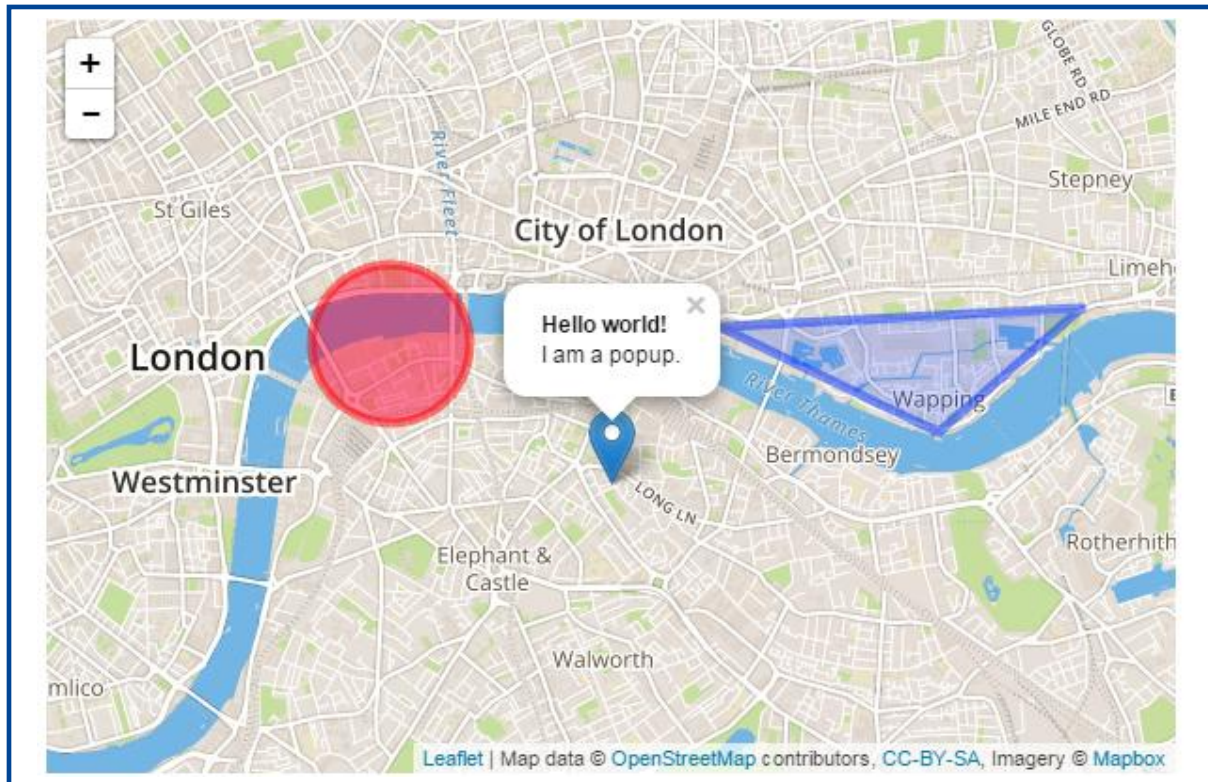


Fig.05: carte basique Leaflet avec trois formes géométriques différentes

c) Bootstrap

Toute application web a besoin d'une bonne ergonomie. Bootstrap est un ensemble d'outils et d'extensions en Javascript pour rendre plus dynamique et ergonomique les fonctionnalités d'une application web. Il est compatible avec toutes les dernières versions des navigateurs internet. Cet outil combiné avec AngularJS devient une véritable arme en développement web. L'exemple ci-dessous montre quelques fonctionnalités proposées par l'outil: menus responsables, menus déroulants, barres de navigations, boutons...

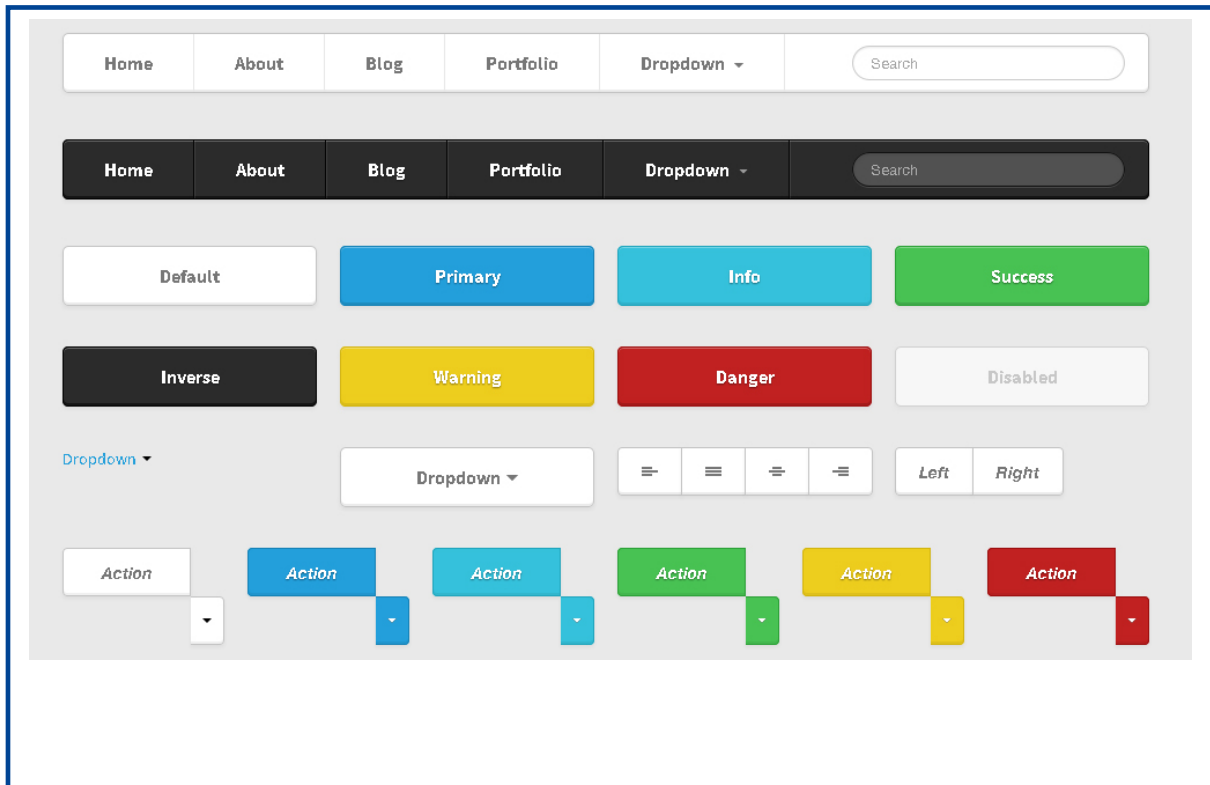


Fig.06: quelques fonctionnalités de Bootstrap

3. Autres outils

a) WinSCP

WinSCP est un outil open source gestionnaire de fichiers sur ou entre des serveurs distants. Il s'agit d'un client SFTP (Secure Shell File Transfer Protocol). Il est facile d'utilisation, sécurisé et pratique dans la gestion des serveurs.

b) PuTTY

PuTTY est un programme qui permet de se connecter à distance à des serveurs. On utilise les commandes pour une question de performance et de fiabilité comme le transfert des fichiers lourd de serveur à serveur ou de serveur à local, la suppression des répertoires, la création de liens symboliques vers nos répertoires, la sauvegarde et la restauration des bases de données, la configuration de la sécurité (PORT de connexion), le vidage des cache sur Symfony2...

c) Sublime Text 3

Un éditeur de texte puissant, Sublime Text 3 est développé en C++ et en Python. Il présente un ensemble de fonctionnalités intéressantes comme l'auto-complétion qui se base sur des bibliothèques, la vue map, qui est une petite fenêtre à coté montrant tout le code. Il supporte des plugins en Python pour son évolution. Il supporte aussi des packages ou bibliothèque des différents langages qu'il supporte pour aller plus vite dans la saisie grâce à ses fonctionnalités d'auto-complétion. On peut aussi rapide modifier des attributs dans le code en même, ce qu'on appelle la sélection multiple. Il est aussi très utilisé par la communauté des développeurs web.

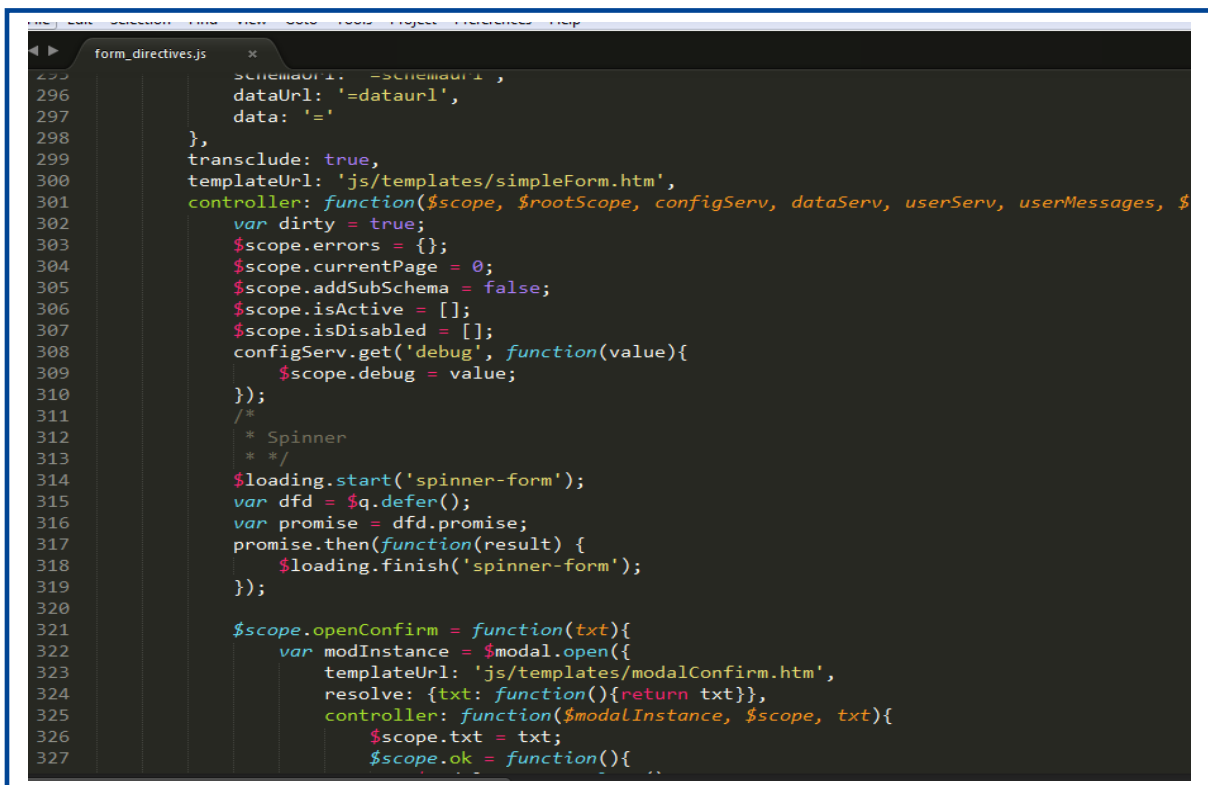


Fig.07: capture d'écran de Sublime Text 3

d) Outil de débogage

Il faut aussi noter que pendant les phases de développement d'une l'application, des outils pour déboguer les erreurs sont utilisés. Moi personnellement, j'utilise la console de Google Chrome, qui est très bien. Elle nous montre toutes les erreurs liées à une page donnée et identifie la nature des erreurs et les fichiers concernés comme si l'on a oublié de déclarer une variables dans le code.

Chapitre II - Développement de l'application

J'ai eu un grand plaisir, pendant mon stage, de plonger à la fois, dans la vie d'un développeur et d'un géomaticien et d'affronter toutes les difficultés de la pratique dans ces deux domaines. Je disposais d'autonomie dans ma façon de travailler bien que les solutions de développement et les fonctionnalités de base de l'application étaient déjà choisies et définies à l'avance. Aucune contraintes techniques de développement ne m'ont été imposées.

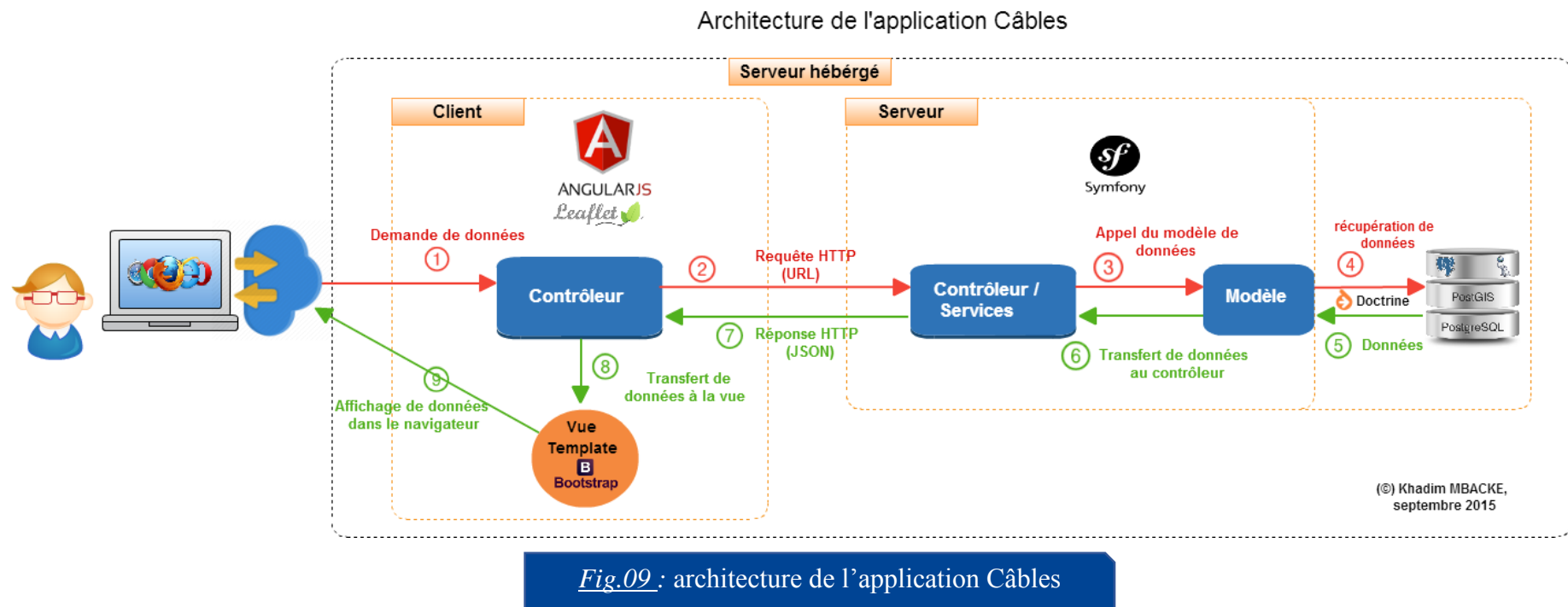
Dans le cadre de ce projet aucun planning n'a été défini mais on peut noter trois étapes principales :

- ✓ une autoformation sur les différents outils choisis sur lesquels je n'avais aucune expériences (Symfony2 et AngularJS) et une montée en compétence sur ceux que j'avais déjà utilisés (PostgreSQL, commandes Linux, Leaflet, JavaScript).
- ✓ le développement de l'application (mise à jour de la base de données du côté serveur et du côté client mise en place des différentes fonctionnalités, débogage, documentation...).
- ✓ animation technique et fonctionnelle de l'application.

Dans cette partie ce rapport, je vais présenter, de manière détaillée, les missions que j'ai effectuées pendant mon stage.

La mise en place d'une application web suit un ensemble de processus. La configuration se fait en deux étapes : le côté serveur et le côté client. Les données affichées à l'écran ne sont que le résultat de ces processus. L'utilisateur n'a pas besoin de connaître ce qui se passe du côté serveur. Seules ces données affichées du côté client l'intéressent.

Le schéma ci-dessous explique le fonctionnement de ce type d'application :



I. Le côté serveur

Il s'agit ici de décrire la configuration de tous les outils présélectionnés ci-dessus nécessaires pour mettre en place le serveur : *PostgreSQL/PostGIS* pour nos bases de données géographiques et attributaires et *Symfony2* pour définir l'architecture et préparer toute la communication entre le côté serveur et le côté client.

1. La base de données (BDD)

Une base de données est obligatoire dans une application de ce type. Son but est de stocker des informations de manières structurées. Dans ce projet, on se base sur le modèle de données de l'application existante. L'ajout, la modification et la suppression de certaines tables ont été faits en fonction des besoins de la nouvelle application. Une base de données a donc été créée avec deux schémas :

- *câbles*, qui regroupe toutes tables de données attributaires et géométriques de l'application ;
- *utilisateur*, regroupant les tables sur les données utilisateurs.

Les tables sont constituées de couches métiers sur les zones sensibles, les mortalités par électrocution et percussion et des couches de référence rattachées aux partenaires comme ERDF et RTE (voir modèle de la base de données en *Annexe II*).

Une reprojection des couches géographiques de cette base de données a été faite de la projection *Lambert-93 (EPSG Projection 2154)* vers la *projection WGS-84 (EPSG 4326)*. Dans un premier temps, ces données ont été reprojctées dans *PostgreSQL* via Pg Admin III, puis dans un second temps, des contraintes ont été ajoutées aux tables géométriques pour qu'elles ne reçoivent que la bonne projection.

Les données qui sont manipulées dans l'application sont stockées dans deux types de représentations : les tables physiques et les *vues* (tables virtuelles). Les vues permettent de rassembler des informations provenant de plusieurs tables afin d'optimiser les performances d'afficher de ses données et de simplifier le développement nécessaire pour accéder à la base de données.

2. Symfony2 et Doctrine

La mise en place du Framework *Symfony2* permet d'architecturer tout le côté serveur de l'application, c'est-à-dire tous les processus entre le côté client et la base de données. Son installation se fait en ligne de commande dans ce projet. Pour le détail de l'installation, voir le site de Symfony2⁴.

Symfony2 permet de structurer de manière fonctionnelle un projet par l'intermédiaire des **Bundles** (voir Annexe III) qui sont même temps interdépendants dans un projet et qui peuvent être génériques afin d'être portables dans un autre projet. Dans le projet câbles, trois bundles ont été créés:

CablesBundle : squelette de base qui gère toutes les fonctionnalités de l'application.

ExtBundle: gestion spécifiques des tables dictionnaires.

UsersBundle : gestion des tables utilisateurs. L'évolution de l'application au niveau régionale fait qu'elle sera utilisée par des partenaires multiples avec des profils d'accès différents aux fonctionnalités et aux données. Ce bundle permet de gérer tout le processus durant la connexion d'un utilisateur sur l'application.

Outre la structuration d'un projet, Symfony2 permet de facilement établir la connexion avec la base de données par la configuration fichier unique.

La communication entre Symfony2 et la base de données se fait par l'intermédiaire de l'ORM Doctrine (cf. schéma : #4, 5). Cette bibliothèque indépendante permet de gérer le modèle de données (la base de données). Son but est de créer une image des tables et des vues de la base appelée *mapping* (voir Annexe IV) dans Symfony2 pour éviter l'utilisation des requêtes brutes. A partir de ce *mapping*, Symfony2 crée automatiquement des fichiers (classes d'entités) contenant des objets qui permettent de manipuler les données entre l'utilisateur et la base de données.

Certaines tables métier contiennent des informations géométriques (point, ligne, polygone). De base, Symfony2 via Doctrine ne permet pas gérer ce type de données. C'est pourquoi une

⁴<http://symfony.com/>

extension supplémentaire a été installée: *Doctrine2Spatial*. Elle permet à Symfony2 de reconnaître et à de traiter la géométrie venant de la base de données.

3. Requêtes et réponses HTTP

L'application a aussi bien été développée sur le modèle d'architecture *MVC** coté serveur (*Symfony2*) et coté client (*AngularJS*), ce qui facilite la communication entre les deux Framework mais aussi le développement sans avoir à développer tout le processus classique de communication HTTP entre le côté client et côté serveur.

Le modèle MVC permet de cloisonner le développement d'une application. Il s'agit d'une bonne pratique de développement, le travail qui traite l'apparence et l'interface du site (**Vue**, cf. schéma : #8) sera indépendant de la logique métier (**Modèle** cf. schéma : #3), lui-même indépendant du traitement des requêtes de l'utilisateur (**Contrôleur**, voir Annexe V) sans perdre toute la chaîne du fonctionnement de l'application.

Quand l'utilisateur fait une demande pour afficher des données (cf. schéma : #1), cette demande se fait sous forme d'adresse *URL**. Ces adresses sont définies dans le *routing* qui permet d'avoir des URL très simples (par exemple */cables/zonesensibles*). Les routes sont interprétées par les *contrôleurs* qui vont traiter l'information sous forme de demande (*requête HTTP** cf. schéma : #2) à la base de données et le résultat renvoyé (**Response http** cf. schéma : #7) par cette dernière.

Il y a plusieurs formats de requête HTTP ou réponse HTTP mais dans le cadre de l'application, c'est le format *GeoJSON ou JSON** qui a été utilisé respectivement selon que les données contenaient ou pas des informations sur la géométrie.

Le format *JSON* (ou GeoJSON) a été utilisé car la combinaison de son format (texte) et d'un encodage (sérialisation) des données permet une communication performante entre le côté client et le côté serveur, particulièrement pour les informations contenant de la géométrie qui augmente le volume du flux JSON.

II. Le côté client

Le côté client gère toute la partie interface homme-machine. Dans cette partie, le Framework **AngularJS** et la bibliothèque **Leaflet** ont été utilisés pour gérer toute la partie web et cartographie.

L'installation d'AngularJS et de Leaflet est simple. Il s'agit de copier leurs sources dans le répertoire de travail. Dans le projet, pour qu'ils fonctionnent, il faut y faire référence dans la page HTML principale de l'application (*index*).

1. Une application avec AngularJS

Une application développée avec AngularJS se base sur un ensemble de modules et de contrôleurs. Un **module** dans AngularJS définit l'application. C'est le conteneur global des différentes fonctionnalités de l'application. Le **contrôleur** gère toutes les demandes faites à la base de données et les réponses de celle-ci du côté serveur via le protocole de communication HTTP.

Dans le projet câbles, **cinq (5) modules** ont été créés:

- Le module principal de l'application (**app.js**) dans lequel tous les autres modules de l'application et les contrôleurs doivent être déclarés (voir **Annexe VI**).
- Le module des services (**services.js**) qui regroupe tous les services qui gèrent les communications HTTP avec le serveur et les services qui gère les données mises en cache pour les utiliser dans la session en cours sans avoir à envoyer des requêtes à la base de données.
- Le module de l'affichage (**display_directives.js**) qui gère l'affichage des messages d'information et d'erreur, l'affichage des listes et des détails des données métier.
- Le module des formulaires « **FormDirectives.js** » qui gère l'affichage, les contrôles de saisie et la soumission des données vers la base de données (voir **Annexe VII**).

2. La partie cartographie: Leaflet et Angular

Leaflet et *AngularJS* sont écrits tous les deux en JavaScript ce qui leur permet de se communiquer facilement. Leaflet a été utilisée pour développer « *l'élément carte* », c'est-à-dire la carte elle-même, l'affichage des couches et des fonctions de navigation...

Leur communication se fait par des modules et des contrôleurs. Un module gérant toute cette partie cartographie a été créé: le « *mapServices.js* ». Il regroupe des *services* qui sont des fonctionnalités mutualisées utilisables à différents endroits de l'application. Il regroupe aussi des *directives* qui exécutent des tâches (services) différentes dans l'application. Elles sont utilisées pour la manipulation du *DOM** (éléments de l'interface web) et se traduisent par des comportements HTML qui vont être réutilisables dans les pages (templates) de l'application.

Un service et trois directives ont été créés dans ce module (*mapService.js*) :

- Le service qui gère la configuration des couches raster (*WMS**) de références (*LeafletServices*) dont l'API *IGN*.
- La directive pour la gestion de la carte (*leafletMap*). Cette directive est la plus importante de l'application. C'est ici qu'on initialise la carte *Leaflet* et ses fonctionnalités comme les outils de navigation, la légende, les couches métiers...
- La directive qui gère les événements entre la carte et la liste de données (*maplist*). Par exemple, un élément sélectionné dans la carte est automatiquement sélectionné dans la liste de données et vice versa.

Les fonctionnalités par défaut de Leaflet ne sont pas suffisantes. Des extensions supplémentaires proposées par la communauté des développeurs ou développées spécifiquement pour l'application (légende) ont été ajoutées (voir tableau en **Annexe IX**).

3. L'ergonomie de l'application : Bootstrap et CSS

L'interface homme-machine doit être simple et instinctive, c'est pourquoi la réflexion sur l'ergonomie et l'aspect graphique d'une application est primordiale, même si souvent elle est négligée dans le processus de développement.

Dans cette application, un temps de réflexion important a été consacré à l'ergonomie et à l'aspect graphique. Le Framework Bootstrap permet de répondre à ces exigences. Il fournit un panel considérable de fonctionnalités courantes déjà développées telles que les *barres de navigation*, les *menus*, les *grilles* pour diviser facilement la page (carte et tableau de données par exemple), les *formulaire*s de saisies, les *pictogrammes*, les *boutons*, les *blocs pliables/dépliables*, les *fenêtres modales*...

Conclusion

Je rappelle qu'il me reste quelques semaines sur mon stage au moment de la rédaction de ce rapport afin d'avancer sur le développement des fonctionnalités et travailler sur l'ergonomie de l'application pour qu'elle réponde parfaitement aux besoins du Parc et des utilisateurs.

La cartographie en ligne prend aujourd'hui de l'ampleur. Elle permet de diffuser des données géo-référencées en ligne via un navigateur internet et de faire des analyses spatiales sans avoir besoin d'installer des outils sur des postes. Plusieurs structures comme le PNV et les parcs nationaux de France utilisent cette nouvelle avancée des SIG dans la réalisation d'application de type webmapping.

Ce stage m'a permis de mettre en application les enseignements acquis dans la formation Master1/Master2 SIG Système Territoire Environnement et Patrimoine (STEP). Il m'a été très bénéfique et ces apports sont résumés dans les points suivants :

1. Intérêt de la mission

L'intérêt de cette mission est double. Elle a permis dans un premier temps au Parc de disposer d'un outil réalisé avec les technologies open source et qui répond parfaitement aux besoins. L'outil, une fois terminé, certaines de ses fonctionnalités pourront être mutualisées pour la réalisation d'application de webmapping dans les autres parcs.

Le deuxième intérêt de la mission est personnel. Ce stage m'a permis d'abord de comprendre à nouveau l'intérêt des SIG dans une structure. Il m'a permis de mobiliser mes compétences acquises tant en SIG qu'en Informatique (développement web).

Le travail de réflexion fonctionnelle avec mon maître de stage et la responsable du projet câble, l'utilisation d'outils que je ne connaissais pas avant le stage et l'autonomie que j'avais pour effectuer mes missions, m'a permis d'acquérir des compétences tant au niveau de la définition d'un projet, qu'au développement applicatif et qu'au niveau organisationnel. Ceci est un avantage pour moi pour continuer à travailler dans le domaine du webmapping.

J'ai aussi appris beaucoup sur le fonctionnement d'un établissement public sur les relations humaines entre les agents des différents services. J'ai ainsi pu, pendant ma mission, développer mon sens du contact et mes capacités en communication ce qui est un atout important dans le monde professionnel. En travaillant dans une équipe pluridisciplinaire composée de beaucoup d'agents, j'ai réussi au bout de quelques semaines, à m'intégrer

facilement en trouvant ma place et en ne me considérant pas comme un stagiaire mais comme un agent à part entière du Parc national de la Vanoise.

2. Difficultés rencontrées

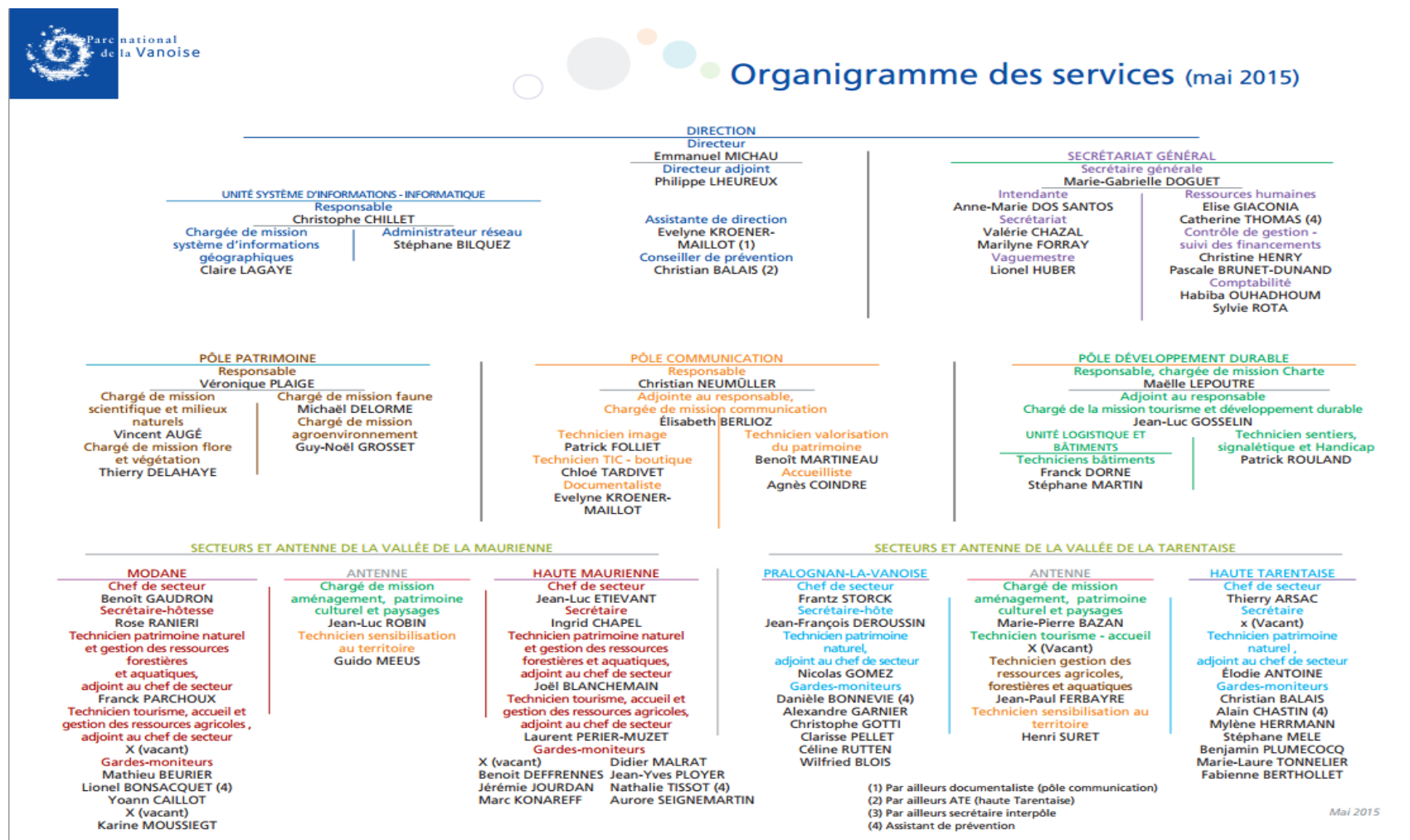
Les difficultés dans ce genre de mission sont tout à fait normales. Nous avons décidé d'utiliser des briques de développement du Parc national des Cévennes. Cela m'a permis de développer aisément les fonctionnalités de base. Cela m'a permis de monter en compétences non sans difficultés car la compréhension et la reprise d'un code même défini comme générique peut amener beaucoup de questions demandant des explications et de rencontrer beaucoup d'obstacles.

Le monde du développement offre des outils puissants mais il faut savoir les utiliser pour bien mener son projet à terme. La difficulté est de savoir comment bien traduire les besoins fonctionnels en lignes de code. Cela était le cas par exemple pour la mise en place de la légende dans l'application vu que ce n'est pas un élément de base dans la bibliothèque Leaflet.

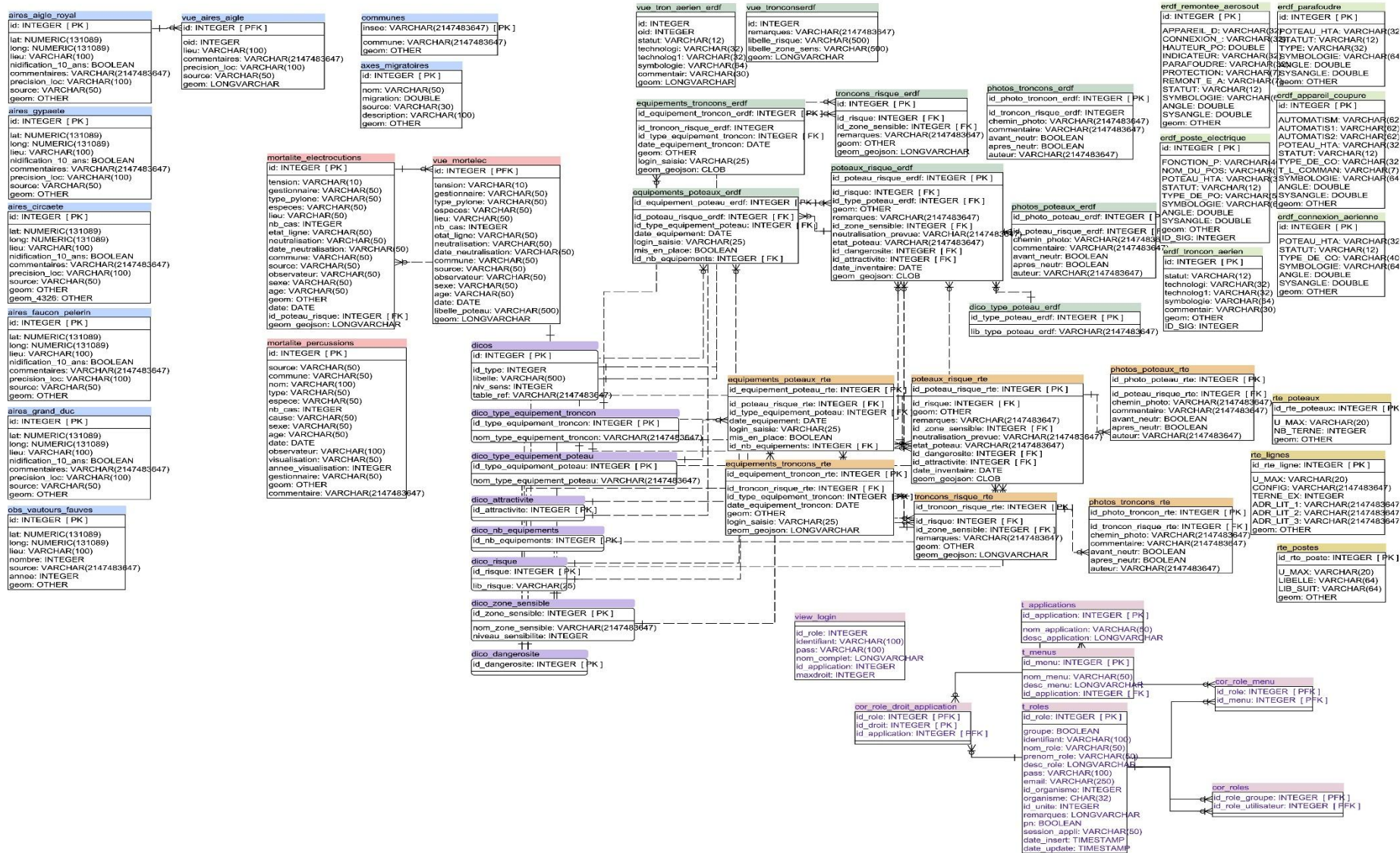
Annexes

Annexe I: organigramme du Parc national de la Vanoise (novembre 2014)	35
Annexe II: modèle de la base de données	36
Annexe III: création d'un bundle sur Symfony2	37
Annexe IV: mapping de la base de données avec Doctrine	38
Annexe V: exemple de contrôleurs	40
Annexe VI: extrait du module général de l'application (appCables)	41
Annexe VII: exemple de configuration d'un schéma	42
Annexe VIII: tableau des fonctionnalités	44

Annexe I: organigramme du Parc national de la Vanoise (novembre 2014)



Annexe II: modèle de la base de données



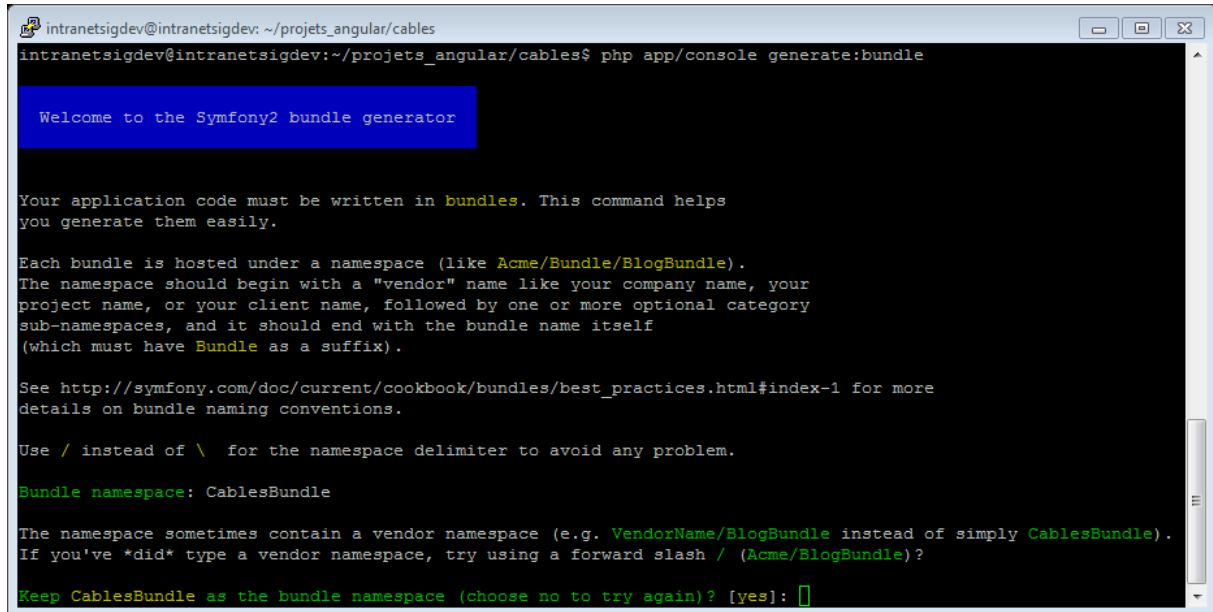
Annexe III: création d'un bundle sur Symfony2

Je montre ici un exemple de la création d'un squelette ou bundle sur Symfony2:

```
php app/console generate:bundle
```

Cette commande génère une nouvelle structure de bundle et l'active automatiquement dans le projet.

Il faut ensuite suivre les instructions de la commande qui nous demandera par la suite le nom du bundle, le format des fichiers de configurations...



```
intranetsigdev@intranetsigdev: ~/projets_angular/cables
intranetsigdev@intranetsigdev:~/projets_angular/cables$ php app/console generate:bundle

Welcome to the Symfony2 bundle generator

Your application code must be written in bundles. This command helps
you generate them easily.

Each bundle is hosted under a namespace (like Acme/Bundle/BlogBundle).
The namespace should begin with a "vendor" name like your company name, your
project name, or your client name, followed by one or more optional category
sub-namespaces, and it should end with the bundle name itself
(which must have Bundle as a suffix).

See http://symfony.com/doc/current/cookbook/bundles/best_practices.html#index-1 for more
details on bundle naming conventions.

Use / instead of \ for the namespace delimiter to avoid any problem.

Bundle namespace: CablesBundle

The namespace sometimes contain a vendor namespace (e.g. VendorName/BlogBundle instead of simply CablesBundle).
If you've *did* type a vendor namespace, try using a forward slash / (Acme/BlogBundle)?

Keep CablesBundle as the bundle namespace (choose no to try again)? [yes]:
```

Le bundle sera automatiquement ajouté dans le projet, par défaut dans le répertoire *src* du projet :

/home/intranetsigdev/projets_angular/cables/src/PNV		
Name	Size	Changed
..		20/07/2015 10:20:18
CablesBundle		27/07/2015 17:36:47
ExtBundle		01/07/2015 16:18:37
Utils		15/07/2015 09:12:20

Annexe IV: mapping de la base de données avec Doctrine

Il faut lancer une commande linux qui va créer l'image de la base de données dans le projet Symfony2 :

```
$ php app/console doctrine:mapping:convert yml
./src/PNV/CablesBundle/Resources/config/doctrine/metadata/orm --from-database --
force
```

Le fichier de métadonnées généré ressemble à ça :

```
1. PNV\CablesBundle\Entity\AiresView:
2.     type: entity
3.     table: cables.vue_aires_aigle
4.     schema: cables
5.     repositoryClass: PNV\CablesBundle\Repositories\AiresView
6.     id:
7.         id:
8.             type: integer
9.             id: true
10.            generator:
11.                strategy: AUTO
12.    fields:
13.        commentaires:
14.            type: string
15.        lieu:
16.            type: string
17.        source:
18.            type: string
19.        geom:
20.            type: json_array
```

Ce fichier représente l'image de la table ou de la vue dans la base de données et permet de générer par la suite les classes entités.

Les entités sont des classes métiers qui représentent chaque objet de notre application. Elles sont stockées par défaut dans le répertoire « Entity ». Pour les générer à partir d'une base existante, il faut exécuter deux requêtes :

```
$ php app/console doctrine:mapping:import PNVCablesBundles annotation
$ php app/console doctrine:generate:entities PNVCablesBundles
```

La première requête va importer le schéma depuis la base de données pour construire les classes d'entité. Doctrine va ainsi transformer les champs des tables de la base en propriétés privées (*private*).

La deuxième génère des méthodes qu'on appelle les « *getters* » et les « *setters* » pour ces propriétés déclarées. Le getter est un accesseur qui permet de récupérer la valeur d'une

propriété donnée (*GetId* par exemple) et le setter est un mutateur qui lui permet de définir cette valeur (*SetId* par exemple). Le fichier entité généré ressemble à ça :

```
1. <?php
2.
3. namespace PNV\CablesBundle\Entity;
4.
5. use Doctrine\ORM\Mapping as ORM;
6.
7. /**
8.  * AiresView
9.  */
10. class AiresView
11. {
12.     /**
13.      * @var integer
14.      */
15.     private $id;
16.
17.     /**
18.      * @var string
19.      */
20.     private $commentaires;
21.
22.     /**
23.      * Get id
24.      *
25.      * @return integer
26.      */
27.     public function getId()
28.     {
29.         return $this->id;
30.     }
31.
32.     /**
33.      * Set commentaires
34.      *
35.      * @param string $commentaires
36.      * @return AiresView
37.      */
38.     public function setCommentaires($commentaires)
39.     {
40.         $this->commentaires = $commentaires;
41.
42.         return $this;
43.     }
44.
45.     /**
46.      * Get commentaires
47.      *
48.      * @return string
49.      */
50.     public function getCommentaires()
51.     {
52.         return $this->commentaires;
53.     }
54. }
```


Annexe V: exemple de contrôleurs

- Exemple de type **NomController.php**

```
1. <?php
2.
3. namespace PNV\CablesBundle\Controller;
4.
5. use Symfony\Bundle\FrameworkBundle\Controller\Controller;
6.
7. use Symfony\Component\HttpKernel\Exception\AccessDeniedHttpException;
8.
9. use Symfony\Component\HttpFoundation\JsonResponse;
10. use Symfony\Component\HttpFoundation\Request;
11.
12. use Commons\Exceptions\DataObjectException;
13. use Commons\Exceptions\CascadeException;
14.
15. use Symfony\Component\HttpFoundation\BinaryFileResponse;
16.
17.
18. class AiresViewController extends Controller{
19.
20.     // path: GET /cables/aires
21.     public function listAction(){
22.         /*
23.          * retourne la liste des aires "cables"
24.          */
25.         $ss = $this->get('airesService');
26.
27.         return new JsonResponse($ss->getList());
28.     }
29. }
```

- Exemple de type **NomConfigController.php**

```
1. class AiresConfigController extends Controller{
2.
3.     // path : GET cables/config/aires/form
4.     public function getFormAction(){
5.
6.         $file = file_get_contents(__DIR__ . '/../Resources/clientConf/aires/form.yml
7.     ');
8.         $out = Yaml::parse($file);
9.
10.         return new JsonResponse($out);
11.     }
12. }
```

Ce fichier appel ainsi d'autres fichiers de configuration dans lequel sont définies les schémas pour nos listes, détails et formulaires.

Annexe VI: extrait du module général de l'application (appCables)

```
1. var app = angular.module('appCables', [ 'cablesControllers', 'baseMort', 'baseAires'  
  , 'baseTroncons', 'baseTronAerien', 'cablesServices', 'FormDirectives', 'DisplayDire  
  ctives', 'ui.bootstrap', 'darthwade.loading', 'mapServices', 'LocalStorageModule' ] );  
2.  
3. // module de gestion de la page d'accueil  
4. angular.module('cablesControllers', [ 'cablesServices', 'mapServices', 'ngRoute', 'ng  
  Table' ] );  
5.  
6. // module de gestion test mort  
7. angular.module('baseMort', [ 'cablesServices', 'mapServices', 'ngRoute', 'ngTable' ] );  
8.  
9. // module de gestion test aires  
10. angular.module('baseAires', [ 'cablesServices', 'mapServices', 'ngRoute', 'ngTable' ] )  
  ;  
11.  
12. // module de gestion test tronc  
13. angular.module('baseTroncons', [ 'cablesServices', 'mapServices', 'ngRoute', 'ngTable'  
  ] );  
14.  
15. // module de gestion test  
16. angular.module('baseTronAerien', [ 'cablesServices', 'mapServices', 'ngRoute', 'ngTabl  
  e' ] );  
17.  
18. // services de l'application  
19. angular.module('cablesServices', [ 'mapServices' ] );  
20.  
21. // directives formulaires  
22. angular.module('FormDirectives', [ 'angularFileUpload', 'mapServices' ] );  
23.  
24. // directives affichage  
25. angular.module('DisplayDirectives', [ 'mapServices' ] );  
26.  
27. // directives map  
28. angular.module('mapServices', [ 'cablesServices' ] );
```

Annexe VII: exemple de configuration d'un schéma

- Pour la liste

```
1. title: Zone Sensible Aigles Royal
2. emptyMsg: Aucune zone pour le moment
3. createBtnLabel: Nouvelle zone
4. createUrl: "#/cables/edit/aires/"
5. editUrl: "#/cables/edit/aires/"
6. detailUrl: "#/cables/aires/"
7. editAccess: 5
8. fields:
9.   - name: id
10.     label: ID
11.     filter:
12.       id: text
13.       options:Ã
14.       visible: false
15.
16.   - name: commentaires
17.     label: Commentaires
18.     filter:
19.       commentaires: text
20.     options:
21.       visible: true
22.   - name: lieu
23.     label: Lieu
24.     filter:
25.       lieu: text
26.     options:
27.       visible: true
28.   - name: source
29.     label: Source
30.     filter:
31.       source: text
32.     options:
33.       visible: true
```

- Pour le détail

```
1. editAccess: 3
2. subEditAccess: 2
3. groups:
4.   - name: Informations
5.     fields:
6.       - name: commentaires
7.         label: "Commentaires"
8.         type: string
9.       - name: lieu
10.        label: "Lieu"
11.        type: string
12.       - name: source
13.        label: "Source"
14.        type: string
15.        #help: ""
```

- Pour l'édition

```
1. deleteAccess: 5
2. groups:
3.   - name: Localisation
4.     fields:
5.       - name: geom
6.         label: Coordonnées GPS
7.         type: geom
8.         options:
9.           geometryType: circle
10.          dataUrl: cables/aires
11.   - name: Informations
12.     fields:
13.       - name: id
14.         label: ID
15.         type: hidden
16.       - name: source
17.         label: Source
18.         type: string
19.         options:
20.           maxLength: 250
21.       - name: lieu
22.         label: Lieu
23.         type: string
24.         options:
25.           maxLength: 250
```

Annexe VIII: tableau des fonctionnalités

	Fonctionnalité	Description
Légende dynamique	Pliable/dépliable	Une légende qui est pliable et dépliable sur la carte
	Légende transparente	Une légende transparente sur la carte
	Arborescence couche	Possibilité de gérer l'arborescence des couches
	Visibilité couche	Possibilité d'afficher ou masquer les couches
	Symbologie	Mettre des symboles sur différents sur les couches
	Fonds raster	Ajouter des couches rasters
	Session	Possibilité de garder les cases cochées et décochées
Données attributaires dans tableau	Pliable/dépliable	Un tableau pliable et dépliable depuis un bouton
	Interface	Garder la même interface en liste, détail et édition
	Filtre	Filtrer directement à partir du tableau de données
	Zoom	Faire un zoom sur l'élément filtré dans la carte
	Détail	Affichage de données en fonction du niveau de l'utilisateur
		<ul style="list-style-type: none"> Bouton Afficher pour montrer le détail d'un élément Bouton Editer pour modifier l'élément Bouton Supprimer sur tableau données et possibilités de supprimer en lot Ajout des boutons Modifier et Supprimer en mode détail
	Boutons	
Saisie	Ajout de données	Ajout de données depuis un bouton devant le tableau
		Enregistrer une donnée et rester sur la même page pour en recréer : <ul style="list-style-type: none"> on reste sur l'interface saisie et le formulaire est vidé + un bouton pour récupérer données saisies précédentes on reste sur l'interface saisie et le formulaire reste plein avec possibilité de supprimer données dans champs + bouton vider les données
	Enregistrer/nouveau	
	Enregistrer/fermer	Enregistrer une donnée et retourner sur la liste de données
	Point Déplaçable	Possibilité de pouvoir déplacer automatique la géométrie sur la carte sans cliquer sur Editer de Leaflet.
	Bloc collapsed	Regrouper les catégories de données dans des blocs pliables en mode création ou édition de données
	Légende	Possibilité de garder la légende pendant la saisie de données
	Interface	Garder la même interface qu'en liste et détail

	Boutons	<ul style="list-style-type: none"> • Bouton Supprimer • Bouton Annuler
Recherche	Recherche de données	<ul style="list-style-type: none"> • Critère géographique + critère métier + filtre métier dans tableau résultats • Critère géographique + filtre métier dans tableau résultats • Récapitulatif des critères de recherche depuis le tableau résultat
Fichier	Photos	Possibilité d'ajouter des photos en mode saisie
Sortie	Impression/export	<ul style="list-style-type: none"> • Possibilité d'exporter le tableau de données • Exporter la carte sous divers formats

Bibliographie

Bulletin de la Société Française de Photogrammétrie et de Télédétection (SFPT), 1989.

Joliveau T. (1996). Gérer l'environnement avec des S.I.G. Mais qu'est ce qu'un S.I.G. ?
Revue de Géographie de Lyon 71 (2/96) : 101-110

Webographie

Parc national de la Vanoise

<http://www.vanoise-parcnational.fr/>

Symfony

<http://symfony.com/>

AnularJS

<https://angularjs.org/>

PostgreSQL

<http://www.postgresql.org/>

PotsGIS

<http://www.postgis.fr/>

GitHub

<https://github.com>

PHP

<https://secure.php.net/>

Bootstrap

<http://getbootstrap.com/>

API Geoportail

<http://api.ign.fr/accueil>

GeoRezo

<http://georezo.net/>