Rapport d'analyse du projet développement

Mise en place d'une infrastructure de diffusion des données cartographiques au Muséum National d'Histoire Naturelle

Nom du projet :

BBCH Bioviewer

Réalisé par :

Walid HOUFAF-KHOUFAF
Gaspard COTHIAS-FAURE
Amina BARMANI
Florentin BRISEBARD

Année universitaire 2019-2020

Contexte du projet :

Ce projet de développement web a été commandité par Frédéric Vest, Nicolas Boulain et Guillaume Grech pour l'UMS Patrimoine Naturel, qui est une unité officiant sous la tutelle du Muséum National d'Histoire Naturelle (MNHN), du Centre National de la Recherche Scientifique et de l'Office Français de la biodiversité. Le projet sera en partie mené dans les locaux de l'UMS PatriNat se situant à côté du Muséum National d'Histoire Naturel à Paris, ainsi qu'à l'ENSG.

Dans le cadre de ses activités, l'UMS PatriNat diffuse de nombreuses données cartographiques au grand public. Ces données sont accessibles via un visionneur cartographique sur le site de l'INPN (Inventaire National du Patrimoine Naturel). Ce visionneur n'a pas été réalisé en collaboration avec l'UMS PatriNat, ainsi son fonctionnement et ses performances ne sont pas en adéquation avec les souhaits de ce dernier. De cette manière, l'unité voudrait avoir une infrastructure de diffusion de données commune à l'ensemble du MNHN, en utilisant des logiciels de cartographie web tels que Geoserver et GeoNetwork. Pour se faire nous disposons d'une base de données contenant l'ensemble des informations relatives au patrimoine naturel.

L'interface de visualisation cartographique actuel fonctionne à l'aide de données au format GeoJSON, or ce mode de diffusion ne respecte pas les standards de l'OGC, de plus il est uniquement vectoriel et de nombreuses fonctionnalités disponibles grâce à Geoserver ne sont pas réalisables (comme par exemple de pouvoir effectuer des généralisations plus ou moins précises en fonction du zoom).

Finalement, la création d'un composant web a été choisi par l'UMS comme étant la solution la plus optimale. Si la réalisation se passe bien et que le composant est généralisable, alors il pourra être utilisé sur d'autres projets de visualisation de données cartographiques et pourra faire gagner un temps certain. Néanmoins s'il venait à ne pas aboutir, cela ne constituerait pas un problème grave pour l'unité.

Le projet ne nécessite pas l'achat de licence puisque les quelques données extérieures (notamment les fonds de cartes) proviendront d'open-sources, ainsi le coût monétaire du projet est nul. Dans la suite du projet, nous envisageons de séparer le travail en différentes tâches réparties entre les membres. Néanmoins, les décisions sur les potentiels futurs choix seront prises ensemble et certaines parties du développement pourront être effectuées à plusieurs.

Objectifs du projet:

L'objectif principal de ce projet est à terme de livrer une interface de visualisation inspirée de celle déjà existante qui soit à la fois plus optimisée et performante. En effet dans un premier temps, l'objectif de notre projet porte sur les modalités de diffusion de la donnée existante, la diffusion de la donnée doit être plus rapide et optimisée (par exemple utilisation d'une image png ou d'un vecteur pour afficher dans le viewer). La diffusion de la donnée devra aussi offrir un contrôle sur le catalogue de données accessibles ou non, le client souhaite pouvoir appliquer des restreintes sur l'accessibilité au catalogue.

Concernant l'interface de visualisation en elle-même, ayant été réalisée par les développeurs sans la collaboration étroite de l'UMS Patrinat, elle n'est pas complètement satisfaisante pour le client, par rapport à l'utilisation qui est faite de la donnée par les utilisateurs. L'UMS Patrinat souhaite que la nouvelle interface d'utilisation soit plus facile et intuitive à manier, elle devra intégrer un moteur de recherche intelligent qui permette à l'utilisateur de rechercher sa couche par mots-clés de manière aisée. L'interface de visualisation devra surtout permettre des affichages et un accès aux métadonnées idoines par rapport à l'utilisation qui est faite par les utilisateurs de ces données.

Le dernier objectif essentiel du projet est sa répétabilité: le projet doit être amplement documenté de la création de l'environnement de travail jusque l'utilisation de l'interface de visualisation afin que l'interface puisse être utilisée à l'avenir pour d'autres bases de données de manière souple en suivant simplement la documentation produite lors de ce projet.

Le client étant familier avec les spécifications techniques du projet, ses demandes sont exprimées de manière claire, explicite et détaillée. Ainsi de nombreuses contraintes de développement font partie inhérentes du projet du fait de la volonté très détaillée du client jusque dans le détail des logiciels à utiliser pour l'environnement de travail. Ainsi concernant l'environnement de travail, nous devons mettre en place sur nos PC, le même environnement de travail que celui utilisé à l'UMS-Patrinat dans un souci de portabilité aisée du projet, ainsi nous devons utiliser les mêmes versions de Geoserver, GeoNetwork, pgAdmin et Solr.

Concernant le développement web, il devra se faire en utilisant le framework Angular et en utilisant exclusivement des fonds de cartes et cartes interactives open Source, donc par exemple pas Google Maps qui à partir d'un certain nombre de visites sur le site facturerait l'UMS-Patrinat.

- **Le recueil du besoin**

Le projet a deux destinataires principaux: l'UMS-Patrinat qui gère les bases de données et contrôle l'accès aux bases de données depuis l'interface de visualisation. Le second destinataire est évidemment le futur utilisateur du site, son besoin est d'avoir accès à une information géographique claire de qualité, facile d'utilisation et d'extraction. L'expérience de l'utilisateur sur le site doit être agréable, et lui permettre de chercher la couche souhaitée par une simple entrée de mots-clés liés à la couche.

Le diagramme de cas d'utilisation ci-dessous illustre en détail les besoins des acteurs :

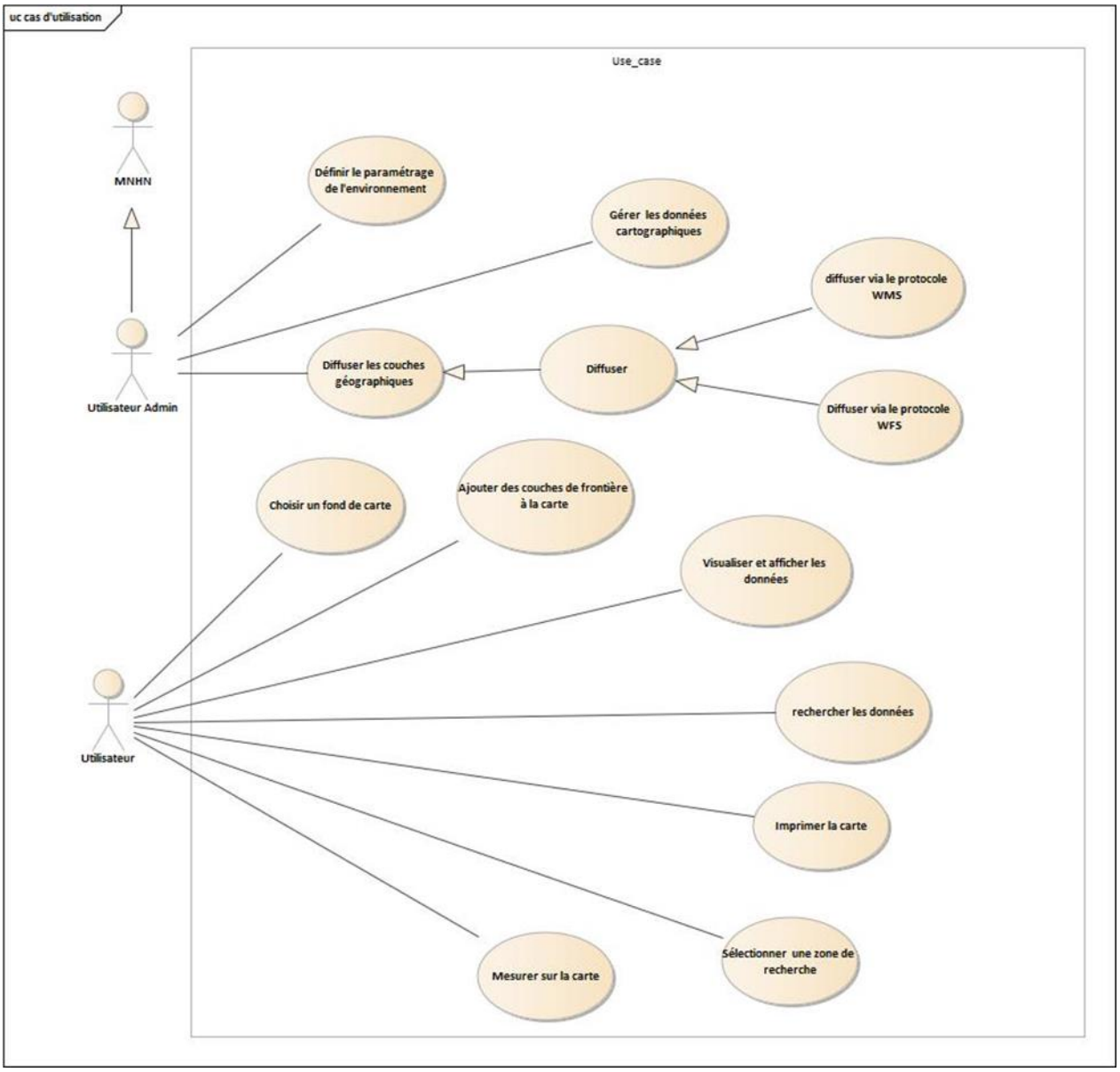


Figure1 : diagramme de cas d'utilisation

Analyse fonctionnelle :

L'étape primordiale de l'analyse pour un projet de développement est de comprendre la structure interne du système et pouvoir distinguer ses différents éléments et leurs interactions.

Nos commanditaires ont proposé au début de ce projet un environnement de diffusion constitué de 4 logiciels :

- ✓ PgAdmin 4 avec installation de postgreSQL et PostGIS
- ✓ Geoserver avec un ensemble de plugins à installer
- ✓ GeoNetwork
- ✓ SOLR

L'architecture que nos commanditaires percevaient au début est la suivante :

Les couches et les données cartographiques stockées dans la base de données Postgres seront publiées sur Geoserver par une connexion Postgis en paramétrant les différents protocoles de diffusion et de publication et en indexant les couches par indexation SOLR. Le logiciel GeoNetwork permet d'associer aux flux Geoserver (WMS/WFS…) des fiches de métadonnées. L'application Angular récupérera les données et leurs métadonnées par requêtage sur GeoNetwork.

L'utilisation des 2 logiciels GeoNetwork et SOLR a été abandonné vers la fin du projet. Nous avons pu simplifier cet environnement et juste utiliser le logiciel Geoserver pour la publication des couches en lui ajoutant le plugin CSW.

Lors de la phase d'analyse de besoin, nous avons pu aussi identifier les différentes fonctionnalités que nous allons développer :

- **Fonctionnalités du viewer cartographique:**

  Les fonctionnalités peuvent être regroupé en 3 catégories :

  Module carte : qui représente les fonctionnalités générales :
  - ✓ Zoom
  - ✓ Mesure sur la carte
  - ✓ Choisir un fond de carte
  - ✓ Impression de la carte courante
  - ✓ Afficher frontières administratives des Pays/Régions/départements
  - ✓ Sélection d'éléments d'une couche à partir d'une zone tracée

  Module Catalogue /Affichage :
  - ✓ Afficher les couches en récupérant les protocoles de diffusion
  - ✓ Supprimer les couches ajoutées à la carte

  Module recherche :

- ✓ Recherche de couches par nom et type
- ✓ Supprimer le résultat de la recherche

Le diagramme de séquence permet de spécifier les interactions entre Geoserver et l'interface utilisateur en ce qui concerne les fonctionnalités d'affichage et la recherche des couches :

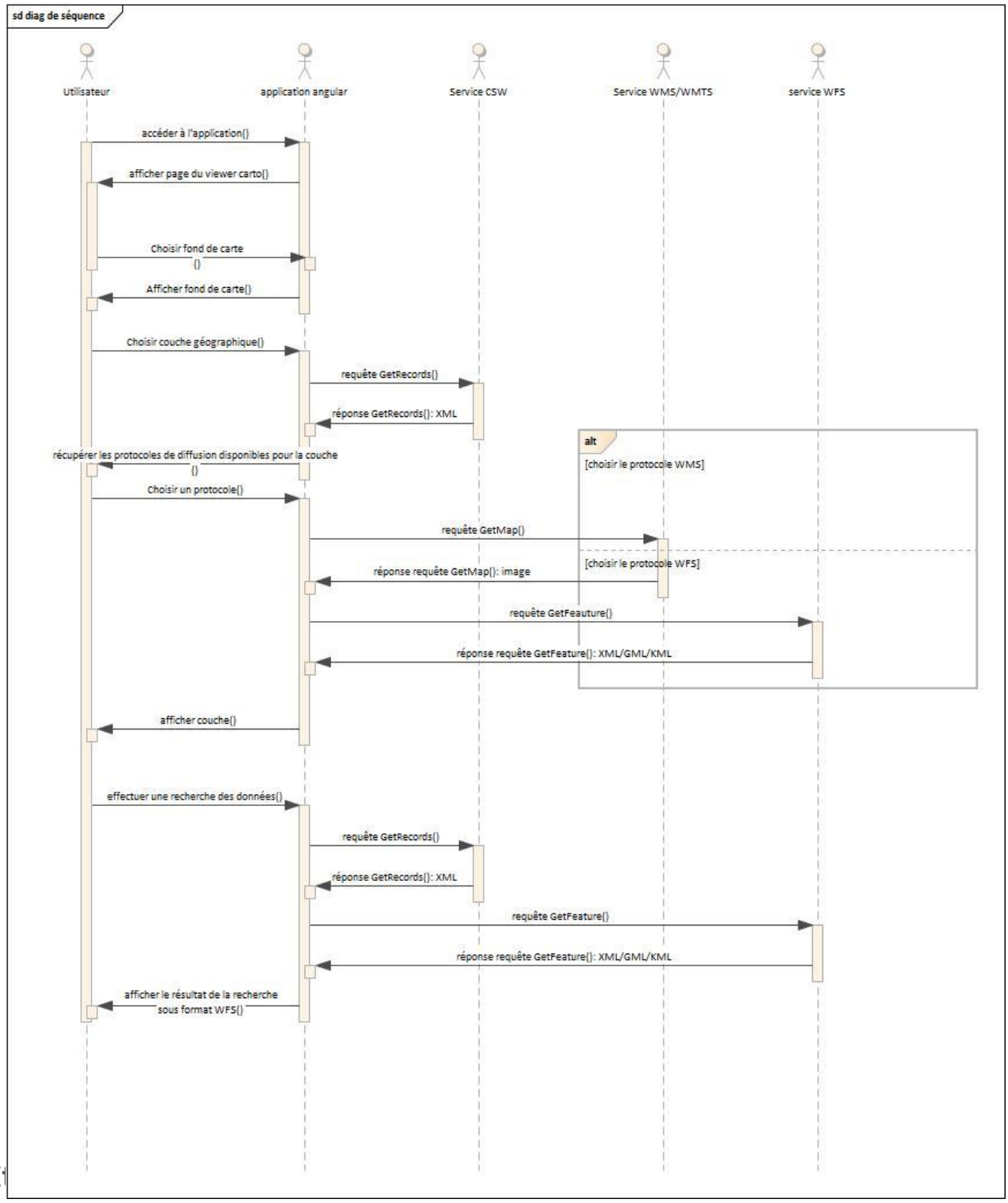


Figure2 : diagramme de séquence

Les choix d'outils technologiques que nous allons utiliser dans ce projet sont très limités. En effet comme le projet dérive d'un composant déjà existant qu'il faut renouveler, l'équipe de l'UMS savait par avance quelles solutions technologiques étaient nécessaires. De plus, l'unité utilise déjà de nombreux outils et il convient de s'adapter à la façon dont ils souhaitent travailler.

Ainsi le composant web sera développé en Angular (Framework de JavaScript), qui est utilisé sur toutes les autres applications web réalisé par le pôle développement. De plus, cette solution se prête bien à la création d'un web component (et non pas d'un site complet).

- **Architecture globale:**

Le viewer cartographique sera développé en utilisant Angular-CLI, il va alors s'occuper de compiler l'ensemble du projet et de lancer un serveur web pour visualiser l'application Angular (sur le port 4200 en localhost). Pour ce projet, nous utilisons une base de données PostgreSQL/Postgis pour les couches et les données géographiques qui seront publiées sur le serveur cartographique (Geoserver). Nous pouvons récupérer les couches publiées sur le Geoserver en utilisant les requêtes WMS/WFS… selon le besoin d'utilisateur.

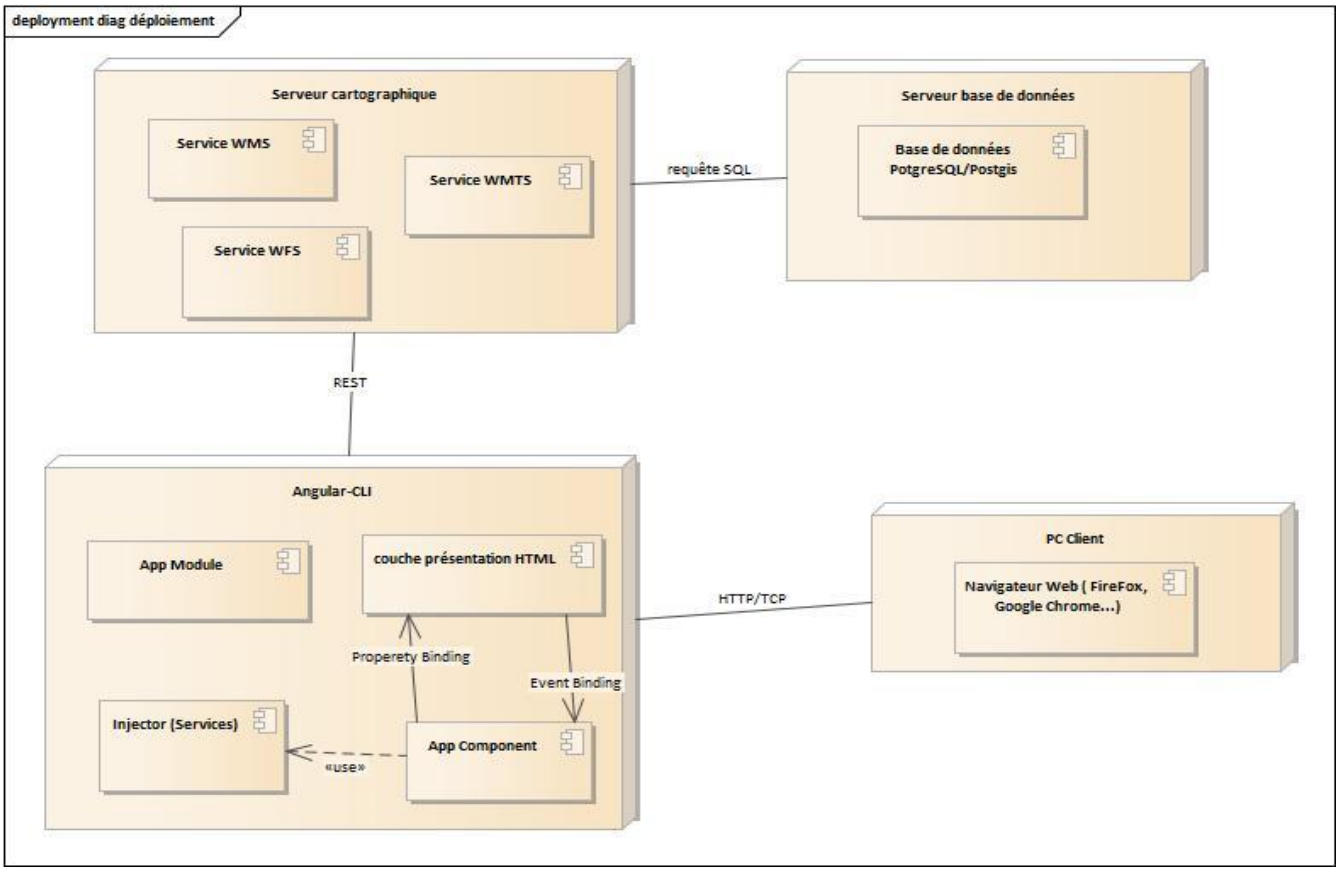


Figure3 : diagramme de déploiement

- **Architecture de l'interface utilisateur :**

Au niveau de l'architecture visuelle de l'interface de visualisation, nous allons nous inspirer de l'interface déjà existant de l'INPN (voir ci-dessous), car elle correspond à un standard ergonomique très utilisé et apprécié par de nombreux utilisateurs.

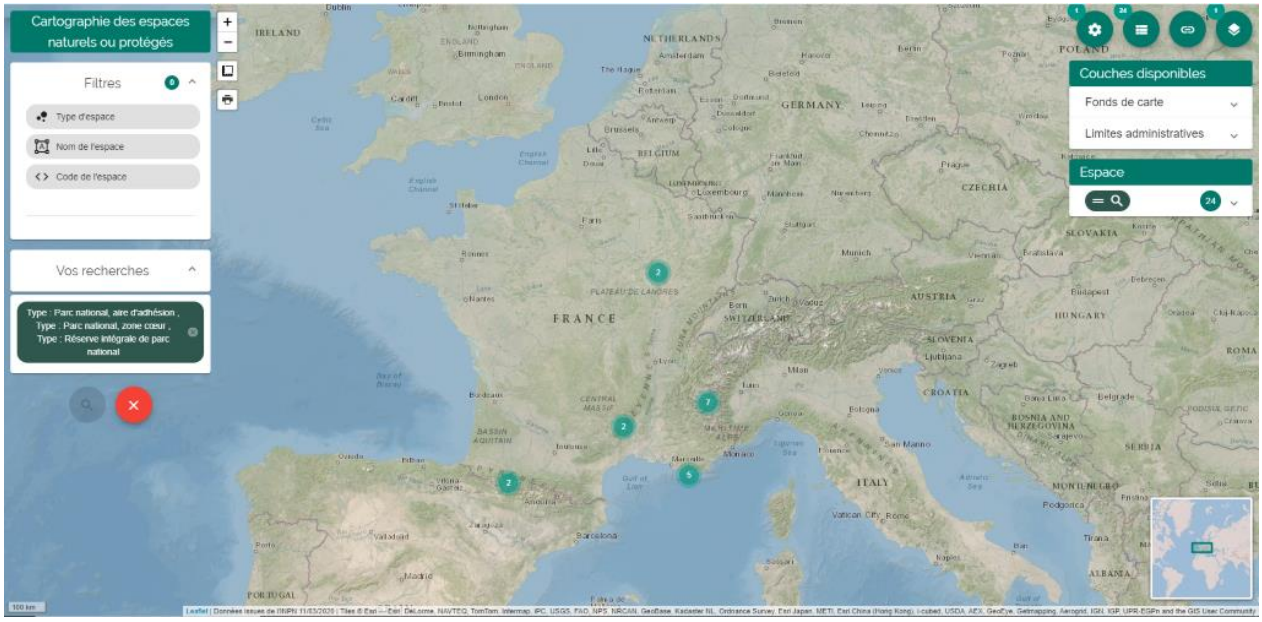


Figure4 : *Capture d'écran du site* https://inpn.mnhn.fr/viewer-

Néanmoins le client nous a souligné plusieurs imperfections de l'outil actuel, qu'il espère ne pas retrouver dans le viewer que nous lui livrerons. Parmi les imperfections principales, se trouvent la relative lenteur dans l'exécution du viewer, lorsqu'on cherche à multiplier les zooms et naviguer sur la carte, certaines latences significatives et désagréables pour l'utilisateur apparaissent. Une autre imperfection à corriger dans notre viewer est l'affichage en cluster des couches sélectionnées par l'utilisateur. Les couches se regroupent en cluster en dessous d'un niveau de zoom trop élevé. Ainsi pour des couches relativement grandes, il est impossible de les visualiser dans leur entièreté, puisqu'en dézoomant, elles vont disparaître pour faire apparaître le cluster qui constitue une information relativement peu intéressante pour l'utilisateur. Il est apparu que les couches qui constituaient un cluster pouvaient s'avérer trop dispersées, et ainsi lors du zoom se dissocier trop loin l'une de l'autre rendant la visualisation peu pratique.

Concernant la recherche de couche actuelle, elle apparaît peu satisfaisant aux yeux du client. Fonctionnant par check box à sélectionner, il convient de manière relativement peu intuitive une fois la couche sélectionnée de relancer une recherche sans que cela ne soit explicitée par le viewer.

Au niveau des fonds de carte de notre interface, nous allons proposer à l'utilisateur d'utiliser le fond de carte de son choix parmi tous ceux disponibles.

Évaluation des services et tests de performance :

Avant de commencer tout développement, les clients nous ont demandé d'évaluer les performances actuelles avec leur base de données complète. Cette première phase de test avait pour objectif de quantifier les performances de requêtages afin d'en mesurer la marge de progression, et les progrès prioritaires à réaliser. Les tests vont nous permettre aussi d'identifier les services les plus adaptés à notre projet pour réduire le temps de réponse du serveur et du navigateur et assurer une navigation fluide sur la carte pour l'utilisateur.

Nous avons réalisé plusieurs tests:

- ✓ Test de généralisation des couches géographiques à partir des fonctions Postgis.
- ✓ Test de configuration des caches et mise en place de couches tuilées.
- ✓ Test de montée en charge

○ Test de généralisation :

La première phase de tests concerne la généralisation. La généralisation se réalise par requêtage sur pgAdmin, on lance les tests sur les deux couches les plus importantes de la base de données metrop_znieff [1]1 et 2. Les fonctions ST_Simplify et St_SimplifyPreserveTopology nous ont servi à réaliser la généralisation, la différence entre les deux étant que ST_SimplifyPreserveTopology comme son nom l'indique réalise une généralisation en préservant la topologie c'est-à-dire que la géométrie des objets ne changent jamais, et les plus petits objets ne disparaissent pas. Pour nos tests nous avons décidé de faire varier la tolérance avec 3 valeurs : 10, 30 et 50m, et de comparer les résultats produits par ces deux fonctions avec différents niveaux de zoom.

✓ Résultats :

Temps d'exécution de chacune des requêtes (en millisecondes):

	Metrop_znieff1			Metrop_znieff2		
Tolérance	10	30	50	10	30	50
ST_Simplify	12s 626 ms	5s 346ms	11s 727ms	9s 57ms	7s 289ms	3s 536ms
ST_SimplifyPreserveTopology	27s 962ms	23s 522ms	22s 287ms	25 s 287ms	19s 700ms	19s 837ms

On compare les résultats de généralisation sur znieff1 généralisé avec la fonction ST_Simplify pour les polygones, on note à partir de quel niveau de zoom, les effets de la généralisation apparaissent :

[1] Les couches metrop_znieff représentent les Zones Naturelles d'Intérêt Écologique, Faunistique et Floristique. Nous effectuons des tests sur ces couches car ils représentent les couches les plus volumineuse de la BDD de MNHN.

Znieff1		
	ST_Simplify	ST_SimplifyPreserveTopo
10	1 :17k	1 :17k
30	1 :68k	1 :68k
50	1 : 136k	1 :136k

Znieff2		
	ST_Simplify	ST_SimplifyPreserveTopo
10	1 :8521	1 :34k
30	1 :34k	1 :68k
50	1 : 68k	1 :136k

Ainsi on a pu voir que la fonction **SimplifyPreserveTopology** remplit mieux la demande du client vis-à-vis de son viewer, avec une généralisation qui apparaît plus tôt et ne fait pas disparaître les petits objets, son temps de requêtage apparaît assez élevé, ce qui est un vrai souci de performance, que le futur viewer devra pallier.

- **Test de mise en cache et tuilage :**

Le 2ème test concerne la mise en cache et le tuilage des couches sur le Geoserver. Nous testons le temps de réponse du serveur et du navigateur pour la requête WMS en faisant varier le niveau de zoom

Nous avons réalisé les tests sur les 2 couches metrop_znieff1 et metrop_znieff2.

- ✓ **Résultats :**

Temps de réponse serveur (en millisecondes) pour la couche metrop_znieff1

Niveau de zoom	Temps de réponse (sans cache)	Temps de réponse (avec cache)	Temps réduit
1 :4M	332 ms	294 ms	10%
1 :2M	153 ms	145 ms	5%
1 :1M	78 ms	69 ms	11%
1 :545k	46 ms	45 ms	2%
1 :273k	35 ms	29 ms	17%
1 :136k	32 ms	25 ms	21%
1 :68k	38 ms	29 ms	23%

Temps de réponse serveur (en millisecondes) pour la couche metrop_znieff2

Niveau de zoom	Temps de réponse(sans cache)	Temps de réponse (avec cache)	Tems réduit
1 :4M	373 ms	294 ms	21%
1 :2M	215 ms	149 ms	31%
1 :1M	100 ms	72 ms	28%
1 :545k	64 ms	42 ms	34%
1 :273k	48 ms	35 ms	27%
1 :136k	41 ms	28 ms	31%
1 :68k	37 ms	26 ms	30%

Temps de réponse navigateur :

Pour la couche metrop_znieff1 et en testant le niveau de zoom 1/4M, nous obtenons les résultats suivants :

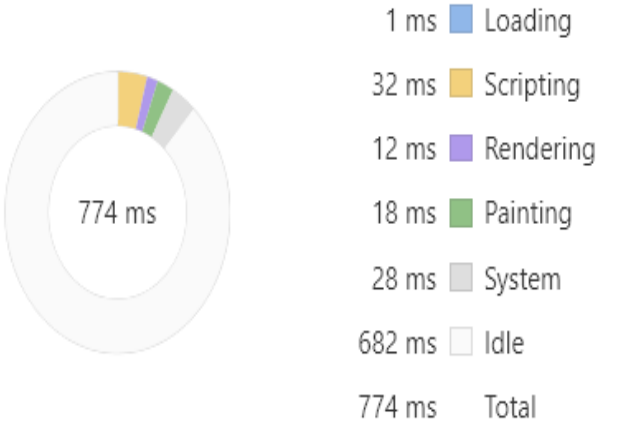


Figure5 : *Temps de réponse du navigateur sans mise en cache*

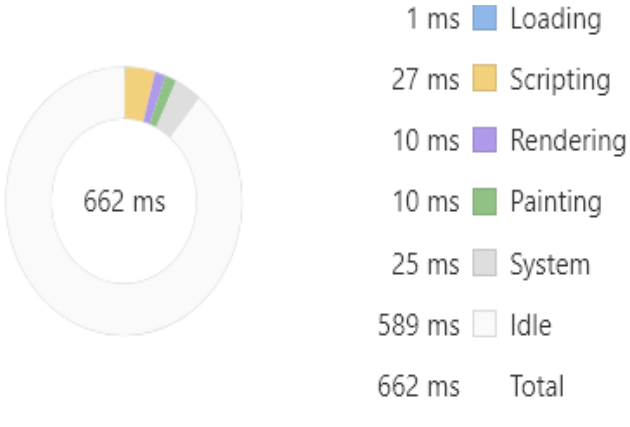


Figure6 : *Temps de réponse du navigateur avec mise en cache*

La mise en cache et le tuilage permet de réduire le temps de réponse du serveur (requêtes WMS) jusqu'à 35% surtout pour des niveaux de zoom plus précis ce qui va engendrer une navigation fluide sur la carte. On gagne aussi du temps au niveau de la réponse du navigateur.

- **Test de mise en cache et tuilage :**

Le 3ème test concerne la performance des requêtes au serveur. Pour cela nous avons pris en main le logiciel gratuit Gatling qui est un outil open-source de test de montée en charge et de performances pour les applications web. Cette application utilise le langage Scala pour créer des "scénarios" utilisateur – une suite d'actions effectuées les unes après les autres –, afin de tester les temps de réponses pour un nombre plus important d'utilisateurs.

Ces tests ont été réalisé sur la couche metrop_znieff1 et ses dérivées (différents types de généralisation). On s'intéresse surtout ici à étudier les différentes performances en fonction des types de requêtes, directement sur la plateforme Geoserver. Nous avons choisi un nombre de 50 utilisateurs simultanés.

- ✓ **Résultats :**

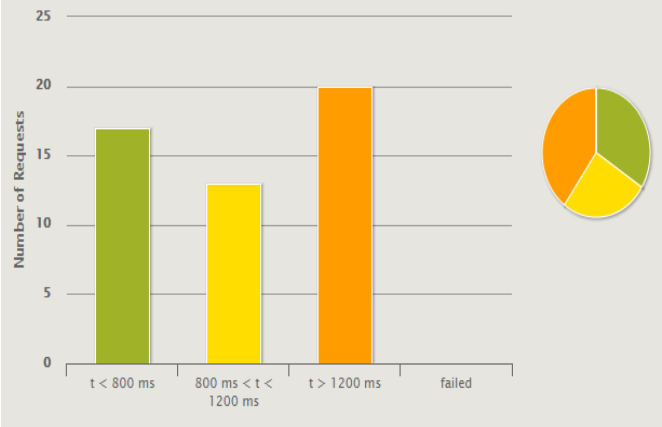


Figure7 : *Temps de réponse t pour* une requête WMS de la couche metrop_znieff1

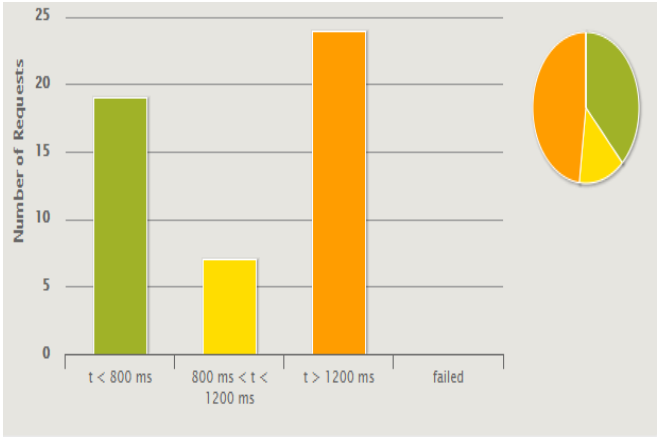


Figure8 : *Temps de réponse t pour* une requête WMS de la couche metrop_znieff1 généralisée à 100m

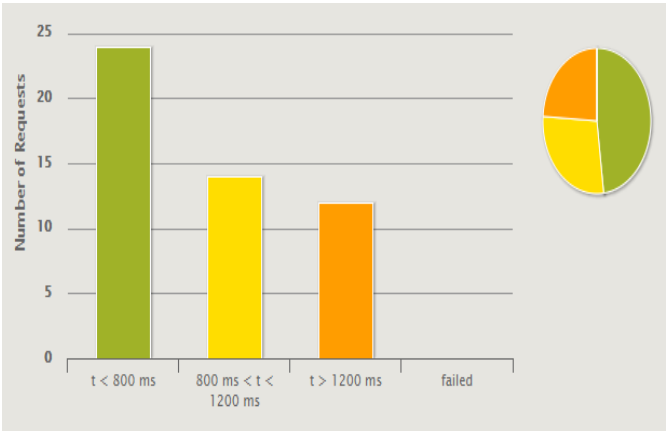


Figure9 : *Temps de réponse t pour* une requête WMS de la couche metrop_znieff1 généralisée à 100m en préservant la topologie

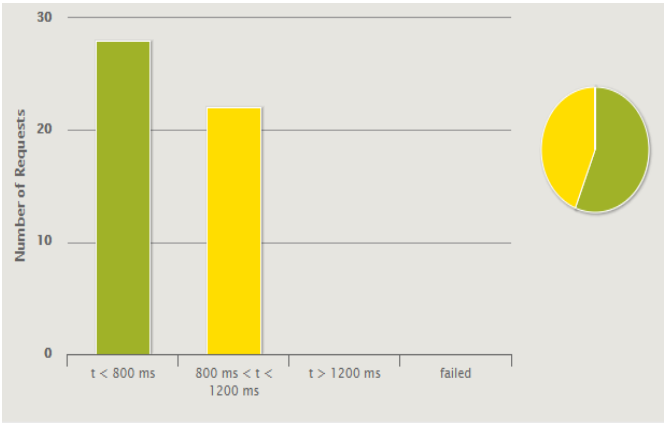


Figure10 : *Temps de réponse t pour* une requête WFS (en GML) de la couche metrop_znieff1

On remarque que la préservation de la topologie des objets permet un meilleur temps de réponse pour une requête WMS sur une couche ayant subi une généralisation. De la même manière il est

évident que la requête pour une couche généralisée obtient un temps de réponse moins élevé que pour la couche initiale.

Comme attendu, une requête WFS obtient une réponse du serveur bien plus rapide qu'une requête WMS.

Réalisation et suivi du projet

- **Les risques :**

Concernant les risques et difficultés liés au projet, le principal défi pour notre équipe de développeurs est d'appréhender plusieurs nouveaux logiciels jamais traités ou alors simplement entrevus lors de notre formation au sein de l'ENSG. Mais aussi pour le développement web de travailler avec un Framework (Angular) qui a simplement été évoqué lors de notre formation au développement web. Cette difficulté liée à notre inexpérience représente une véritable opportunité professionnalisante, mais aussi un temps supplémentaire à consacrer à la découverte et la prise en main de ces logiciels.

Pour gérer ce risque, nous avons très rapidement discuté avec nos commanditaires de l'étendu de nos compétences et savoir-faire. Compréhensifs et à l'écoute, ils nous ont envoyé vers la documentation des logiciels pour lesquelles, sa lecture suffisait à la prise en main, et ont aménagé un créneau pour nous former à l'utilisation d'Angular, framework dont l'utilisation est plus technique et ambitieuse.

Comme évoqué précédemment, cette formation et prise en main ont nécessité un temps supplémentaire pour devenir opérationnel, et pouvoir commencer à travailler au sein de notre environnement de travail. Ce temps pris a ainsi engendré un risque vis à vis des délais. Nous avons ainsi convenu de ménager du temps en dehors de ces créneaux pour travailler sur le projet et établi une hiérarchie de priorités des fonctionnalités à implémenter avec le client, pour assurer un produit fini qui propose les fonctionnalités jugées les plus essentielles par le client.

Afin de surmonter les risques, nous avons établi une matrice de risque qui regroupent tous les risques liés au projet et qui contient les solutions. Vous trouverez cette matrice en annexe(annexe1).

- **Le planning prévisionnel:**

Pour bien gérer notre projet de développement, nous avons identifié les grandes tâches à réaliser en précisant leur durée en jours. Nous avons établi un planning en utilisant le diagramme de Gant (annexe 2) pour qu'il nous aide à bien gérer notre temps et respecter les délais des rendus.

Nous avons aussi réalisé un diagramme de Pert pour organiser et coordonner l'enchainement des tâches (voir annexe 3).

- **Organisation du travail:**

Initialement, les commanditaires ont proposé de travailler le projet en utilisant une méthode agile basée sur de sprint de 15 jours avec un point de suivi hebdomadaire et une revue de sprint tous les 15 jours. Ils ont aussi proposé que nous travaillons dans leurs locaux une semaine sur deux.

Dans le contexte global de crise sanitaire. Nous avons prévu, en accord avec nos commanditaires, d'autres méthodes pour organiser le travail et communiquer à distance :

Nous avons utilisé un groupe privé sur un réseau social pour communiquer entre nous, ainsi que l'application Slack pour communiquer par écrit avec nos commanditaires, ou s'échanger des fichiers. Nous faisions chaque mercredi une réunion via StarLeaf ou Jitsi Meet pour s'assurer du bon déroulement du projet et communiquer de manière plus aisée sur nos différents problèmes rencontrés.

Nous avons utilisé Gitlab pour le développement en collaboratif.

Pour l'organisation, nous avons utilisé l'onglet *issues* sur Gitlab pour définir les tâches de chaque sprint.

Résultat du développement :

Après le développement du viewer cartographique, nous avons lancé le build du projet par commande et nous l'avons intégré dans un serveur HTTP

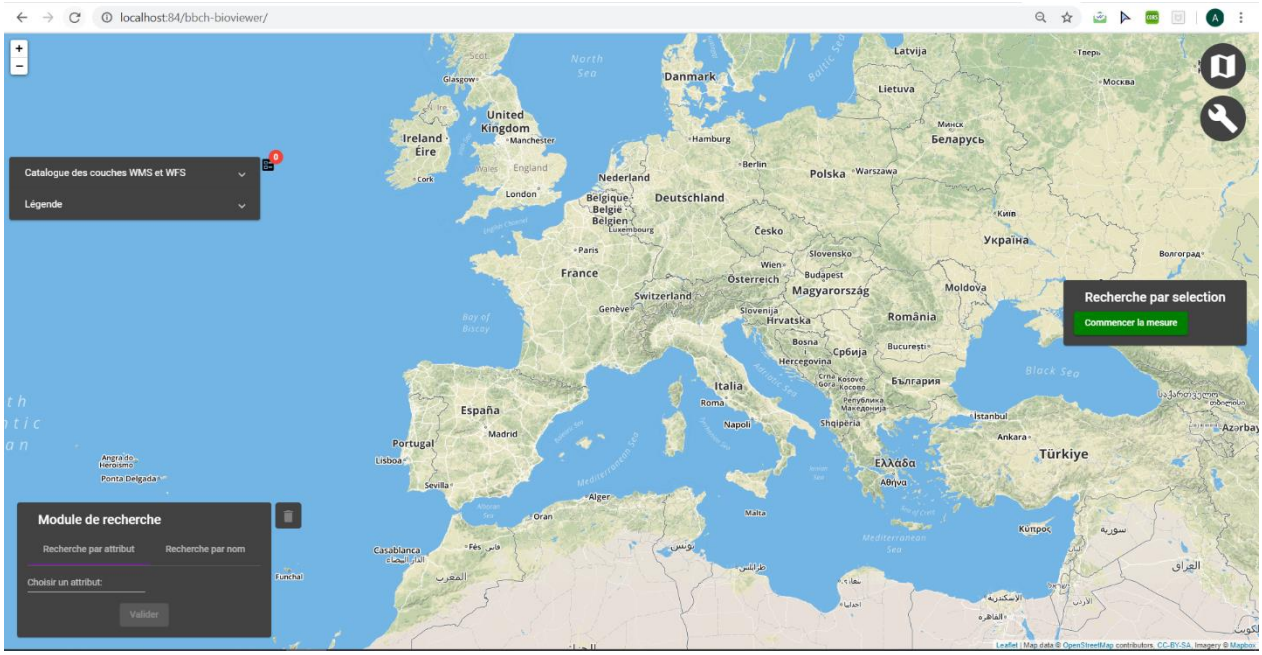


Figure11 : *Une capture d'écran du viewer développé après le chargement* de la page du navigateur

- **<u>Fonctionnalités développées :</u>**

Choisir un fond de carte : Cette fonctionnalité permet à l'utilisateur de choisir et changer le fond de carte Leaflet défini par défaut.

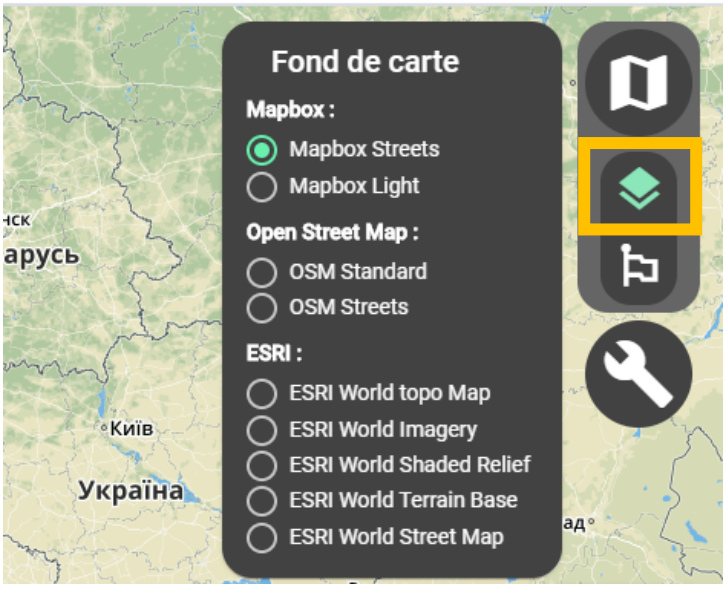


Figure12 : *Une capture d'écran du composant fond de carte*

Ajout des couches de frontières : cette fonctionnalité permet à l'utilisateur d'ajouter les 2 couches régions et département de la France

Figure13 : *Une capture d'écran du composant frontière en ajoutant les 2 couches sur la carte*

Mesurer sur la carte : cette fonctionnalité permet de mesurer sur la carte et calculer la distance totale des segments dessinés.

Figure14 : *Une capture d'écran d'un résultat de mesure de distance*

Imprimer la carte : cette fonctionnalité permet d'imprimer la carte en renseignant les champs d'impression

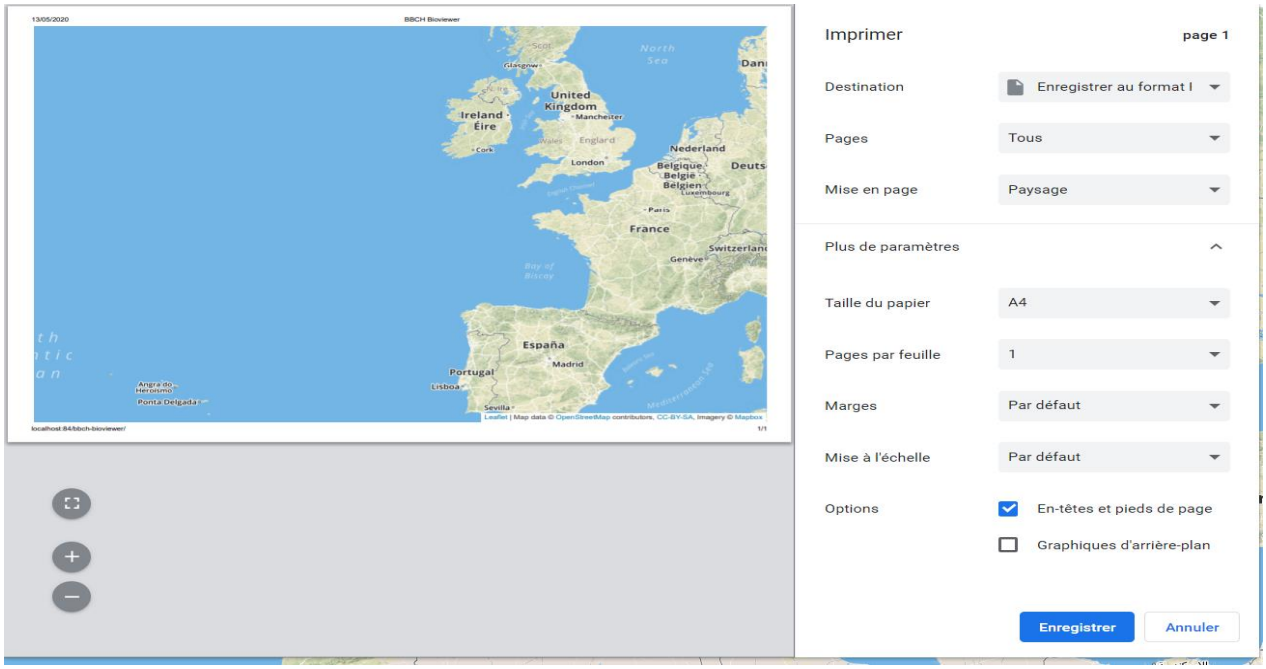


Figure14 : *Une capture d'écran de la fenêtre d'impression*

Catalogue des couches : Ce composant permet à l'utilisateur de récupérer toutes les couches diffusées via le Geoserver pour les afficher sur la carte. En choisissant la couche, on propose à l'utilisateur de choisir les protocoles de diffusion disponibles pour cette couche. Il permet aussi de supprimer les couches ajoutées à la carte dans la Légende.

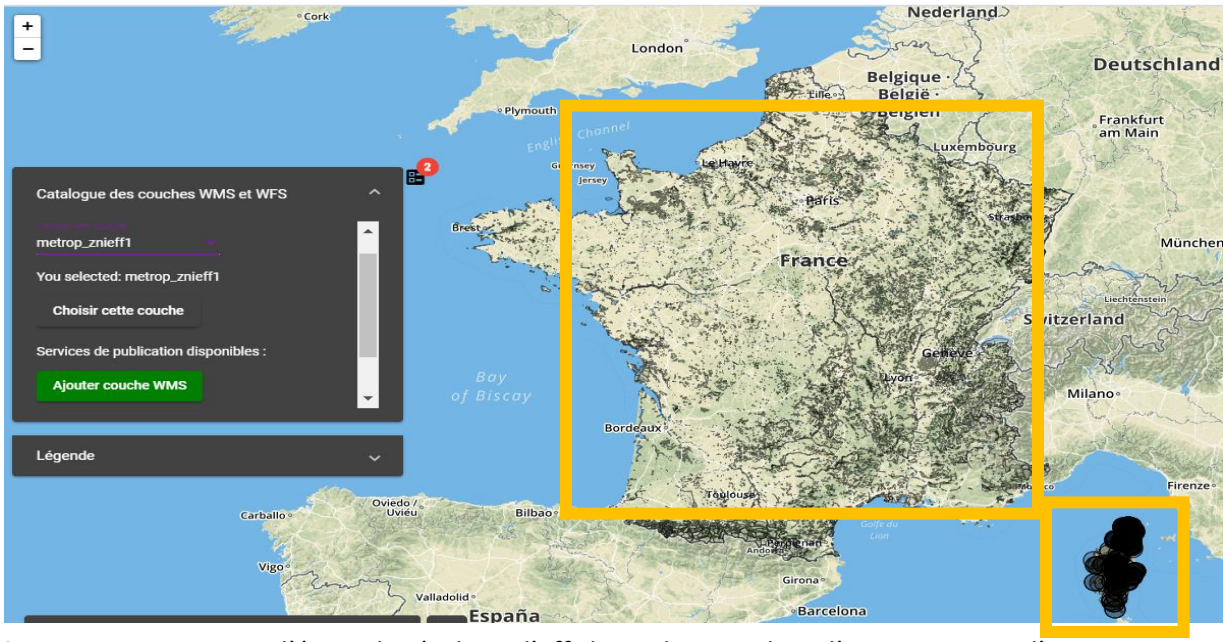


Figure15 : *Une capture d'écran du résultats d'affichage de 2 couches : l'une en WMS et l'autre en WFS*

Recherche des couches : ce composant permet 2 types de recherche : par nom de couche et par type

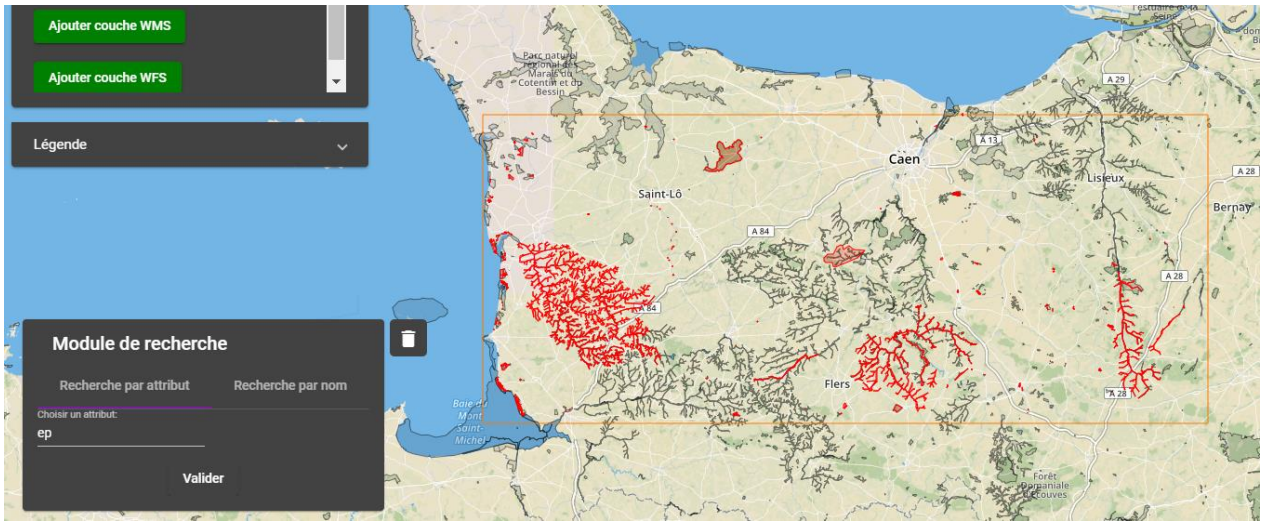


Figure16 : *Une capture d'écran du résultat de recherche par type espace protégé (ep)*

Sélection : l'utilisateur peut sélectionner une zone sur la carte et afficher le nom ou bien le type des couches disponibles

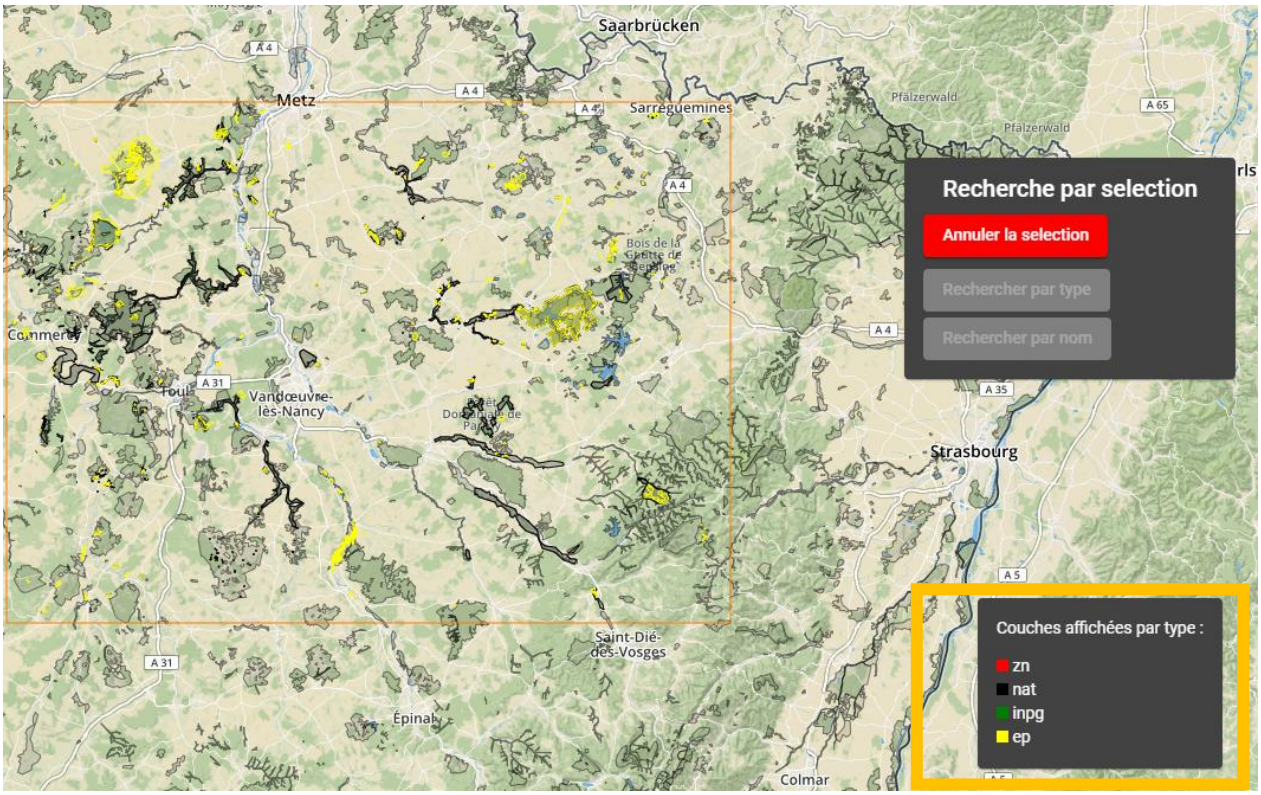


Figure16 : *Une capture d'écran du résultat d'une sélection*

Annexes :

Annexe 1 : matrice de risque

Nature de risque	Description	Gravité 1-4	Action en regard des risques
Risque sur les délais	Risque de dérapage sur le planning lié à une mauvaise estimation initiale de la durée nécessaire à l'exécution de chaque tâche	4	Vers la fin du projet prévoir une marge de temps pour la finalisation des tâches.
Risques intrinsèques à la gestion de projet	Mauvaise affectation des responsabilités/tâches aux membres du groupe	3	Discussion hebdomadaire sur l'avancement des tâches et réaffectation si nécessaire
Risque technique	Erreur et problèmes d'exécution des programmes/logiciels	3	Installer tous les logiciels nécessaires sur nos 4 ordinateurs pour avoir d'autres alternatives si un logiciel crash sur un ordinateur
Risque technique	Utilisation d'une nouvelle technologie pas encore maîtrisée par les membres du groupe	4	Évaluation du besoins et autoformation
Risque humain	Maladie de l'un des membres du groupe	2	Réaffectation des tâches sur les autres membres
Risques intrinsèques à la gestion de projet	Difficultés de communication avec nos commanditaires lors de la période de confinement	3	Utilisation des outils de communication comme Slack (réunions, messages…). Si l'un des commanditaires ne reçoit pas les messages sur Slack, nous utilisons les adresses mails personnelles.
Risques intrinsèques à la gestion de projet	Problème et erreurs générées par imbrication de code développé par chacun des membres	2	Envisager la création d'une nouvelle branche pour chaque fonctionnalité sur le Gitlab et réaliser un merge au fur et mesure du développement en corrigeant les bugs à chaque étape.
Risque en développement applicatif	Ne pas réussir à implémenter une fonctionnalité	4	Faire des recherches avancées et demander de l'aide aux commanditaires
Risque humain	Manque d'implication d'un membre de l'équipe	2	Lui parler pour identifier le problème et lui attribuer des tâches qui se sent plus à l'aise à faire
Risque humain	Scepticisme sur une tâche ou bien une démarche	1	Expliquer à tous les membres l'importance d'une telle tâche/démarche et les améliorations qui vont suivre
Risque Conception	Etablir une mauvaise conception du site et des composants	4	Revoir et discuter la conception du site avec les commanditaires au fur et à mesure de l'avancement du projet pour confirmer qu'elle correspond aux besoins demandés
Risques intrinsèques à la gestion de projet	Perturbation du déroulement du projet lors de la phase de confinement	4	Établir des modalités de travail à distance et réorganiser les tâches à faire
Risques intrinsèques aux changements de logiciels à utiliser	Décision de ne pas utiliser un logiciel prévu initialement	2	Concevoir une autre méthode pour suppléer ce logiciel avec un autre, ou bien avec une autre méthode

Annexe 2 : Diagrammes de Gantt

Diagramme de Gantt macroscopique du projet

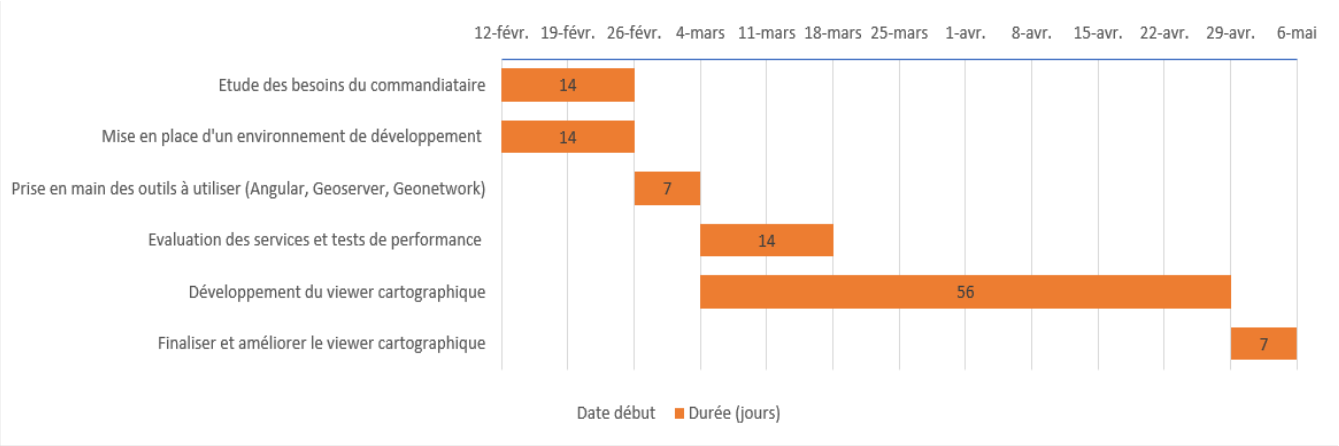
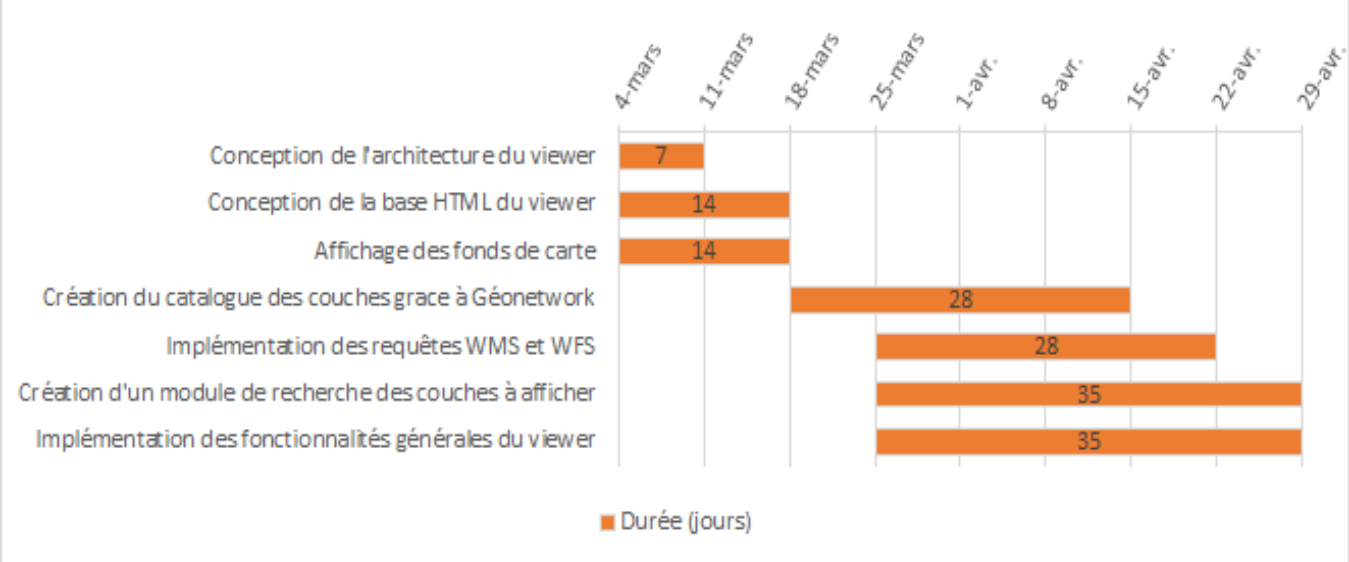
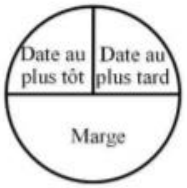


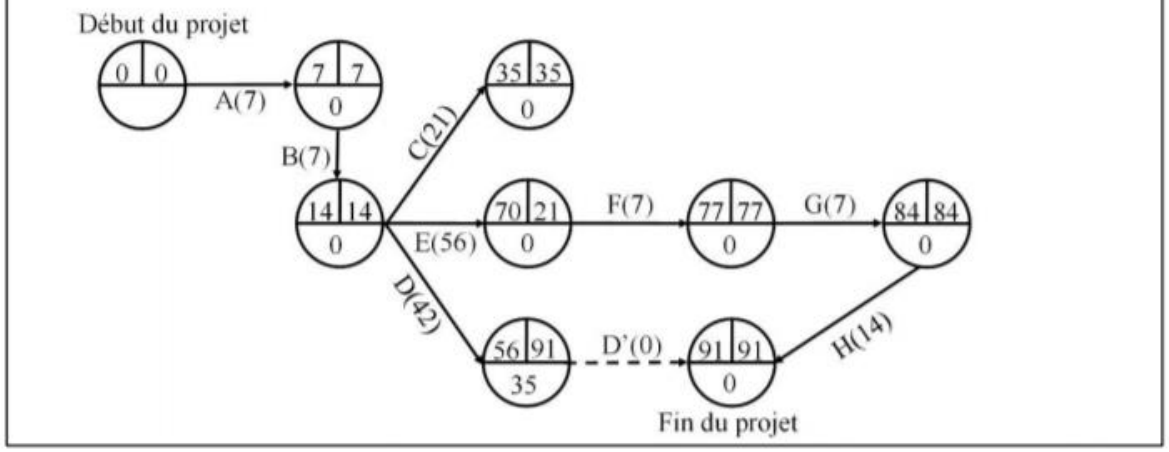
Diagramme de Gantt de la phase de développement du viewer



Annexe 3 : Diagrammes de Pert

Tâches	Nature	Durée (jours)	Antériorité
A	Étude des besoins du commanditaire	7	-
B	Mise en place d'un environnement de développement	7	A
C	Prise en main des outils	21	B
D	Évaluation des services et tests de performance	42	B
E	Développement du viewer cartographique	56	B
F	Amélioration et finalisation du viewer	7	E
G	Retouches et amélioration du viewer si necessaire	7	F
H	Préparation des livrables	14	D, E





Annexe 4 : Logo du projet

BBCH Bioviewer

Nous avons choisi ce nom en regroupant les 1ères lettres de nos noms et nous avons ajouté Bioviewer pour faire référence au domaine de la biodiversité et au viewer cartographique que nous allons développer.