

Projet Tuteuré FIE5

“Yuka des lieux bios”

Commanditaire : Fabien Guilloton

Tuteur école : Francis Faux

Membres du groupe :

En-Guady Anas

Majid Saifuldeen

Reghem Romain

Sommaire

I - Introduction-----	2
II - Gestion de projet-----	3
III - Contexte-----	4
IV - D'Angular à React-----	5
V - Amélioration de l'UX/UI-----	6
VI - Mise en place de la base de données-----	8
VII - Mise en place de l'API-----	10
VIII - Amélioration des performances-----	12
IX - Nouvelles fonctionnalités-----	12
X - Récapitulatif des calques-----	15
XI - Automatiser les mises à jour de la BDD-----	16
XII - Crédits et mentions-----	22
XIII - Type de monétisation-----	23
XVI - Les mentions obligatoires-----	26
XVII - Perspectives et conclusion-----	27
XVIII - Resources -----	28

I - Introduction

Yuka des lieux bios est un projet d'application web référençant sous forme de carte la géographie des exploitations biologiques, et des zones polluées du Tarn. Ce projet n'est pas commandité par une entreprise, il doit simplement dans un premier temps répondre aux besoins des apiculteurs soulevés par l'agence pour le développement de l'apiculture. L'objectif est d'abord d'aider les apiculteurs bio à respecter la loi qui leur impose de placer leurs ruchers de telle façon que dans un rayon de 3 kilomètres, les sources de nectar et de pollen soient constituées essentiellement de cultures bio.

Nous reprenons l'application que nous avons faite l'an dernier, afin d'améliorer son architecture, ses fonctionnalités et ses performances, ainsi que pour aller plus loin sur certains aspects non techniques comme la création d'un brevet.

II - Gestion de projet

Pour gérer ce projet, nous avons commencé par identifier les tâches à accomplir pour réaliser le projet et nous avons constaté que ce projet se compose de 9 tâches principales. Ensuite, nous avons divisé ces tâches entre notre équipe qui est composée de 3 membres (Romain, Saif et Anas) comme le montre le diagramme de gantt ci-dessous.

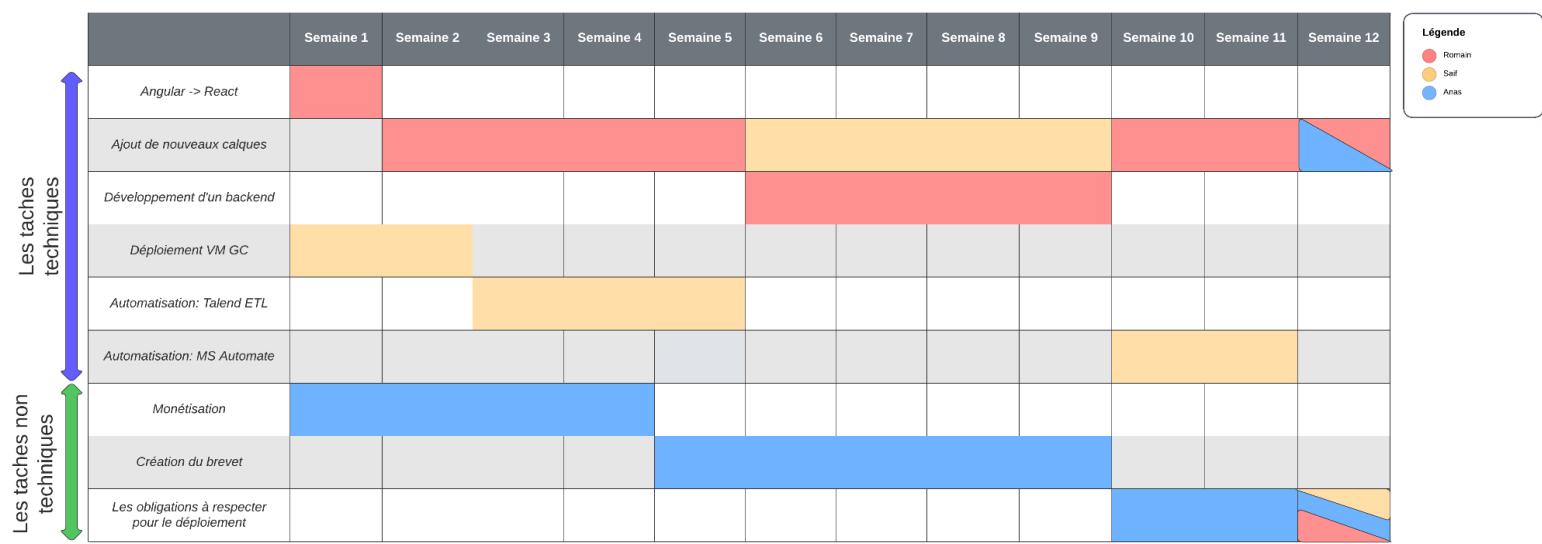


Fig 1. Diagramme de Gantt de la gestion de projet

III - Contexte et besoins

Cette année, nous avons décidé de continuer le projet de l'an dernier. Il nous avait bien plu, et nous savions qu'il y avait encore beaucoup de fonctionnalités à rajouter. Nous partons donc d'une application web destinée principalement aux apiculteurs. Cette application permet d'ajouter sur une carte des calques représentant des zones bios et des zones polluées, dans le but d'aider les professionnels à placer leurs ruches. L'utilisateur peut aussi ajouter des cercles de rayons personnalisés (pour la législation des 3km par exemple). L'an dernier, nous avions déjà un peu dépassé l'utilité aux apiculteurs en ajoutant la qualité des rivières par exemple.

Malgré tout, nous n'avons pas vocation à faire une application qui regrouperait toutes les données possibles trouvées sur internet. [Géoportail](#) le fait déjà, et notre application doit garder une plus value sur ce site. Notre plus-value provient de la facilité d'utilisation, qui n'est pas le cas de Géoportail, l'accessibilité, la concentration des informations dans le domaine du bio et de la pollution.

Au niveau des technologies, l'application utilisait le framework Angular, et la bibliothèque de cartographie Leaflet pour l'affichage de la carte et des calques. Les données étaient stockées au format GeoJSON dans les fichiers de l'application. Elle était hébergée gratuitement sur Netlify. C'était donc un site web statique.

lien de la version de l'année dernière: <https://v0beeo.netlify.app/>

lien de la version actuelle: <https://beeo.netlify.app/>

Les objectifs principaux du prolongement de ce projet étaient, pour la partie technique, de mettre en place un serveur avec une base de données adaptée aux données géographiques, reliée à l'application via une API. Aussi, d'utiliser une solution ETL pour automatiser l'importation des données afin de garder l'application à jour automatiquement. Enfin, ajouter de nouveaux calques.

Pour la partie non technique, nous avions pour objectif de faire les démarches pour breveter, et également étudier la monétisation de l'application.

IV - D'Angular à React

Angular et React sont tous les deux des framework de développement front-end populaires, mais ils ont des approches différentes pour créer des applications.

L'an dernier, aucun membre du groupe n'était à l'aise avec un quelconque framework JS et nous avons donc choisi Angular, qui nous était proposé dans le sujet. Cette année, un membre du groupe a eu plusieurs expériences sur React, donc nous avons donc décidé de passer à ce framework pour gagner du temps sur le long terme.

Présentation de React

ReactJS est une bibliothèque JavaScript open-source développée par Facebook. Elle est utilisée pour créer des interfaces utilisateur côté client, notamment pour les applications web. React permet de créer des composants à partir de données qui peuvent être facilement réutilisés, ce qui rend le développement d'applications plus rapide et plus facile. De plus, React utilise un DOM virtuel qui rend les performances plus rapides que celles des autres bibliothèques JavaScript. De plus, il existe en React un package [react-leaflet](#), qui réduit grandement le boilerplate code de leaflet.

Le bundler utilisé n'est pas webpack, mais Vite JS, qui est bien plus rapide.

Architecture de l'application React

L'architecture d'une application React est basée sur l'utilisation de composants React. Un composant React est une unité de code qui peut être utilisée pour afficher une partie d'une interface utilisateur. Chaque composant peut avoir des propriétés et un état interne, ce qui lui permet de gérer les données et les interactions avec l'utilisateur de manière indépendante.

Voici ci-dessous quelques composants que nous avons créés pour notre application. Ces composants affichent chacun une partie de l'application, mais il existe encore d'autres composants qui exécutent simplement des fonctions de l'application, comme *Flyer.jsx*, qui permet de voyager automatiquement au marqueur sélectionné dans la barre de recherche.

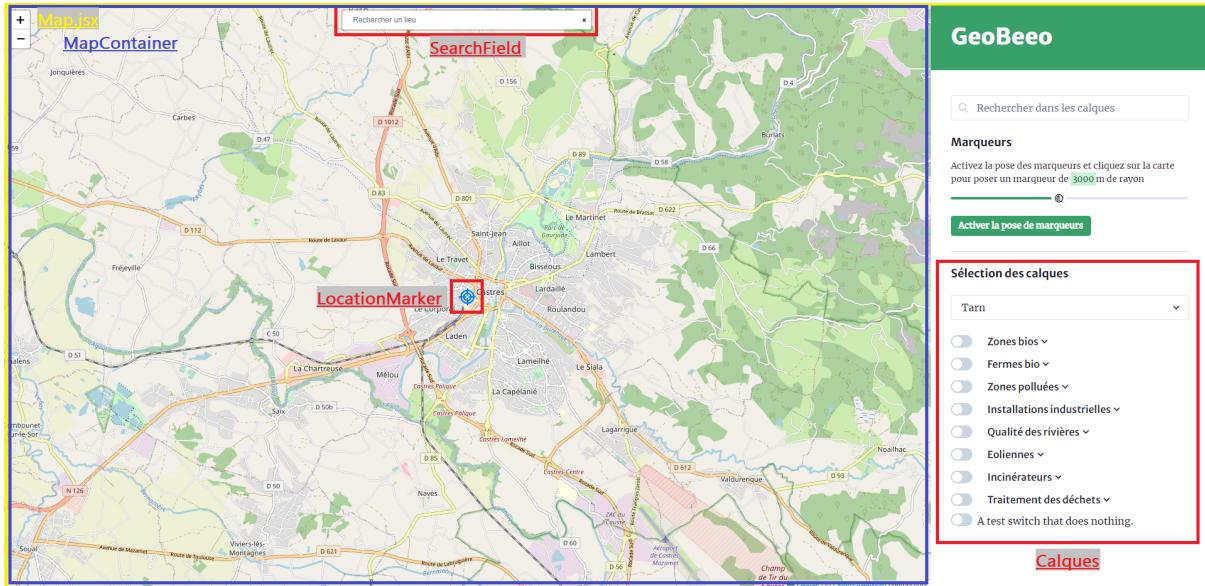


Fig 2. UI de l'application montrant les calques

React inclut également un moteur de rendu qui permet de gérer la mise à jour de l'interface utilisateur en fonction des changements dans les données et l'état interne des composants. Cela nous permet de créer des applications web interactives et réactives sans avoir à gérer manuellement les mises à jour de l'interface utilisateur.

V - Amélioration de l'UX et de l'UI

l'UX (user experience) et l'UI (user interface) sont des aspects clés du design d'interfaces utilisateur. L'UX se concentre sur l'expérience globale de l'utilisateur, tandis que l'UI se concentre sur l'apparence et la disposition des éléments visuels de l'interface.

Le passage à React nous a permis d'utiliser Chakra UI, une librairie de composants populaire, qui permet de réduire le temps passé sur le HTML/CSS. Il fournit un large éventail de composants prêts à l'emploi, tels que des boutons, des formulaires, des modaux et des menus, qui peuvent être facilement intégrés dans une application React. L'utilisation de librairies de composants permet de donner à l'application une uniformité dans l'apparence.

Au niveau du placement des marqueurs utilisateur, le slider pour la sélection du rayon de butinage a été modifié pour n'avoir que des valeurs d'intervalles 100, permettant plus facilement de choisir 3000 mètres par exemple. Les boutons en dessous ont été remplacés par un seul bouton plus clair ("Activer les marqueurs" → "Activer la pose de

marqueurs”), et la suppression des marqueurs se fait directement sur la carte, en les sélectionnant.

Marqueurs

Activez la pose des marqueurs et cliquez sur la carte pour poser un marqueur de 3000 m de rayon



Fig 3. Marqueurs

Les calques devenaient assez nombreux et sur les plus petits écrans, il fallait scroller pour les voir tous. Comme nous avions prévu de rajouter encore des calques, nous avons dû simplifier cette partie. Chaque calque est maintenant cliquable pour faire apparaître sa description, et sa source. Nous sommes aussi passés d'une checkbox à un switch, qui est plus clair et plus joli.

Zones polluées ▾

Installations industrielles ^

Correspond aux Installations classées pour la protection de l'environnement (ICPE) soumises à autorisation ou à enregistrement (en construction, en fonctionnement ou en cessation d'activité)
[source ↗](#)

Fig 4. Calques détaillés



Responsive

Sur mobile, l'application change d'affichage, la carte passe en haut et les menus en bas. Un affichage mobile compatible est important pour notre application qui est destinée à des gens qui peuvent souvent être à l'extérieur.

Fig 5. UI Responsive

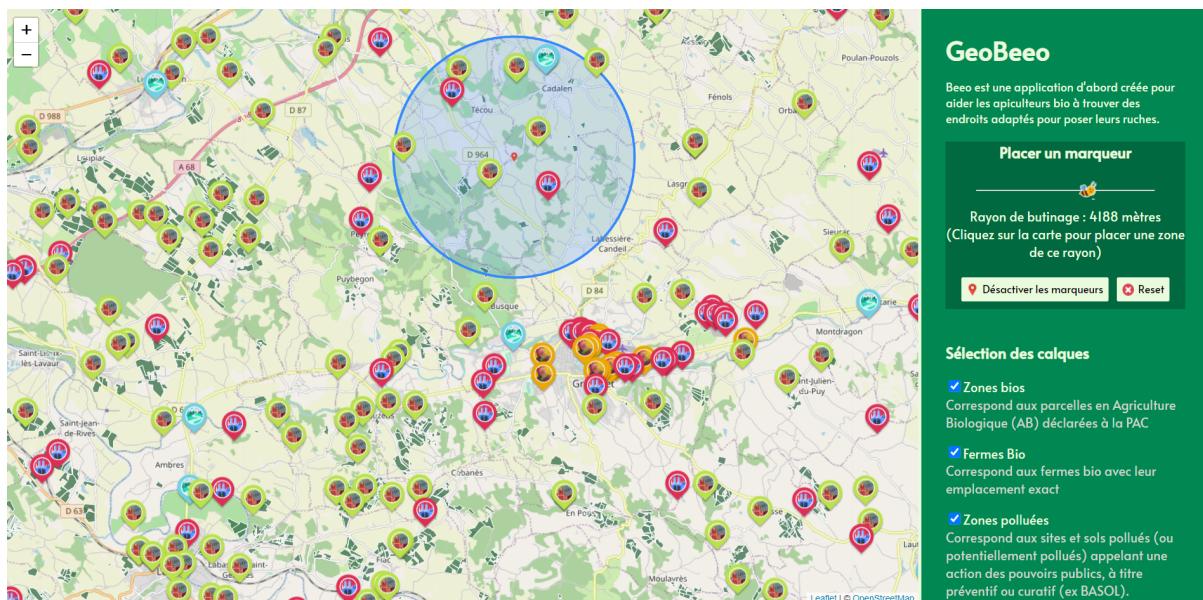


Fig 6. Apparence générale de l'ancienne version

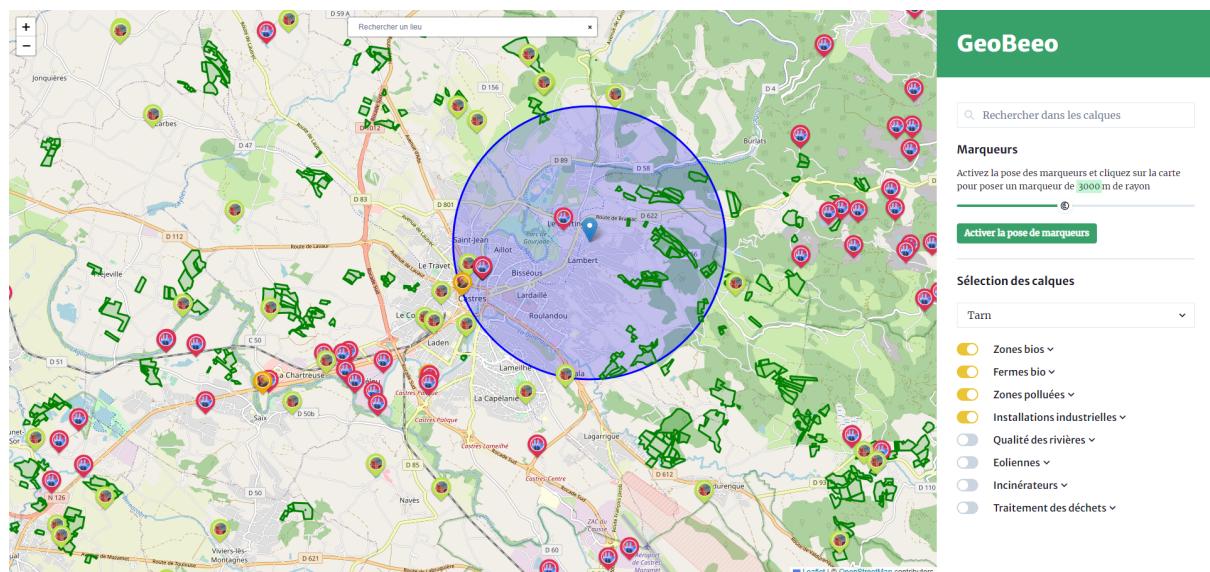


Fig 7. Apparence générale de la version actuelle

VI - Mise en place d'une base de données

L'an dernier, les données des calques étaient des fichiers statiques JSON directement dans l'application. Cependant, il y a plusieurs inconvénients à utiliser des fichiers statiques plutôt qu'une base de données pour stocker des données.

Au niveau de la scalabilité, les fichiers statiques peuvent devenir volumineux et difficiles à gérer à mesure que la quantité de données augmente, ce qui peut entraîner des problèmes de performance. Au niveau de la récupération des données,

les fichiers statiques doivent être lus manuellement pour accéder aux données, ce qui peut être fastidieux et inefficace. Nous verrons plus tard que certaines fonctionnalités n'ont pu être implémentées que grâce au passage à une base de données.

Il y a aussi des avantages de sécurité évidents aux bases de données mais pour notre projet, nous n'utilisons que des données déjà libres, trouvées sur internet.

Pour commencer, nous avons créé une base de données PostgreSQL locale, et installé PostGIS. PostGIS est une extension pour PostgreSQL qui permet d'ajouter des fonctionnalités de géospatiale à PostgreSQL, en offrant des outils pour stocker, manipuler et analyser des données géographiques.

L'extension permet d'upload des fichiers Shapefile dans la base de données, donc il a fallu convertir tous nos fichiers GeoJSON en Shapefile, à l'aide de [mapshaper](#). Mapshaper permet aussi de passer toutes nos données géographiques au format WGS84, qui n'est pas forcément celui par défaut sur les sites où nous trouvions nos données, ce qui posait problème à l'affichage.

Ensuite, nous avons dû passer à une base de données hébergée, et avons choisi dans un premier temps Google Cloud SQL, qui nous paraissait être la meilleure option. Google Cloud SQL est un service de base de données relationnelles qui permet entre autres de faire du postgres, et d'installer PostGIS. Google fournit aussi une période d'essai gratuite de quelques mois et il est très rapide et facile de tout mettre en place.

Nous avons ensuite fait le choix de passer à Supabase, pour pouvoir garder la gratuité de l'hébergement de manière permanente. En effet, sur la durée, Google allait devenir payant, tandis que Supabase possède un forfait basé sur l'utilisation effective de la base qui permet, pour les petites applications, de ne rien payer.



Notre base de données est donc un ensemble de 8 tables qui correspondent aux 8 calques que l'utilisateur peut activer sur l'application.

VII – Mise en place de l'API

Pour accéder aux données de la base, il a fallu créer une API. Nous avons choisi Node JS, que nous connaissons bien, et qui permet de développer avec javascript côté serveur. Par dessus Node JS, on utilise Express, un framework qui ajoute des fonctionnalités et facilite le développement.

Cette API REST est composée de trois chemins GET, qui exécutent chacun une requête SQL sur la base de données Postgres et retourne les résultats au format GeoJson.

le chemin GET /layers

GET /layers permet de récupérer toutes les données d'une table passée en paramètres. C'est le chemin utilisé lorsque l'utilisateur utilise un switch sur l'application pour sélectionner un calque.

La requête SQL est construite de manière à retourner un format GeoJson, grâce aux fonctions *jsonb_build_object*, *jsonb_agg* et *to_jsonb* de postgres. Deux conditions facultatives sont intégrées dynamiquement à la requête, dans des clauses WHERE :

- La première si l'utilisateur choisi un département (détailée [ici](#))
- La seconde si le calque sélectionné est "zones bios" (détailée dans la [partie suivante](#))

Chaque ligne retournée est donc ensuite transformée en Json, et la colonne correspondant aux coordonnées est transformée pour passer du format Shapefile à GeoJson, pour pouvoir être utilisée par Leaflet dans l'application. En effet, les données géographiques sont maintenant stockées dans la base au format Shapefile, mais Leaflet a toujours besoin des données au format GeoJson pour les lire et les afficher. Nous utilisons pour cela la fonction *ST_AsGeoJSON* de Postgis, qui permet de traduire directement le shapefile en GeoJson dans la requête SQL.

les chemins GET /search & GET /one

GET /search permet de récupérer toutes les données qui correspondent à la saisie de l'utilisateur dans la barre de recherche du site.

La requête est moins compliquée que la précédente. L'idée est de chercher dans toutes les tables compatibles avec la recherche le mot clé tapé par l'utilisateur. Les tables sont définies comme compatibles ou non avec la recherche dans l'API. Nous revenons plus tard sur ce fonctionnement.

GET /one permet de récupérer uniquement la donnée qui nous intéresse lorsque l'utilisateur sélectionne un élément bien précis dans la barre de recherche.

Le fonctionnement de ces deux derniers chemins est développé dans [cette partie](#).

Hébergement de l'API

L'API est hébergée sur Heroku, une plateforme très populaire pour héberger des applications Node JS sans avoir à gérer un serveur. Depuis le 28 novembre 2022, Heroku a perdu son niveau gratuit et n'a plus que des options payantes. Il faudra donc trouver une nouvelle option d'hébergement de l'API.

Requête depuis React

Les requêtes à l'API depuis React sont effectuées avec Axios, une bibliothèque JavaScript qui permet de faire les requêtes HTTP. Axios est très similaire à Fetch, mais nous avons choisi Axios par habitude. Dans une fonction asynchrone, on appelle l'API sur un de ses chemins /GET, et on attend sa réponse. Lorsqu'une réponse est obtenue, on stocke les données dans une variable et on les affiche avec Leaflet. Lorsque la base de données devient grande, et avec le traitement qui doit être fait en plus dans la requête SQL, le temps d'attente peut devenir assez long. C'est ce qui nous est arrivé avec le calque des zones bios, et c'est pourquoi nous avons dû trouver des solutions pour améliorer les performances.

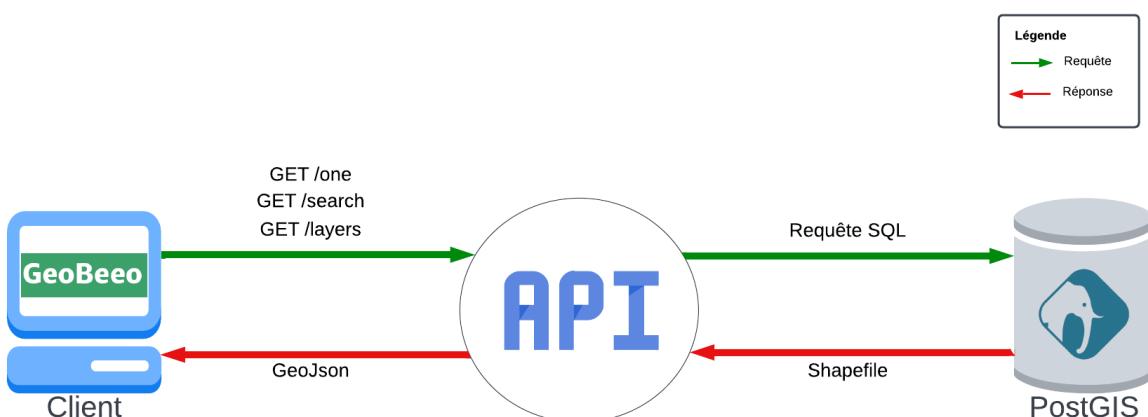


Fig 8. Schéma de l'architecture globale de l'application

VIII - Amélioration des performances

Le calque principal, celui des zones bios, en contient 210 000 différentes, ce qui pose de gros soucis de performances pour les envoyer comme pour les afficher, même en ayant simplifié le nombre de points. Il faut donc un moyen de limiter dès la requête le nombre de zones retournées. Nous avons pensé à deux solutions :

- A partir de la position de l'utilisateur, dessiner un cercle de rayon X kilomètres, et ne garder que les zones se trouvant à l'intérieur.
- Tracer un rectangle qui correspond aux limites de la carte affichée sur l'écran de l'utilisateur, afin de ne garder que ce qu'il voit.

La première solution pose problème si la personne n'a pas activé sa localisation, ou si elle veut voir les zones ailleurs qu'autour d'elle, ce qui nous a fait opter pour la seconde solution. Pour récupérer les coordonnées de la carte affichée, on utilise la fonction Leaflet `map.getBounds`, qui renvoie les points haut-gauche et bas-droit, qui permettent de tracer le rectangle de délimitation. A chaque fois que l'utilisateur zoom, dezoom, ou déplace la carte, on retrace un nouveau rectangle. Ce rectangle est ensuite envoyé en paramètre facultatif à l'API.

Du côté de l'API, on utilise la fonction `ST_Contains` de PostGIS qui retourne `true` si un polygone en contient un autre. En utilisant cette fonction dans le `WHERE` de la requête SQL, on peut ne sélectionner que les zones bios se trouvant dans le rectangle de délimitation envoyé en paramètres.

Cette méthode nous a permis de passer du Tarn à toute l'Occitanie pour les zones bios. Néanmoins, si l'utilisateur essaye d'afficher une grande zone, les problèmes de performances seront toujours là.

IX - Nouvelles fonctionnalités

Nouveaux calques

Avec la simplification du code et la nouvelle base de données, l'ajout de nouveaux calques est beaucoup plus rapide qu'avant. La principale difficulté est de trouver des données intéressantes et exploitables pour notre application.

Au niveau des zones bios, nous sommes passés de 12 000 à 210 000 zones.

Nous avons rajouté les éoliennes, qui dans le contexte de notre application, représentent une nuisance sonore, et peuvent déboussoler les abeilles avec les infrasons qu'elles produisent. La base de données utilisée semble incomplète, bien qu'elle provienne de Géorisques, le site du gouvernement.

Nous avons ajouté un calque avec les installations de traitement des déchets résiduels. Il recense :

- les installations de stockage de déchets non dangereux
- les unités d'incinération des ordures ménagères
- les unités de valorisation énergétiques
- les installations effectuant un traitement mécano-biologique.

Barre de recherche localisations

La barre de recherche provient du composant leaflet-geosearch, et utilise l'API d'OpenStreetMap. Elle permet de rechercher une ville, une région, un département...

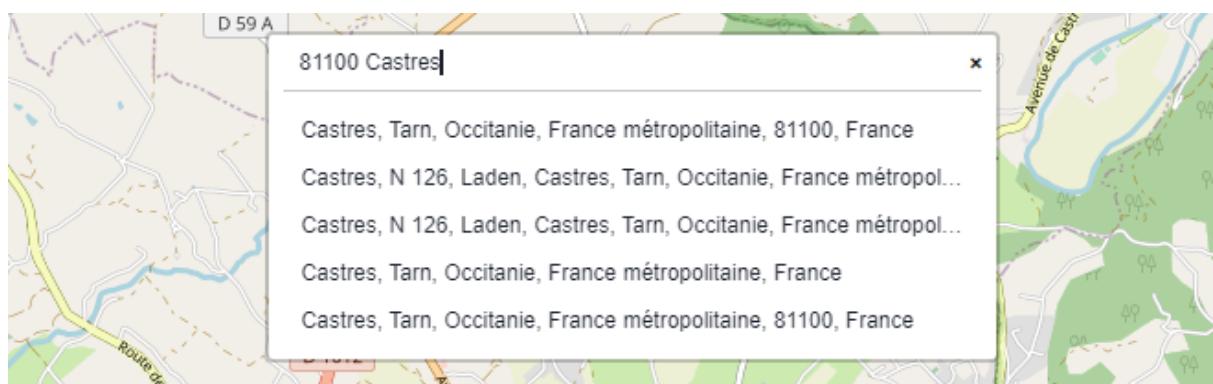


Fig 9. La barre de recherche localisations

Barre de recherche des calques

Le passage à une base de données postgres nous a aussi permis de mettre en place une recherche dans les calques. L'utilisateur peut aller chercher une installation industrielle qu'il connaît et y placer un marqueur de 3 km de rayon par exemple. Pour les fermes bios, il peut rechercher par type de culture...

La recherche se fait à chaque lettre saisie, à partir de 4 lettres (pour éviter de récupérer toute la base en tapant "e" par exemple).

Une table de correspondance dans l'API permet de définir les calques concernés par la recherche, et dans quel colonne des données doit se faire la recherche pour chaque calque.

Quand on récupère les résultats sur l'application, on les affiche avec un badge sur la gauche qui informe du type de calque. Lors du clic sur un résultat, l'API est appelée sur le chemin GET /one, en passant en paramètres le type de calque et son ID. Une fois le résultat récupéré, on affiche sur la carte le marqueur correspondant, avec toutes les informations habituelles.

Tri par département

Certaines tables de la base de données ont une colonne pour le département. Nous avons donc mis en place un champ select du département, qui permet pour l'utilisateur de mieux cibler la zone voulue, et pour l'application de charger moins de marqueurs d'un coup.

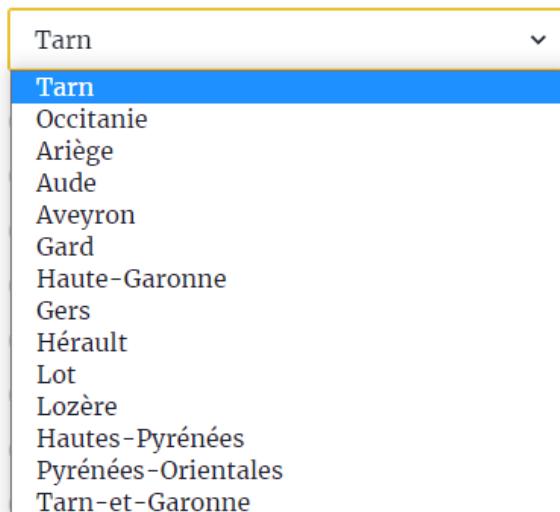


Fig 11. Tri par département

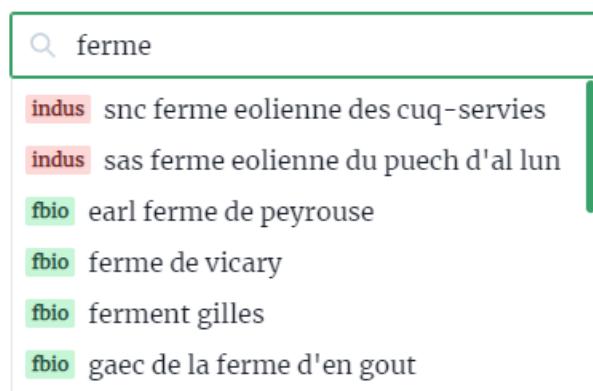


Fig 10. La barre de recherche des calques

Lorsqu'un département est sélectionné, les requêtes à l'API sont faites avec un nouveau paramètre : le numéro du département. Dans une table de correspondance dans l'API, on définit le nom de la colonne département pour chaque calque, s'il y en a une, afin de construire dynamiquement la requête SQL. S'il n'y a pas de colonne correspondant au département, alors on renvoie toutes les données.

X - Récapitulatif des calques

Finalement, voici les 8 calques qui composent notre application à ce jour, avec leur description et leur source :

Zones bios

Correspond aux parcelles en Agriculture Biologique (AB) déclarées à la PAC.

Les données proviennent de [data.gouv](#).

Fermes Bios

Correspond aux fermes bio avec leur emplacement exact.

Les données proviennent de [agence bio](#)

Qualité des rivières

Correspond aux qualité des rivières avec l'emplacement de la station de mesure.

Les données proviennent de [eaufrance.fr](#)

Zones polluées

Correspond aux sites et sols pollués (ou potentiellement pollués) appelant une action des pouvoirs publics, à titre préventif ou curatif (ex BASOL).

Les données proviennent de [georisques.gouv](#)

Installations industrielles

Correspond aux Installations classées pour la protection de l'environnement (ICPE) soumises à autorisation ou à enregistrement (en construction, en fonctionnement ou en cessation d'activité)

Les données proviennent de [georisques.gouv](#)

Les éoliennes

Correspond aux éoliennes composant les parcs éoliens terrestres

Les données proviennent de [georisques.gouv](#)

Les incinérateurs

Liste des incinérateurs d'ordures ménagères de la région Occitanie.

Les données ont été trouvées via la DREAL de Occitanie, elles sont disponibles [ici](#)

Les traitements de déchets

Localisation des installations de traitement de déchets résiduels en Occitanie en 2020. Il s'agit des Installations de Stockage de Déchets Non Dangereux (ISDND), des Unités d'Incinération des Ordures Ménagères (UIOM), les Unités de Valorisation Energétique (UVE),et les installations effectuant un Traitement Mécano-Biologique (TMB).

Les données ont été trouvées via la DREAL de Occitanie, elles sont disponibles [ici](#)

XI - Automatiser la mise à jour de la BDD

Pour automatiser la mise à jour de la base de données, nous avons décidé d'essayer 2 approches. La première approche est Talend ETL et la seconde est MS Power automate. Pour ce faire, nous avons déployé une vm google cloud avec une disquette de démarrage windows. Cela nous permet d'avoir nos applications en marche tout le temps afin de faire des mises à jour programmées.

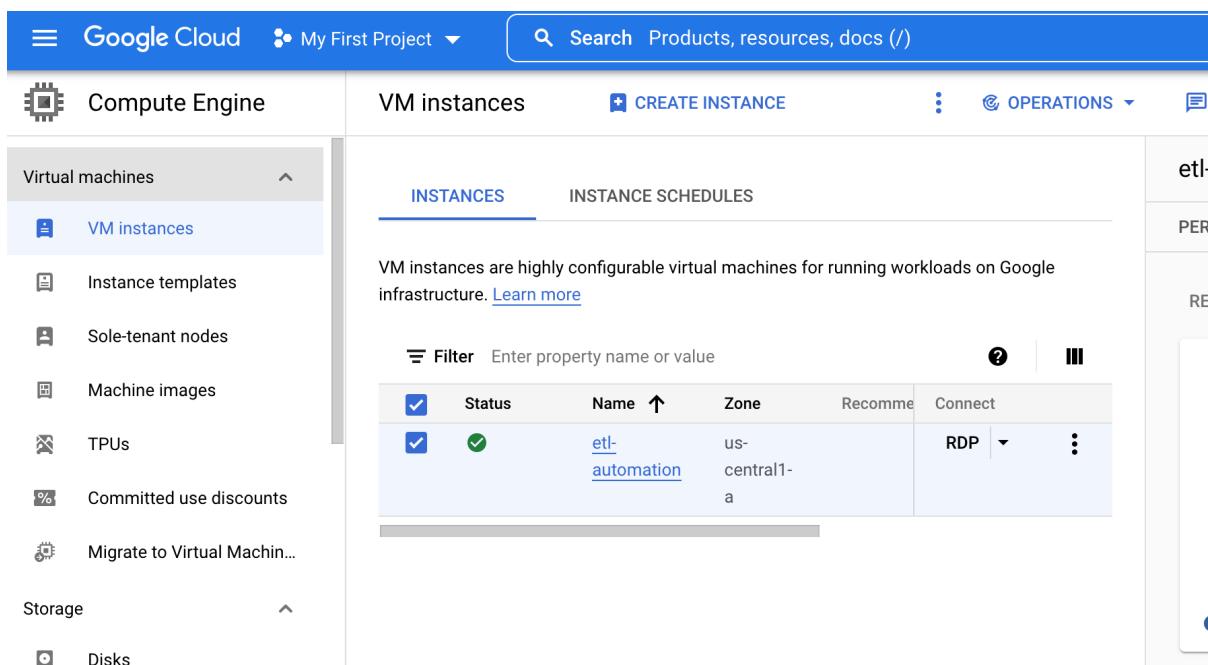


Fig 12. Google Cloud VM

Talend ETL

L'ETL Talend est une solution logicielle qui permet de réaliser des opérations d'extraction, de transformation et de chargement de données (ETL en anglais). En d'autres termes, il s'agit d'un outil qui permet de collecter des données à partir de différentes sources, de les nettoyer et de les organiser de manière à les rendre utilisables pour des analyses ou des prises de décision. Le logiciel Talend est conçu pour faciliter et automatiser ces processus, ce qui permet aux utilisateurs de gagner du temps et de la précision dans le traitement de leurs données.

En effectuant les étapes suivantes, nous avons réussi à télécharger le shapefile de georisque pour (installations industrielles) et à le transformer en csv pour l'intégrer à notre base de données postgres.

Nous avons commencé par créer un nouveau travail qui va éventuellement télécharger le shapefile depuis le georisque, le dézipper, le lire, le convertir en CSV et l'importer dans la base de données postgres.

Nous avons fait glisser le composant tFileFetch sur l'espace de travail de conception. Ce composant sera utilisé pour télécharger le fichier de forme à partir de l'URL.

Nous avons configuré le composant tFileFetch en spécifiant l'URL du fichier de forme de georisque et le répertoire où vous voulez sauvegarder le fichier téléchargé.

Nous avons ensuite utilisé un composant tFileUnarchive. Ce composant sera utilisé pour décompresser le fichier de forme téléchargé.

Ensuite, nous avons connecté le composant tFileFetch au composant tFileUnarchive en utilisant une connexion Trigger > OnSubjectOk. Cela permettra de s'assurer que le fichier de forme est décompressé après avoir été téléchargé.

Ensuite, nous avons configuré le composant tFileUnarchive en spécifiant le répertoire où le fichier de forme a été téléchargé et le répertoire où vous souhaitez enregistrer les fichiers décompressés.

Nous avons ensuite téléchargé un plugin préétabli afin d'utiliser le composant dShapefileInput. Ce composant sera utilisé pour lire le fichier de forme que nous avons téléchargé.

Nous avons ensuite connecté le composant tFileUnarchive au composant dShapefileInput à l'aide d'un Trigger > OnSubjectOk. Cela permettra de s'assurer que le fichier de forme est lu après avoir été décompressé.

Nous avons ensuite configuré le composant sShapefileInput en spécifiant le répertoire où se trouve le fichier de forme décompressé.

Nous avons ensuite utilisé le composant tFileOutputDelimited. Ce composant sera utilisé pour convertir le fichier de forme en CSV.

Nous avons connecté le composant sShapefileInput au composant tFileOutputDelimited en utilisant une connexion Row > Main. Cela garantira que le fichier de forme est converti en CSV après avoir été lu.

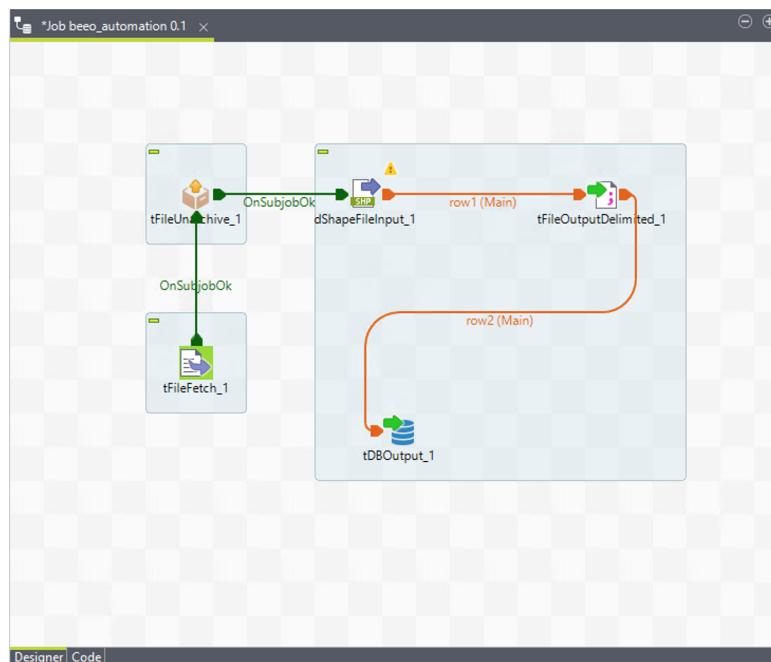
Nous avons configuré le composant **tFileOutputDelimited** en spécifiant le répertoire où vous souhaitez enregistrer le fichier CSV et le format du fichier CSV.

Ensuite, nous avons utilisé le composant **tPostgresqlOutput**. Ce composant sera utilisé pour écrire les données du fichier CSV dans la base de données Postgres.

Nous avons connecté le composant **tFileOutputDelimited** au composant **tPostgresqlOutput** en utilisant une connexion Row > Main. Cela garantira que les données du fichier CSV seront écrites dans la base de données Postgres après avoir été lues.

Nous configurons le composant **tPostgresqlOutput** en spécifiant les détails de connexion pour votre base de données Postgres (par exemple, hôte, port, nom d'utilisateur, mot de passe, etc.) et le nom de la table et des colonnes où nous voulons importer les données.

Enfin, nous avons programmé la tâche pour qu'elle commence aujourd'hui et se répète tous les 30 jours. Cela permettra de répéter le job tout seul une fois par mois et d'automatiser le téléchargement et l'intégration des données dans la base de données.



MS Power Automate

Fig 13. Connecteur Talend ETL

Microsoft Power Automate est un logiciel qui permet aux utilisateurs d'automatiser des tâches répétitives, telles que la création de flux de travail et d'intégrations de

données. Il peut être utilisé pour automatiser un large éventail de processus, du simple transfert de données aux processus commerciaux complexes.

En effectuant les étapes suivantes, nous avons réussi à télécharger le fichier de forme de georisque pour (Installations industrielles) et à le transformer en une table pour l'intégrer à notre base de données postgres.

Tout d'abord, nous avons créé un workflow planifié qui s'exécute tous les 30 jours. Ensuite, nous avons ajouté les actions suivantes :

1. **Launch New Chrome** : Cela ouvrira une page de chrome vers notre URL désignée "<https://www.georisques.gouv.fr/donnees/bases-de-donnees/installations-industrielles>" où nous pouvons trouver le fichier zippé à télécharger.
2. **Cliquez sur l'élément UI** : Cet élément ajoutera une action de clic au bouton "Départemental" afin de rechercher le fichier requis.
3. **Cliquez sur l'élément UI** : Cela permettra de cliquer sur la barre de recherche
4. **Envoyer les clés** : Ceci enverra une chaîne de "81 - TARN" afin que nous puissions filtrer le fichier requis.
5. **Envoyer les clés** : Ceci enverra la clé "Enter" afin de rechercher la chaîne précédente.
6. **Cliquez sur le lien sur la page** : Ceci va cliquer sur le lien de téléchargement
7. **Fermer la fenêtre** : Ferme la fenêtre du navigateur
8. **Cliquer sur l'élément d'interface utilisateur** : Ferme la fenêtre du navigateur
9. **Dézipper les fichiers** : Ceci va dézipper le fichier téléchargé
10. **Créer un nouveau tableau** : Ceci prendra les informations du fichier Shapefile et les insérera dans une table.
11. **Ouvrir une connexion SQL** : Cette opération crée une connexion à notre base de données Postgres.
12. **Exécuter une déclaration SQL** : Ceci va intégrer les données de notre table à notre base de données.

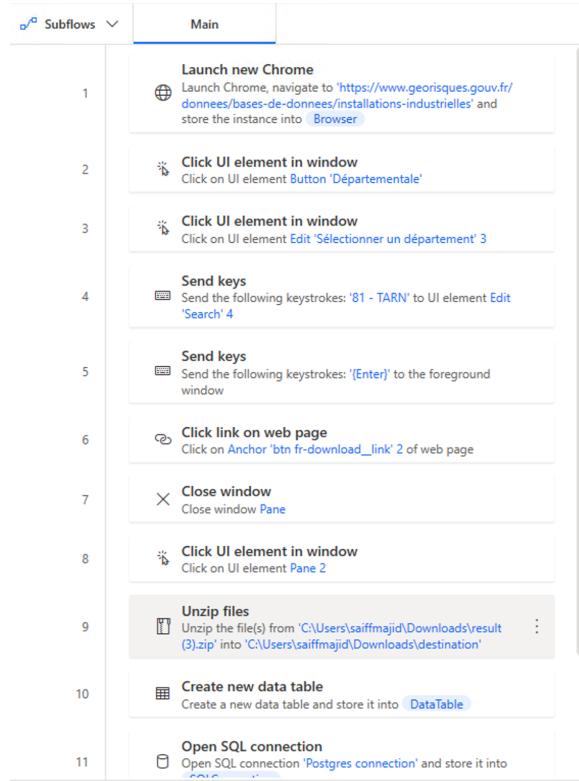


Fig 14. Connecteur MS Automate

Comparaison entre Talend et MS Power Automate

Critères	Talend ETL	MS Power Automate
Prix de la licence	Gratuit	Gratuit
Prix du déploiement GCP (MS 2vCPU - 8GB)	54\$ / Mois	54\$ / Mois
Utilisation du CPU	Moyen	élevé
Pros	<ul style="list-style-type: none"> • Utilisation économique du CPU • Exécution du travail en background 	<ul style="list-style-type: none"> • Facile à utiliser • Ciblez un élément de l'interface utilisateur au lieu d'une URL
Cons	<ul style="list-style-type: none"> • Difficile à utiliser • Difficile à maintenir • Cible une URL 	<ul style="list-style-type: none"> • Utilisation élevée du CPU • Exécution de travaux dans l'environnement de l'utilisateur

Après avoir recherché tous les aspects de chaque approche, nous avons découvert que :

Talend:

Talend ETL ne sera pas très utile pour nos besoins actuels car les données dont nous avons besoin ne sont pas stockées dans une URL unique (l'URL change à chaque fois que Georisque télécharge un nouveau fichier). Cela causera des problèmes dans nos connecteurs. De plus, tout au long de notre projet, Georisque a modifié son UI à deux reprises (par exemple, un nouveau bouton départemental a été ajouté à l'UI), nous ne pouvons donc pas compter sur un tMap pour cibler un certain élément afin d'obtenir le lien de téléchargement.

MS Power Automate:

MS Power Automate utilise une grande quantité de CPU lors de l'exécution d'un job, ce qui peut causer des problèmes aux workflows sachant que nous devons avoir environ 7 workflows pour chaque table de notre base de données. De plus, MS Power Automate fonctionne sur le bureau de l'utilisateur. En d'autres termes, il lance physiquement un navigateur Chrome afin de terminer le workflow. Il n'est vraiment pas pratique d'avoir un workflow entièrement dépendant d'un navigateur car nous pourrions avoir des problèmes à chaque fois que Google envoie une nouvelle mise à jour du navigateur.

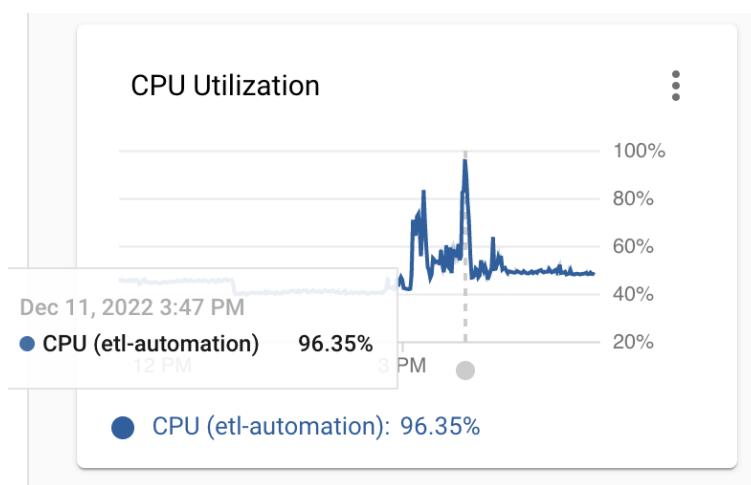


Fig 15. Utilisation du CPU après le lancement du workflow MS automate

La solution optimale :

Nous avons constaté que Georisque et data.gouv.fr vont déployer des services API pour les utilisateurs qui voudraient utiliser leurs données. Ces services sont encore en phase bêta et les données que nous utilisons n'ont pas encore d'API. Nous recommandons donc d'attendre que ces services soient complètement déployés et il sera beaucoup plus pratique de connecter notre application web aux API de data.gouv.fr.

XII – Crédit du brevet

Vu l'avancement qui est plutôt bien de notre projet, nous avons eu l'idée de réaliser un brevet pour notre projet, qui nous permettra de protéger la titularisation de l'invention, mais aussi pour apporter des informations précieuses et constituer une source d'inspiration pour les générations futures de chercheurs et d'inventeurs.

Qu'est-ce qu'un brevet ?

Le brevet est un acte officiel de propriété industrielle qui accorde un monopole d'exploitation au demandeur sur son invention sur le territoire français pour 20 ans au maximum. Le fait de déposer un brevet interdit toute exploitation de cette dernière sans autorisation. Le brevet donne par ailleurs à son inventeur des moyens de conquérir de nouveaux marchés à l'étranger.

Être brevetable ?

Pour pouvoir être brevetable, notre projet doit remplir les critères suivants :

- être une solution technique à un problème technique
- être une invention nouvelle
- impliquer une activité inventive
- être susceptible d'application industrielle

Après plusieurs recherches et plusieurs discussions, nous avons trouvé que nous ne pouvons pas envisager de brevet car il n'y a pas d'algorithme original et innovant dans notre projet.

Comment protéger le projet alors ?

Nous avons reçu quelques documents, que nous avons complété pour avoir le droit d'auteurs et que nous allons déposer auprès de l'INU.

XIII - Types de monétisation :

Site payant

Fixer un prix pour accéder au site sera payant pour l'utilisateur. Cette technique s'applique à des éditeurs qui sont très confiants dans la qualité de leur app et qui ont la certitude que l'utilisateur verra l'utilité de payer l'app avant même de l'avoir testée. Il faut par contre être très efficace dans le processus « d'avant-vente » et donc avoir une fiche description de l'app qui soit parfaitement calibrée. Le grand avantage est qu'on sait à l'avance qu'un téléchargement aura généré du revenu. Donc il faut faire en sorte que le coût d'acquisition soit inférieur au prix de l'app.

Monétiser son site gratuit via la Publicité (Freemium) :

Les modèles publicitaires marketing mobile sont bien connus des éditeurs de sites web. Le principe est exactement le même sur mobile. Les éditeurs d'applications de news ou de réseaux sociaux utilisent beaucoup cette méthode de monétisation. Il y a 2 paramètres à prendre en compte. Les impressions publicitaires sont vendues en général au CPM (Coût pour mille). Il faut générer de très gros volumes d'impressions publicitaires pour espérer gagner de l'argent. Ensuite, le volume n'est pas le seul paramètre. Il faut que les annonceurs puissent cibler précisément les publicités pour optimiser efficacement. Par conséquent il faudra que votre application fasse passer des informations sur vos utilisateurs de manière à ce que les régies publicitaires puissent cibler plus facilement les futurs consommateurs potentiels (par âge, sexe, localisation, centre d'intérêts, etc).

La clé du succès via la monétisation par la publicité sera donc le volume d'utilisateurs actifs de notre site et les capacités de ciblage.

Monétiser son site e-commerce - technique transactionnelle

Il s'agit du schéma classique e-commerce mais sur des sites. L'éditeur de l'application propose en général des produits ou des abonnements. Ces transactions passent par un module de paiement indépendant de Google ou Apple. Il faut en général rentrer les codes de la carte bleue. C'est le cas des applications comme Amazon ou eBay. Ce type de monétisation implique une excellente ergonomie du module de paiement pour faciliter les transactions. Il faut en outre favoriser les actions de ré-achat pour maximiser la rentabilité de chaque utilisateur acquis.

Demander des aides financières à l'état

Nous pouvons tenter notre chance et demander des aides financières à l'état, mais pour faire cela, il faut d'abord créer une entreprise, et demander les aides à des établissements comme :

Le dispositif local d'accompagnement ou DLA :

Ce dispositif gratuit a été mis en place par l'État et l'Europe pour venir en aide aux associations et aux entreprises solidaires en développement ou en difficulté. Il est destiné à financer des stratégies de développement, de communication et d'implantation et de financement. Ainsi que pour pérenniser les emplois.

L'agrément ESUS :

L'agrément ESUS ou Entreprise Solidaire d'Utilité Sociale s'adresse aux associations d'utilité publique. Il donne accès à :

- Des financements spécifiques de Bpi France ;
- Des marchés publics réservés ;
- L'épargne solidaire ;
- Des dispositifs locaux d'accompagnement comme le DLA ;

Certains dispositifs de soutien mis en œuvre par des collectivités territoriales ou des organismes privés comme les banques commerciales, Le dépôt du dossier pour la demande d'agrément ESUS se fait auprès de la DIRECCTE (Direction régionale des Entreprises).

La garantie FOES de France Active

Le FOES ou Fonds Entreprise Solidaire est une garantie apportée à des crédits bancaires pour financer un investissement ou les besoins en fonds de roulement d'une association d'utilité publique.

Cette garantie cautionne des prêts d'une durée de 2 à 7 ans et pour un montant de 5 000 € minimum. Le coût de la garantie est de 2,5 % du montant garanti (50 % du prêt limité aux premiers 100 000 € du prêt).

Le contrat d'apport associatif ou CAA

Le CAA est un dispositif visant à créer ou pérenniser des emplois et à développer des activités à caractère économique. Ce type d'aide prend la forme d'un apport en numéraire avec un droit de reprise.

Le contrat d'apport associatif est destiné aux associations de services à la personne, associations intervenant dans un secteur d'utilité sociale, structures d'insertion par l'activité économique, et aux entreprises de travail adapté. Le montant de l'aide est de 5 000 € à 30 000 € à rembourser en une ou plusieurs fois pendant 2 à 5 ans sans intérêt.

Les aides européennes

145 programmes de financement et 28 appels à projets ont été spécialement mis en place par l'Europe pour venir en aide aux associations à but non lucratif.

Dans la pratique, il en existe deux types dans l'Union européenne :

- Une subvention à l'action pour toute proposition de projet concret dans le cadre d'un programme d'objet communautaire couvrant un certain domaine d'activité, à condition que la proposition ait été retenue
- Une subvention destinée directement à l'association si celle-ci poursuit un but d'intérêt général européen ou un objectif qui s'inscrit dans le cadre d'une politique de l'Union européenne. Dans ce dernier cas, elle est fondée sur la portée européenne des activités de l'association loi 1901

Les aides des fondations

De nombreuses fondations accordent des aides financières aux associations (État, collectivités territoriales, département...). C'est le cas des fondations Nature et Découverte, Agir sa Vie, Auchan pour la Jeunesse, Raoul Follereau

XVI - Les mentions obligatoires :

	Pour une personne physique (micro-entreprise ou entreprise individuelle)	Pour une personne morale (société)
Votre identité	nom et prénom	<ul style="list-style-type: none"> ▶ raison sociale ▶ forme juridique ▶ montant du capital social
Vos coordonnées	<ul style="list-style-type: none"> ▶ adresse du domicile ▶ adresse de courrier électronique ou numéro de téléphone pour contacter votre entreprise 	<ul style="list-style-type: none"> ▶ adresse du siège social ▶ adresse de courrier électronique ou numéro de téléphone pour contacter votre entreprise
Les mentions relatives à la propriété intellectuelle	<ul style="list-style-type: none"> ▶ Si vous utilisez des images, illustrations, photographies : vous devez faire figurer leur propriété intellectuelle ▶ Pour les textes qui ne sont pas les vôtres vous devez recueillir l'autorisation de l'auteur ou tout du moins citer la source du texte 	<ul style="list-style-type: none"> ▶ Si vous utilisez des images, illustrations, photographies : vous devez faire figurer leur propriété intellectuelle ▶ Pour les textes qui ne sont pas les vôtres vous devez recueillir l'autorisation de l'auteur ou tout du moins citer la source du texte
Les mentions relatives à l'hébergement du site	<p>Vous devez prévoir une page relative aux mentions légales qui doit inclure des informations relatives à l'hébergement du site (même si le site est hébergé à titre gratuit)</p> <p>Ces mentions portent sur :</p> <ul style="list-style-type: none"> ▶ le nom de l'hébergeur ▶ la raison sociale ▶ l'adresse ▶ le numéro de téléphone. 	<p>Vous devez prévoir une page relative aux mentions légales qui doit inclure des informations relatives à l'hébergement du site (même si le site est hébergé à titre gratuit)</p> <p>Ces mentions portent sur :</p> <ul style="list-style-type: none"> ▶ le nom de l'hébergeur ▶ la raison sociale ▶ l'adresse ▶ le numéro de téléphone.

Fig 16. Les mentions obligatoires

Pour éviter les risques de non-respect des mentions obligatoires, notre site a besoin d'un cookie traceur permettant d'analyser le comportement des internautes, comme par exemple leurs navigations, leurs habitudes de consommation, leurs déplacements, etc.

Actuellement, nous ne devons pas intégrer un cookie traceur dans notre site web. Par contre, si nous décidons de monétiser notre application web par publicités, nous sommes obligés d'intégrer un cookie traceur pour informer les utilisateurs de la manière dont leurs données sont utilisées et obtenir leur consentement.

La Commission nationale de l'informatique et des libertés (CNIL) liste les cookies concernés par cette obligation.

En ce qui concerne le RGPD (Règlement général sur la protection des données), nous ne sommes pas obligés de les mentionner sur notre site car nous ne collectons pas les données personnelles des utilisateurs.

XVII – Perspectives et conclusion

Certaines de nos données proviennent de Géorisques. Au début du projet il n'y avait pas d'autres solutions que de télécharger leurs données pour les utiliser ensuite. Désormais, ils ont ajouté une API qui pourrait être utilisée pour mettre à jour les données de la base automatiquement.

Maintenant qu'une base de données est créée et fonctionnelle, nous pouvons envisager d'intégrer sur le site un formulaire de demande d'ajout de calques, avec les données de préférence, et ces demandes seraient stockées sur la base. La page de demande pourrait afficher assez simplement les demandes en cours, traitées, refusées...

Le projet “*Yuka des lieux bios*” est une application web fonctionnelle destinée à aider en premier lieu les apiculteurs, et qui peut s'étendre à d'autres publics. Nous pouvons mentionner les adeptes du tourisme vert, ou les consommateurs bio. Nous avons écrit le code de manière à rendre l'ajout de nouveaux calques le plus simple et rapide possible, ce qui permettra d'étendre la portée de l'application en évitant l'ajout de bugs. Pour cela, nous avons modifié l'architecture, créé une API et une base de données.

En parallèle, nous avons étudié les solutions d'automatisation des mises à jour, et dans une partie moins technique, la création d'un brevet et les types de monétisation possibles.

XVII – Ressources

1. <https://cloud.google.com/compute/docs>
2. <https://help.talend.com/r/fr-FR/8.0/release-notes-data-integration-products/documentation>
3. <https://learn.microsoft.com/en-us/power-platform/>
4. <https://leafletjs.com/reference.html>
5. <https://postgis.net/documentation/>
6. <https://www.economie.gouv.fr/entreprises/site-internet-mentions-obligatoire>
7. <https://www.cnil.fr/fr/rgpd-par-ou-commencer>
8. <https://supabase.com/docs>
9. <https://www.geoportail.gouv.fr/>
10. <https://annuaire.agencebio.org/>
11. <https://georisques.gouv.fr/>
12. <https://www.data.gouv.fr/>