

UNIVERSITÉ DE CAEN

MASTER 1 INFORMATIQUE

2021-2022

PROJET ANNUEL

Compte rendu projet annuel

ANIMAUX SAUVAGES

Paul GOSSE

Valentin LEROUGE



UNIVERSITÉ
CAEN
NORMANDIE

Table des matières

1	Introduction	2
2	Carte interactive	3
2.1	Mapbox GL-JS	3
2.2	Spécificités liés à la carte	4
3	Scrapping des données	5
3.1	Événements asynchrones	5
3.2	Exif-JS	6
4	Données de l'application	7
4.1	Table "Users"	7
4.2	Table "comments"	7
4.3	Table "Animaux"	7
5	Connexion/Inscription	9
6	Contenu	10
6.1	Ajout d'un animal	10
6.1.1	Frontend	10
6.1.2	Backend	11
6.2	Galerie des animaux	12
6.2.1	Détail d'un animal	12
6.2.2	Modification d'un animal	14
7	Conclusion	15
7.1	Objectif atteint ?	15
7.2	Pistes d'améliorations	15

1 Introduction

Ce projet consiste à enregistrer et visualiser la faune au travers d'une carte interactive. Les animaux sont représentés par des icones réparties aux 4 coins du monde et sont librement déplaçables en cas de modification. Chaque icône est muni d'une pop-up ouvrable au clic qui répertorie les différentes informations spécifique à l'animal sélectionné. Nous avons inclus 5 catégories : Mammifères, Insectes, Poissons, Oiseaux et Reptiles. Ces catégories sont suffisantes pour regrouper bon nombre d'animaux. En ce qui concerne les fonctionnalités principales de cette application, l'utilisateur à la possibilité d'ajouter des animaux sur la carte interactive principale pour les visualiser par la suite. L'utilisateur peut également modifier, s'il le désire, les animaux qu'il a crée par le biais d'un formulaire de modification. Le site est également muni d'un système de création/authentification de compte qui lui permet d'accéder à l'ensemble des fonctionnalités du site. Ce système est muni d'un envoi de mail automatisé qui se déclenche lors de la création d'un compte sur le site ou lorsque l'utilisateur connecté change son mot de passe. Parmi les ajouts secondaires, on notera la présence d'un espace commentaire qui permet à l'utilisateur d'interagir avec les membres du site. Nous avons également ajouté un système de récupération de données de géolocalisation au travers d'une image, si l'image en question en possède bien entendu. Des systèmes de filtres ont également été ajouté :

- Sur la carte, par le biais d'un sélecteur de catégorie
- Sur la page répertoriant les animaux, par le biais d'un sélecteur de categorie

Toutes ces fonctionnalités ont été pensées pour favoriser l'ergonomie du site ainsi que l'user-experience. Le but est de produire une application web accessible à tous, rapide au chargement et regroupant le plus de fonctionnalités possibles sans pour autant perdre l'utilisateur en cours de chemin.

Enfin, les langages utilisés pour produire cette application web sont les langages habituelles :

- PHP pour le backend
- HTML / CSS / JS pour le frontend
- MySQL pour le stockage de donnée.

Nous avons également utilisé Twig pour les templates du site web, PHP-Mailer pour l'envoi de mail automatisé et Exif-JS pour la récupération de métadonnées EXIF.

2 Carte interactive

2.1 Mapbox GL-JS

Nous avons utilisé Mapbox GL pour visualiser les animaux sur une carte interactive. Cette librairie Javascript que nous avons utilisé dans le passé lors d'autres projets est très complète et permet un niveau de customisation relativement poussé. La question était de savoir comment importer les données de la base de données correspondante dans la carte interactive. Nous avons décidé de stocker l'ensemble des données de la base de donnée Animaux (Celle contenant toutes les informations dont nous avons besoin) dans un GeoJSON. Un GeoJSON est un format de fichier spécialisé dans l'encodage de données géographiques. Ce GeoJSON est obtenue par une requête fetch réalisé en Javascript. Pour chaque "Feature", nous allons créer les différents éléments graphiques correspondant, tel que le marker placé sur la carte en fonction des coordonnées spécifiées dans le fichier lu, ou encore la pop-up contenant les informations de chaque animal. Dans notre modèle, une "Feature" correspond à un animal, avec ses différentes caractéristiques. Chaque marker est muni d'une pop-up qui s'ouvre au clic, contenant les informations de l'animal spécifié, tel qu'enregistré dans la base de données.

Nous avons ensuite ajouté un système de filtre sur catégorie intégré au sein de la carte. Il est possible de trier les différents animaux présents sur la carte interactive en fonction de la catégorie dont ils sont issus. Le résultat donne une carte visuellement attractive et très simple d'utilisation.

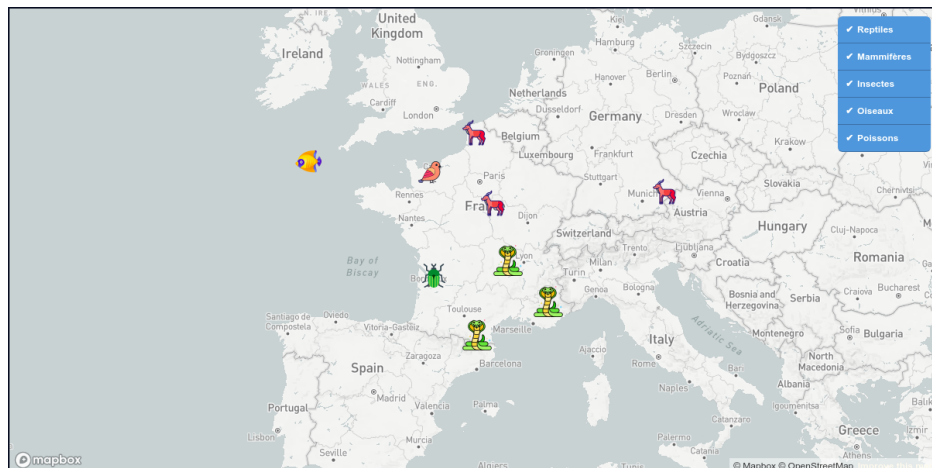


FIGURE 1 – Carte interactive

2.2 Spécificités liés à la carte

La carte interactive Mapbox GL est très polyvalente, cependant elle requiert des formats bien spécifiques notamment pour placer les différentes icônes. En effet, tandis que la majorité des informations de géolocalisations est encodée sous le format DMS (Degrees, Minutes, Seconds), Mapbox GL ne reconnaît que le format DD (Decimal Degrees). Il nous a donc fallu étudier la conversion du format DMS vers le format DD pour pouvoir afficher les icônes au bon endroit. Ainsi, une fonction de convertisseur est utilisée, notamment pour la récupération des informations de géolocalisation via la librairie Exif-JS (que nous verrons dans la suite du rapport). Cette question de conversion ne se pose pas lorsqu'on ping directement sur la carte interactive avec la souris pour déposer un marker (Lors de la création d'un animal ou de sa modification) puisque Mapbox permet de récupérer directement un format exploitable. Concernant les icônes présentes sur la carte, celles-ci sont en réalité des markers auxquels nous avons rattaché une div en guise de layout. Le changement d'image est effectué en fonction de la catégorie de l'animal rattaché au marker. Ainsi vous ne pourrez pas avoir un animal de catégorie "Poisson" avec une icône de reptile sur la carte. De plus, nous avons choisi un layout de carte où les territoires sont colorés en blanc et gris pour faire ressortir les icônes, c'est un choix purement artistique et nous espérons que cela est à votre convenance. Par ailleurs, un second layout de carte est utilisé dans les pages d'ajout et de modifications d'animaux pour varier le design. Éléments centraux de l'application web, ils nous ont fallu choisir une carte qui possède une vaste palette de customisation et Mapbox semblait le meilleur choix parmi les différentes solutions envisagées ¹



FIGURE 2 – Logo de mapbox

1. (Leaflet et Google Maps en tête de liste)

3 Scrapping des données

Concernant le scrapping des données, nous avons opté pour une récupération des données du site Wikipédia étant très complet. De plus, leur API est très bien expliqué et nous aura permis de gagner du temps sur cette partie essentielle du projet.

3.1 Événements asynchrones

Nous avons utilisé des promesses ainsi que des fonctions et événements asynchrones pour mener à bien notre scrapping. Rappelons brièvement ces quelques notions. Un langage de programmation asynchrone est un langage où chaque partie reçoit et traite les informations lorsque c'est possible. Dans Javascript cette asynchronicité est maintenue par l'utilisation de promesses qui sont des objets représentant la complétion (Resolve) ou l'échec (Reject) d'opérations asynchrones. Le but recherché va être d'attendre qu'une opération soit terminée avant d'en enclencher une nouvelle. En temps normal, les opérations sont quasiment instantanées (A moins d'avoir fait un algorithme très coûteux). Cependant dans le cadre de l'asynchrone en Javascript, nous allons pouvoir émettre des requêtes AJAX qui vont nous renvoyer des réponses dont nous allons nous servir en backend par la suite. Etant donnée que notre scrapping de données est réalisé en Javascript, il fallait envoyer des requêtes de type GET pour récupérer les différents éléments depuis le lien Wikipédia avant de les envoyer en POST. La première difficulté dont nous avons fait face fut de saisir cette notion d'asynchronicité des requêtes. En effet, les calls API prennent du temps, de ce fait, nous ne pouvions juste pas soumettre directement les formulaires et avoir la réponse instantanément. Nous avons donc utilisé des fonctions asynchrones (Reconnaisable par leur définition "Async function...") qui sont chargés d'envoyer les requêtes à l'API Wikipédia pour récupérer les informations nécessaires. Ces fonctions asynchrones retournent en réalité des promesses qui, tant qu'elles ne sont pas complètes, ne renverront vers rien d'autre dans Javascript. C'est pourquoi il est important de déterminer les moments clés, où l'on considère que la promesse doit se terminer. Dans le cadre du scrapping, ces promesses doivent bien évidemment se terminer une fois que la réponse à la requête API envoyée est reçue. Avec l'utilisation du mot clé "Await", il est possible d'attendre qu'une fonction asynchrone soit considéré comme terminée avant de poursuivre l'exécution du code. De ce fait, nous avons réalisé chaque call API dont nous avons besoin, puis nous avons transféré en AJAX le résultat de ces différents calls pour pouvoir les traiter ensuite en backend avec PHP.

3.2 Exif-JS

Wikipédia et son API sont très performants sur bien des aspects. Cependant, nous avons eu des problèmes concernant leur politique de sécurité en matière de crawler et de bots. En effet, le téléchargement de ressources externes tels que des images est sujet à une identification par le biais d'User Agent. En temps normal, cela ne pose pas de problème puisque le navigateur envoie automatiquement un user-agent lors de requêtes mais l'utilisation du serveur de la fac avec ses différents pare-feu de sécurité et sa CORS Policy nous empêcha de télécharger facilement les images depuis Wikipédia. Un vrai problème donc pour la recherche de métadonnées sur un fichier étant donné que nous pouvions uniquement obtenir le lien de l'image sur Wikipedia Commons, nous ne pouvions pas utiliser l'outil d'extraction de métadonnées Exiftool présent sur le serveur de la fac. Nous avons donc cherché des solutions alternatives, notamment l'utilisation de CURL pour récupérer l'image en spécifiant un user agent personnalisé ou la fonction "exif_read_data" présente nativement sur PHP. Sans succès. Nous avons donc fini par trouver une librairie Javascript qui permet l'extraction de métadonnées EXIF sur des URLs ou des array buffers, parfait pour nous donc. Cette librairie Open Source, prénommée Exif-JS permet une extraction complète de métadonnées EXIF sur n'importe quel format spécifié. Ainsi, nous avons pu obtenir la géolocalisation des images importés, à condition qu'elles contenaient bien des informations de géolocalisations dans leur métadonnée.



FIGURE 3 – <https://github.com/exif-js/exif-js/blob/master/LICENSE.md>

4 Données de l'application

4.1 Table "Users"

La table Users répertorie les différents comptes présents et utilisables sur le site. Elle est composé :

- d'un identifiant auto-incrémenté pour séparer les différents comptes.
- d'un username unique pour différencier chaque compte (Deux comptes ne peuvent pas avoir le même username pour des raisons de sécurité évidentes).
- d'un mot de passé hachée. Le hachage est générée par la fonction présente nativement sur PHP.
- d'un email, utile pour le système d'envoi de mail automatisé et pour changer de mot de passe.
- d'un rôle indiquant le niveau d'accès au site.

La table Users comme vous pouvez le constater est relativement simple et cela reste suffisant pour l'aspect communautaire du site web.

4.2 Table "comments"

La table Comments répertorie les différents commentaires présents sur notre site. Elle est composé :

- d'un identifiant auto-incrémenté pour séparer les différents commentaires.
- d'un identifiant indiquant la page ou le commentaire est présent (Identique à l'identifiant de l'animal rattaché).
- d'un nom qui est égale au nom d'utilisateur du site.
- d'un contenu textuel qui représente le commentaire en lui-même.

La table comments est-elle aussi relativement facile à comprendre. On crée un espace commentaire pour chaque animal et les utilisateurs peuvent écrire ce qu'ils désirent.

4.3 Table "Animaux"

La table Animaux répertorie les différents animaux présents sur notre site. Elle est composé :

- d'un identifiant auto-incrémenté pour séparer et différencier les animaux
- d'un nom qui est sélectionné par l'utilisateur.
- d'un file_data qui correspond à un blob et qui n'est rempli que lorsque une image est uploadé par l'utilisateur.

- d'un link qui stocke le lien wikipedia choisi lors de la phase de création de l'animal. Ce lien est essentiel pour la réalisation de la phase de scrapping.
- d'une longitude et latitude pour placer le marqueur sur la carte.
- d'une catégorie pour grouper les espèces et effectuer la taxonomie.
- d'une description qui peut être soit la récupération d'un paragraphe Wikipédia qui a été scrappé, soit une description customisé par l'utilisateur.
- d'un wikilink qui correspond au lien qui mène vers l'image de wikipedia quand l'utilisateur a décidé de scraper l'image.
- d'un booléen nommé "scrapped" qui permet de savoir si la description a été récupéré depuis Wikipédia ou si elle est purement customisé afin de préremplir le formulaire de modification.

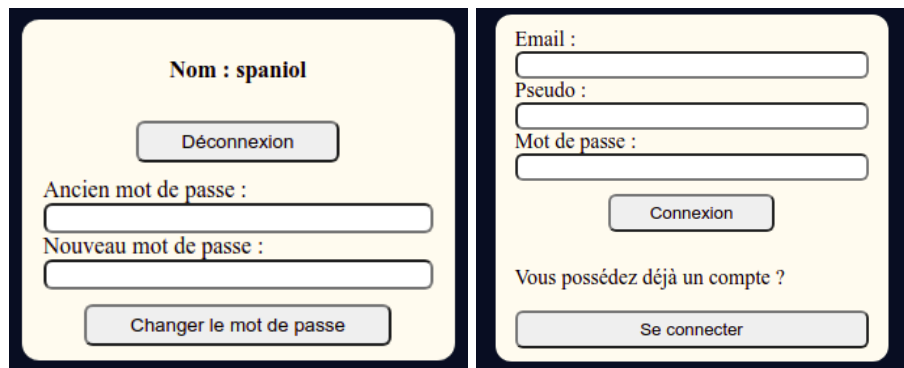
La table Animaux est la table principale du site web, il est donc normal qu'elle soit plus fourni que les autres. Il est important de noter la structure choisi : Stocker les images sous forme de blob et les liens wikipedia séparément. Cela permet à tous les utilisateurs d'avoir accès aux mêmes informations en permanence. Une autre solution envisageable aurait été de stocké en mémoire les différents chemins de fichiers correspondant aux images a afficher mais nous avons réfléchi aux choses suivantes :

- Les serveurs de la fac, soumis à une CORS Policy relativement restreinte, aurait-il été possible pour un autre utilisateur de visualiser les différentes images du site sans avoir a récupérer toutes les images correspondantes ?
- Comme dit précédemment, Wikipédia est très stricte sur sa politique d'User-Agent, aurions nous pu télécharger sans aucun problème tous les fichiers images de la base de données Wikipédia dont nous avons besoin

Il serait bien entendu intéressant d'étudier les deux modèles étudiées pour voir lequel conserver.

5 Connexion/Inscription

Comme dit précédemment, l'application web est muni d'un système d'authentification/inscription. Chaque utilisateur est stocké dans la base de données Users et peut accéder à l'intégralité du site en étant connecté. Un système de mail automatisé est rattaché au compte pour une expérience plus satisfaisante. Nous avons utilisé la librairie PHPMailer pour envoyer ces mails automatisés. Néanmoins, nous n'avons pas configuré de template Twig pour les mails, ce qui leur aurait donné un aspect beaucoup plus vivant. Cependant, le résultat reste convenable mais est bien entendu améliorable. De plus, l'utilisateur connecté peut également poster des commentaires dans un espace dédiée. Ainsi, il peut interagir avec les autres utilisateurs du site. A noter que l'utilisateur ne peut supprimer que ses propres commentaires. Il aurait été intéressant de donner la possibilité à l'utilisateur de modifier son propre commentaire ou de répondre précisément à un commentaire spécifique (Système de reply-to). Ci-joint des visuels des interfaces de connexions présentes sur l'application.



The image displays two side-by-side screenshots of a web application's user interface, both featuring a light yellow background and a dark blue border.

The left screenshot shows a user profile or account management page. At the top, it displays "Nom : spaniol". Below this is a "Déconnexion" button. Further down, there are two input fields for password management: "Ancien mot de passe :" and "Nouveau mot de passe :". A "Changer le mot de passe" button is positioned at the bottom of this section.

The right screenshot shows a login and registration page. It includes three input fields: "Email :", "Pseudo :", and "Mot de passe :". A "Connexion" button is located below the password field. At the bottom, there is a text prompt "Vous possédez déjà un compte ?" followed by a "Se connecter" button.

FIGURE 4 – Visuels des formulaires d'authentification/création de compte

6 Contenu

Afin que vous compreniez l'implémentation qui a été faite, nous allons donc vous expliquer les différentes parties d'édition et de visualisation d'un animal.

6.1 Ajout d'un animal

6.1.1 Frontend

Un des objectifs demandés de ce sujet était de pouvoir laisser l'utilisateur enregistrer un animal et que l'on puisse le visualiser sur une carte interactive. Nous avons donc implémenté un formulaire de création d'un animal. Celui-ci est composé de plusieurs caractéristiques que vous pouvez visualiser sur la capture ci-jointe. Comme nous pouvons le voir, la première caractéristique



FIGURE 5 – Formulaire de création d'un animal

à obtenir est le nom de l'animal, l'utilisateur rentre le nom qu'il souhaite et à chaque changement détecté dans l'input² un processus de scrapping s'exécute et ressort les 5 liens wikipédia correspondant au nom souhaité. Une fois le lien sélectionné par l'utilisateur, nous obtenons une base de travail sur laquelle nous allons pouvoir récupérer la description et/ou l'image si l'utilisateur le souhaite. Cependant, l'utilisateur peut choisir aussi d'écrire sa propre description ou d'importer sa propre image. Auquel cas, le lien scrappé sera uniquement présent dans la pop-up de l'animal sur la carte et n'aura qu'une utilisation purement informative. Si l'utilisateur décide d'y inscrire ses propres informations, des zones dédiées apparaissent pour y remplir leurs informations. La sélection de la catégorie de l'animal enregistré est

2. Événement Keyup dans Javascript

disponible avec l'icône correspondante sur le côté. Et pour finir le formulaire

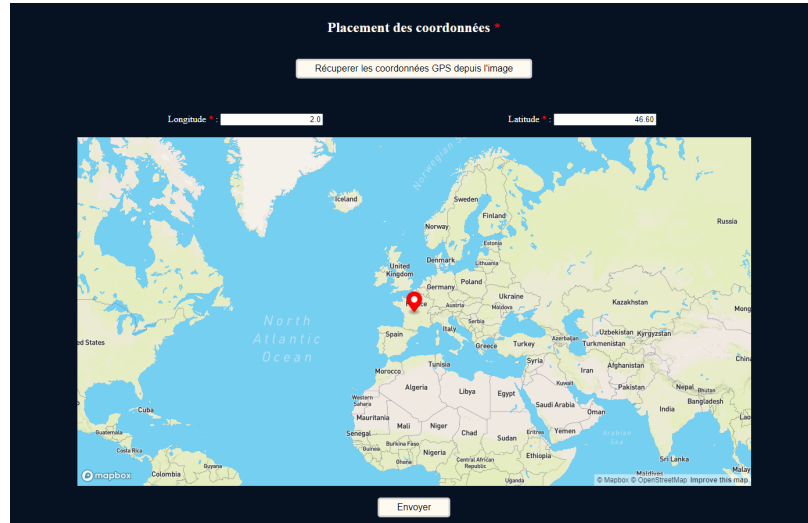


FIGURE 6 – Ajout de localisation pour formulaire de création d'un animal

d'ajout d'un animal, nous demandons à l'utilisateur de remplir les coordonnées souhaitées afin de le placer sur la carte, il est possible de récupérer les coordonnées GPS de la photo si elle en contient il faut sélectionner un lien wikipedia ou upload votre propre image et ensuite cliquer sur le bouton "Récupérer les coordonnées GPS" afin de les obtenir. Il est également possible de simplement cliquer sur la carte pour sélectionner une localisation.

6.1.2 Backend

Afin de réaliser cet ajout à la base de données, nous récupérons les valeurs POST après que le formulaire soit rempli et que les informations soit également récupérées en JavaScript pour le scrapping.³ Suite à cela, nous stockons la requête SQL pour insérer un animal dans une variable puis nous préparons la requête à l'exécution à l'aide de la fonction `prepare()` de PHP. Enfin, nous exécutons la requête en y ajoutant les valeurs POST correspondantes grâce à la fonction `array()`. En opérant de cette manière, nous évitons

3. Le submit du formulaire est bloqué provisoirement en JS le temps d'effectuer les différents calls API au cas par cas. Suite à cela, une requête AJAX s'ouvre pour envoyer un FormData mis à jour contenant toutes les informations dont nous avons besoin en backend

de créer des failles de sécurités sur notre site⁴

```
$sql = "INSERT INTO 'Animaux'('nom', 'file_name', 'file_data', 'link', 'longitude', 'latitude', 'categorie', 'description', 'wikilink', 'scrapped')
VALUES (:nom, :file_name, :file_data, :link, :longitude, :latitude, :categorie, :description, :wikilink, :scrapped)";
$res = $this->dbh->prepare($sql);
$value = $res->execute(array(":nom"=>$name, ":file_name"=>$file_name, ":file_data"=>$file_data, ":link"=>$link, ":longitude"=>$longitude, ":latitude"=>$latitude,
":categorie"=>$categorie, ":description"=>$description, ":wikilink"=>$wikilink, ":scrapped"=>$scrapped));
```

FIGURE 7 – Fonction pour ajouter un animal à la base

6.2 Galerie des animaux

Il nous a été suggéré de réaliser une page de visualisation de l'ensemble des animaux stocké dans notre base de données. Nous avons donc suivi ces indications et voici le résultat ci-dessous :



FIGURE 8 – Galerie des animaux

Comme vous pouvez le constater lors du survol d'une image, l'utilisateur peut accéder à trois fonctionnalités différentes. Un accès à la page détail d'un animal, à la page de modification d'un animal et à la suppression d'un animal. Nous avons également ajouté un bouton pour effectuer un tri d'animaux en fonction de leur catégorie.

6.2.1 Détail d'un animal

Une visualisation des données de l'animal semblait nécessaire, nous avons donc implémenté une page de détail à titre informatif sur les données de l'animal correspondant. Sur cette page, nous pouvons apercevoir l'image qui a été récupéré ou upload, le nom de l'animal, la description et la localisation

4. Nous pensons à l'injection SQL

The image shows a close-up of a bee on a purple flower. Below the photograph is a map of Europe with a red dot indicating the location of the species. The map is titled 'Localisation' and shows the distribution of the species across the continent.

L'implémentation de ce code est plutôt simple grâce au framework TWIG, l'affichage des sections correspondantes se fait via le parcours du tableau qui va lire la table MySQL.

Université de CAEN 13 2021-2022

Ecrire votre commentaire

Soumettre le commentaire

4 commentaires

Valentin
Ca marche
2020-07-22

John Doe
Thank you for taking the time to write this article, I really enjoyed reading it! Thank you, David!
2020-07-22

David Adams
It's good to hear that you enjoyed this article.
2020-07-22

Michael
I appreciate the time and effort you spent writing this article, good job!
2020-07-22

FIGURE 10 – Section commentaire

6.2.2 Modification d'un animal

Pour modifier un animal, nous avons récupéré le formulaire de la page d'ajout d'un animal et nous l'avons pré-rempli avec les données déjà enregistrées, pour la réécriture des données si l'utilisateur décide de modifier les données. Nous analysons la variable POST afin de repérer quels sont les variables ayant été modifiées. Nous ne récupérons que les variables modifiées et nous les comparons avec les données déjà existantes de l'animal sélectionné. On assigne les variables différentes des précédentes dans un tableau pour les insérer dans la requête SQL "UPDATE" afin de modifier les données de l'animal correspondant.

Pour la suppression d'un animal, nous avons créé une simple fonction qui se lance lorsque le bouton est activé et qui exécute une requête SQL "DELETE" pour supprimer l'animal correspondant et toutes ses données avec. Cette action est irréversible.

7 Conclusion

7.1 Objectif atteint ?

L'application web fait le travail demandé : Enregistrer et visualiser des animaux sur une carte interactive, avec un aspect communautaire. Cependant, nous aurions pu peut-être accentuer l'aspect communautaire de l'application. Cependant les principales fonctionnalités sont présentes donc nous pouvons dire que l'objectif est atteint. Avec des fonctionnalités secondaires intéressantes pour l'utilisateur, nous pouvons même dire que nous sommes allé plus loin que l'objectif attendu. Concernant l'aspect purement technique : Le site est entièrement fonctionnelle et responsive.

7.2 Pistes d'améliorations

Un projet est toujours améliorable, celui-ci n'est pas exempt à la règle. Nous aurions pu réaliser un aspect graphique se rapprochant plus de la faune et la flore. Nous aurions pu améliorer l'user-interface et l'user-experience du site web, notamment au niveau des formulaires d'ajout et de modification des animaux. De plus, le scrapping aurait pu être réalisé dans la partie backend du site web (En PHP donc). Malgré tout, le scrapping fait en Javascript est tout à fait convenable et permet plus de liberté en matière de call API et d'événements. Nous aurions pu également ajouté un système de rôles plus complexe (avec par exemple une modification des animaux que l'utilisateur lui-même a ajouter. Actuellement, un utilisateur peut modifier absolument tous les animaux qui sont présents sur le site.) L'insertion d'une fonctionnalité de réponse à un commentaire particulier aurait pu être apprécié même si le système actuelle reste tout à fait convenable. Enfin le dernier point serait la recherche d'images. Nous avons utilisé la base de données d'images de Wikipédia Commons pour rechercher et inclure une image à un animal choisi. Cependant les images contenues dans cette base de données ne contiennent pas d'informations de géolocalisation dans la majorité des cas. Il aurait donc fallu choisir une autre source de données.