Sommaire

Réflexions initiales technologiques sur le sujet

Frontend

- *CSS :* SASS
- *Javascript :* Typescript
- *Framework WebGL :* THREE.js (en fonction du temps disponible)
- *Bundler :* Vite

Pour le **CSS**, plusieurs choix sont disponibles ... Cependant, je souhaite faire quelque chose de fin, sur lequel j'ai un contrôle total mais aussi sur une base que je maitrise. Je pense donc éviter **Bootstrap** et **Bootstrap** car je les maitrise mal, trouve que leurs composants sont trop standards, difficilement personnalisables et empêche de pouvoir experimenter correctement. De plus, cela nécessite de charger de grosse quantités de lignes de code pour un projet assez petit.

Je pense donc partir sur du **SASS** qui me permettra d'être fin dans mon design tout en gardant une certaine rapidité et flexibilité dans la manière d'écrire mon **CSS**.

Concernant le **Javascript**, une fois de plus, il n'y a que très peu de pages à faire ce qui signifie que je peut experimenter et complexifier tout ça si j'ai le souhaite et si le backend ne prend pas trop de temps.

Ne connaissant aucun framework JS, je ne vais pas me lancer dans cette aventure là. De plus, j'aime avoir le controle sur ce que je fais et souhaite expérimenter, je vais donc m'orienter sur du **Javascript** Vanilla ou du **Typescript**.

Typescript permet de solidifier la structure de son code et d'avoir une bonne compatbilité entre navigateurs. Je ne vois donc pas de mauvaises raisons de partir sur du **Typescript** hormis le fait que je vais devoir revoir quelques bases.

En fonction du temps que j'aurais à ma disposition, j'aimerais bien intégrer un peu de **THREE.js** dans le projet afin de faire une visite en direct du zoo dans un environement 3D. Je peut compter ça en bonus si je vois que j'ai du temps devant moi.

Afin de pouvoir optimiser les temps de chargement des pages et l'exécution de mon code, il pourrait être intéressant d'utiliser un Bundler comme **Vite** ou **Webpack** dans l'application.

Comme il s'agit d'un petit projet, ceci reste relativement optionnel. Ceci dit, c'est toujours bon à prendre.

Ne connaissant pas trop le fonctionnement des Bundler, je pense partir sur **Vite** notamment car c'est celui qui est recommandé par la documentation pour les applications utilisant **THREE.js** (voir : https://threejs.org/docs/index.html#manual/en/introduction/Installation)

De cette manière, si la documentation de **THREE.js** fait référence à des commandes dans **Vite**, je ne serais pas perdu.

Backend

- *Langage :* PHP
- *Framework :* Aucun
- *Librairie :* Aucun
- *Moteur de templates :* Twig
- *Base de données relationnelle :* MySQL
- *ORM :* aucun
- *Base de données non relationnelle :* MongoDB

Le seul langage backend que je connais actuellement est **PHP**. Mon choix se tourne donc naturellement vers cela. Le projet sera naturellement réalisé en modèle **MVC** notamment car c'est un standard côté backend et que maitriser ce modèle est important pour maitriser les différents frameworks **PHP**.

Maintenant se pose la question de si j'utilise un framework ou non. Dans un cadre purement professionnel, où le fait de rendre le produit en avance par rapport à la deadline imposée peut être vu correctement par le client et peut influencer l'image de l'agence, je serais parti sur **Symfony** car beaucoup de features demandées sont simplifiées par **Symfony** comme le Mailing (géré par un service) ou même l'interface d'administration (gérée par EasyAdminBundle).

Ceci dit, je suis en formation, j'aime faire tout moi-même et partir de zéro est aussi extrêmement formateur : on découvre pourquoi des personnes ont décidé de créer des frameworks et on comprend mieux pourquoi les choses ont été faites ainsi.

Je suis curieux de voir comment je vais pouvoir gérer le mailing et les différentes relations entre les roles administrateurs, vétérinaires etc ... et je pense que cela très enrichissant de voir comment faire cela.

Je vais donc partir sur du **PHP** Vanilla même si je sais que cela va complexifier la tache.

Ayant déjà eu à faire des projets en **PHP** Vanilla, je sais qu'il est compliqué d'obtenir des fichiers de template facilement lisible lorsque des balises **PHP** s'immissent dedans. Je vais donc utiliser **Twig** pour générer mes Vues.

Concernant les bases de données, il me faut une base relationnelle et une autre non relationnelle.

Pour la base relationnelle, je vais donc faire du **MySql** Vanilla notamment car c'est le plus répandu et ce que je maitrise le mieux.

Je n'utiliserais pas **Doctrine** car je considère qu'il est certes plus sécurisé que du SQL classique mais manque de certaines fonctionnalités (clause MONTH et WITH par exemple). Il faut alors constamment le surcoucher avec des extensions qui peuvent s'avérer être une source de problèmes et qui n'ont généralement pas des documentations bien étoffées.

Pour la base de données non relationnelle, je vais utiliser **MongoDB** car c'est la seule que je connaisse actuellement.

Configuration de votre environnement de travail

Stack de développement

- Sass
- Typescript
- Twig
- SleekDb

Prérequis pour cloner le projet

- Git
- NodeJs
- Composer
- Un serveur web de votre choix (WAMP, XAMP, LAMP)

Déploiement en local

Cloner le projet

Ouvrez votre terminal et lancez la commande :

```
git clone https://github.com/OkaniYoshiii/zoo-arcadia.git
```

Puis :

```
cd zoo-arcadia
```

Installation des dépendances

Sass / Typescript

```
npm install
```

Twig / SleekDb

```
composer install
```

Configuration BDD

Créez un fichier nommé **config.local.php** dans le dossier **config**.

Dans ce fichier, rentrez le code suivant :

```
<?php
        // Variables et constantes spécifiques à un environnement particulier
        // Exemple : l'URL du site, les identifiants de la BDD ...

        // Nom de domaine de votre site
    // http://localhost en local
        define('SITE_URL', 'https://monsite.com');

        // Informations de connexion à votre base de données
        define('DB_DSN', 'mysql:host=localhost;dbname=myDb');
        define('DB_USER', 'user');
        define('DB_PWD', 'pwd');

        // Uri pour se connecter à votre base de données MongoDb
        // Si vous utilisez MongoDbAtlas, cela devrait ressembler à :
        define('MONGODB_URI',
'mongodb+srv://<db_user>:<db_password>@<cluster_name>.h7ubp.mongodb.net/?
retryWrites=true&w=majority&appName=<cluster_name>');

        // Chaine de caractères aléatoire unique à votre projet utilisée pour
sécuriser les     mots de passe
        // Le secret doit être de longueur 14 (112 bits) : Recommandation du
NIST       (National Institute of Standards and Technology)
        // Voir : https://en.wikipedia.org/wiki/Pepper_(cryptography)
    define('APP_SECRET', 'HXTAeiDQWLADBm');
```

[...] et modifiez :

- *SITE_URL :* le nom de domaine de votre site

•*DB_DSN :* les informations liées à votre base de données

•*DB_USER :* le nom d'utilisateur de votre base de données

•*DB_PWD :* le mot de passe de votre base de données

•*APP_SECRET :* un chaine de caractères aléatoires d'une longueur de 14 caratères minimum

Une fois cela fait, importez le fichier *arcadia_db* présent à la racine du projet dans votre SGBDR.

Optionnel :

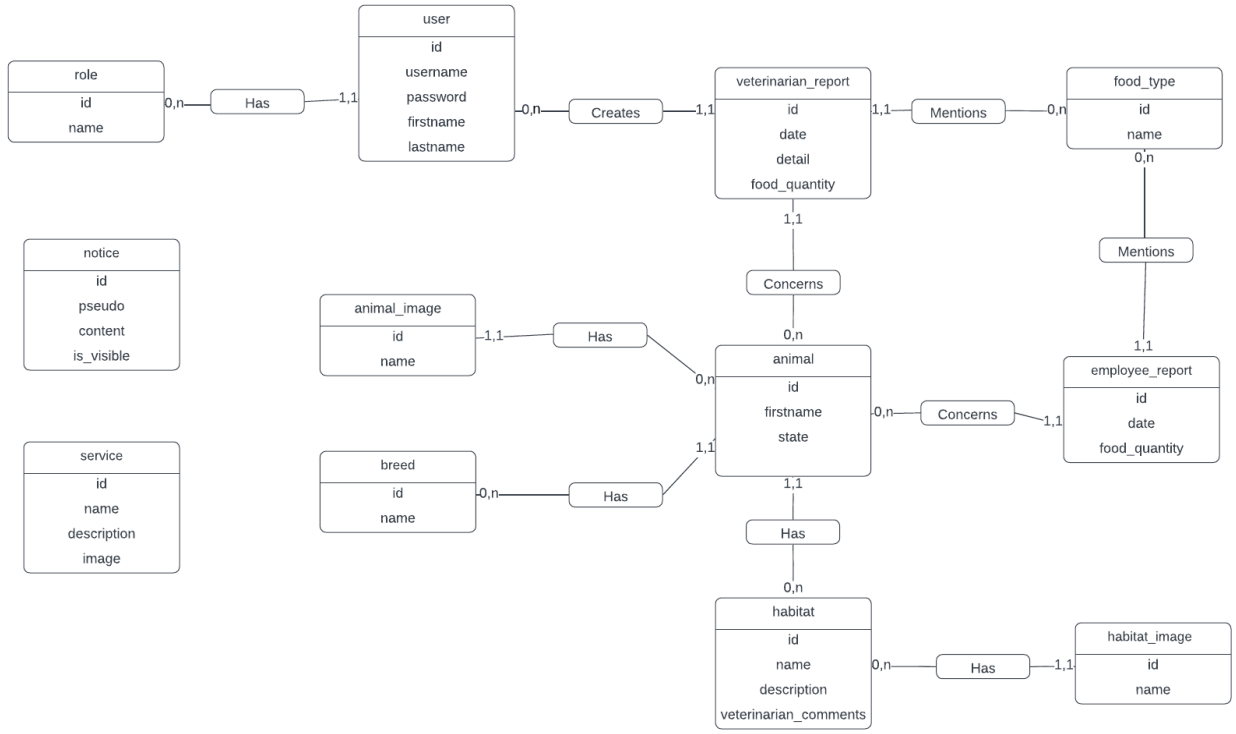
Une fois la structure de la base de données établie, vous devriez pouvoir créer des fixtures en lançant la commande :

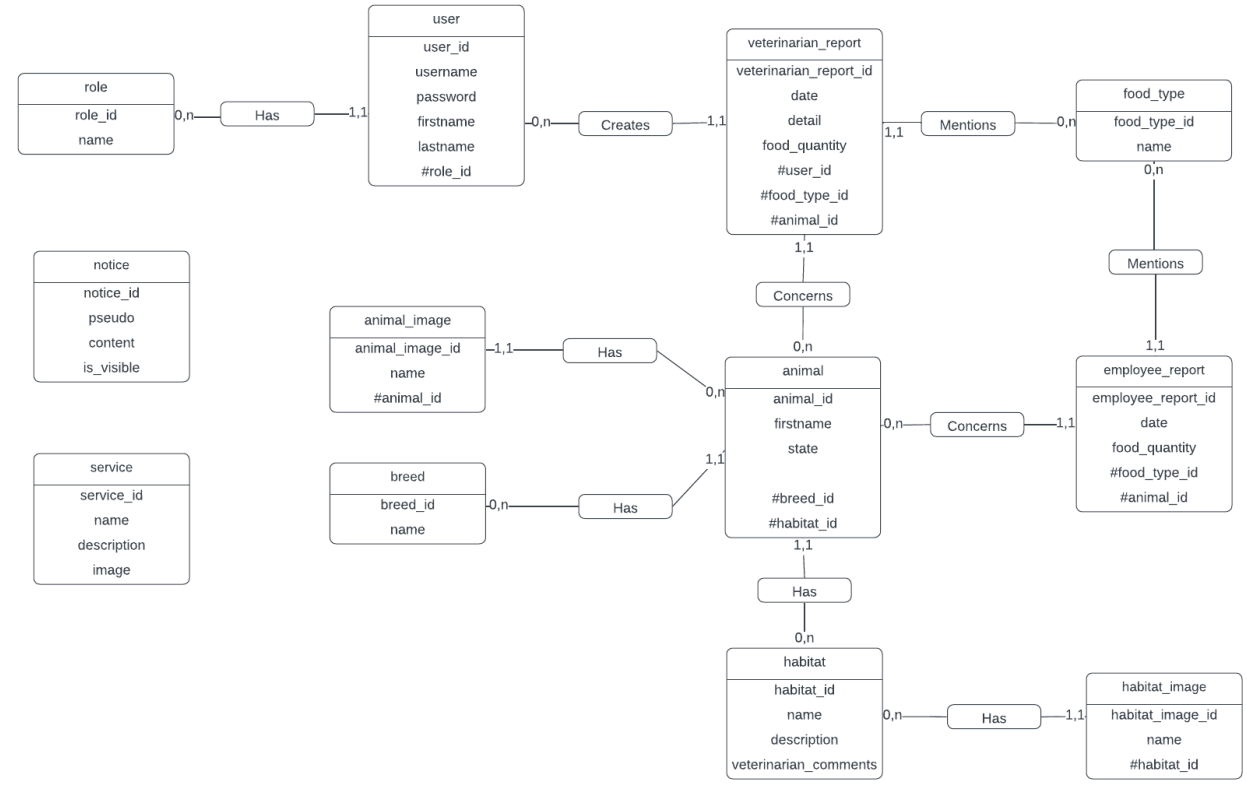
```
php cmd/fixtures/fixtures.php
```

!!! ATTENTION !!!

Ceci efface les données des tables et les remplace par vos fixtures, lancez cette commande uniquement si vous êtes sur de ne pas perdre de données importantes.

Modèle conceptuel de données (ou diagramme de classe)





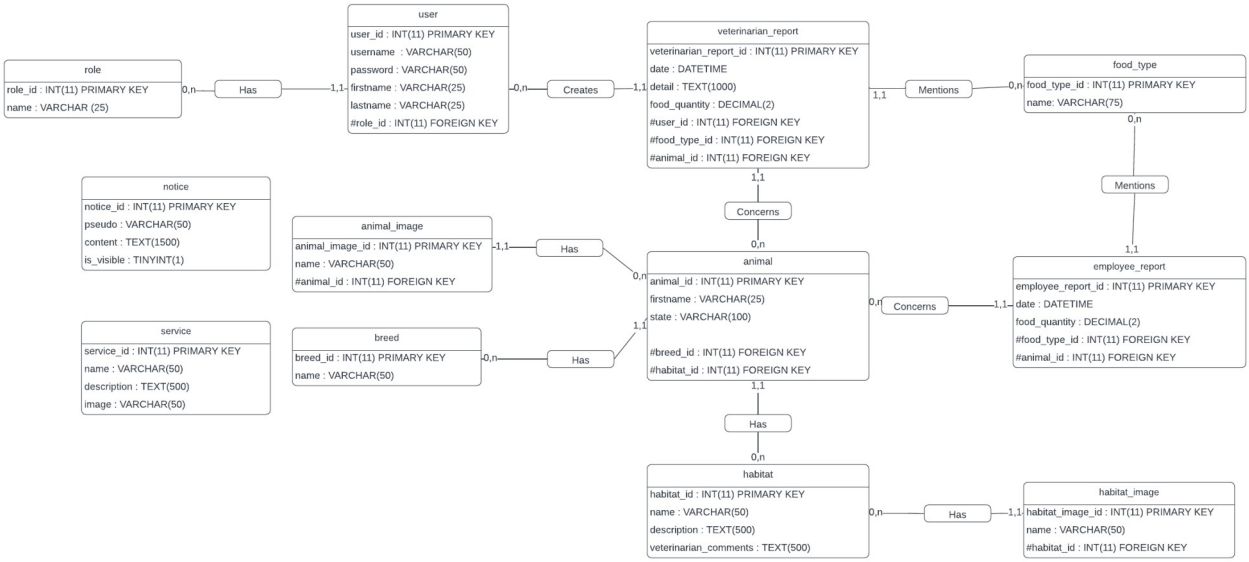
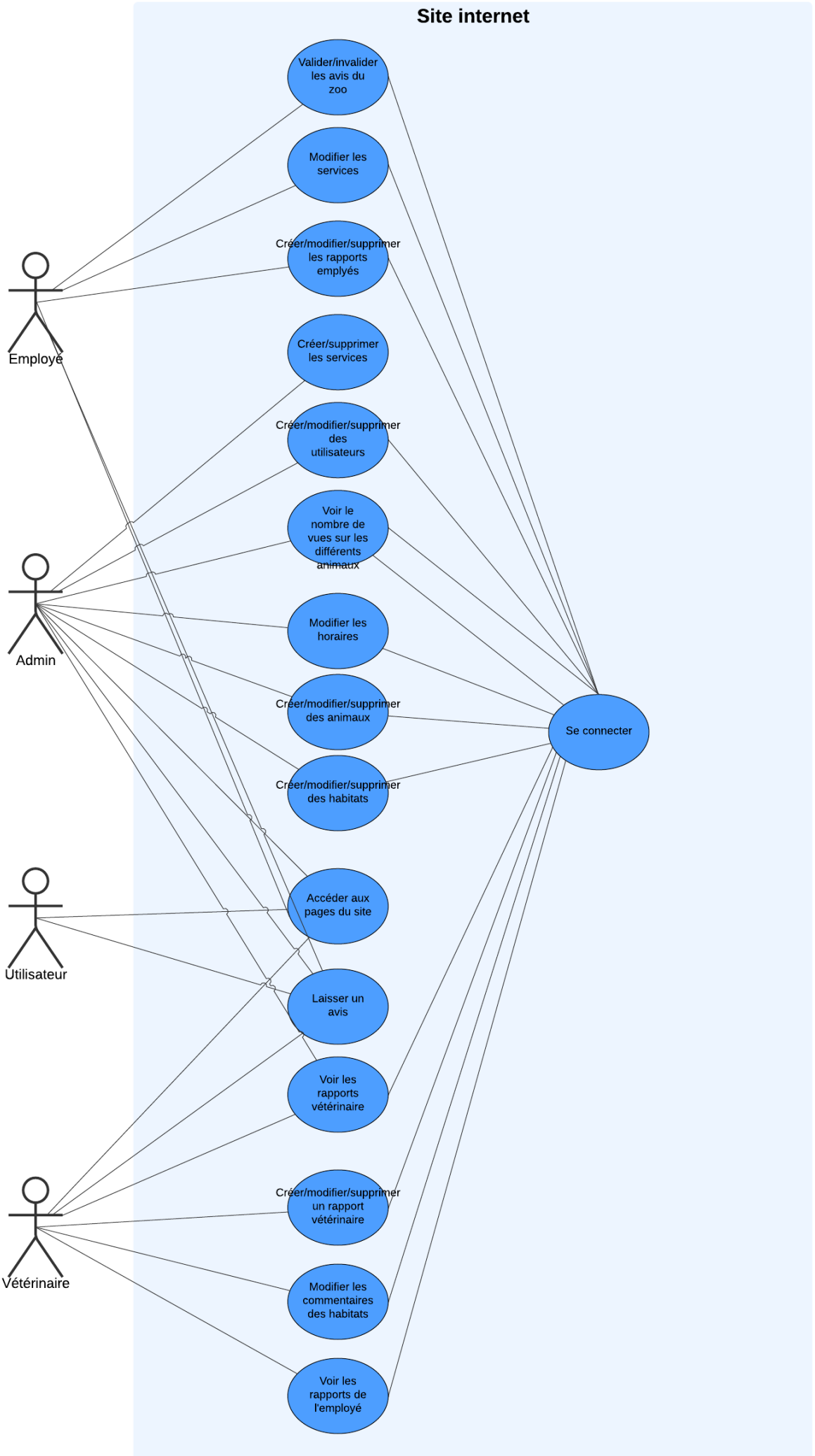


Diagramme d’utilisation ainsi que le diagramme de séquence


Site internet
Valider/invalider les avis du zoo
Modifier les services
Créer/modifier/supprimer les rapports emplyés
Créer/supprimer les services
Créer/modifier/supprimer des utilisateurs
Voir le nombre de vues sur les différents animaux
Modifier les horaires
Créer/modifier/supprimer des animaux
Créer/modifier/supprimer des habitats
Accéder aux pages du site
Laisser un avis
Voir les rapports vétérinaire
Créer/modifier/supprimer un rapport vétérinaire
Modifier les commentaires des habitats
Voir les rapports de l'employé
Se connecter
Employé
Admin
Utilisateur
Vétérinaire

Documentation du déploiement de votre application expliquant votre démarche ainsi que les différentes étapes.

J’ai donc d’abord souscrit à un abonnement mensuel pour un hébergeur web nommé « Hostinger ». Cet hébergement est mutualisé et est déjà préconfiguré pour faire tourner un serveur web (Apache dans ce cas) avec PHP. Un bon nombre d’extensions sont déjà installées pour ce langage et je n’ai eu qu’a activer l’extension « mongodb » via leur interface d’administration.

Concernant l’installation des dépendances, Composer et Nodejs sont déjà pré-installés sur le serveur. L’installation était donc très simple.

Pour les bases de données, phpMyAdmin est MySQL étaient déjà installés et configurés. J’ai donc du uniquement créer une base de données avec nom d’utilisateur et mot de passe puis importer ma base de données locale sur le serveur.

Pour MongoDB, ce dernier n’était pas installé sur le serveur et étant sur un serveur mutusalisé, impossible de l’installer moi-même. J’ai donc du utiliser MongoDB depuis le Cloud grâce à MongoDb Atlas. Pour cela, il me suffisait juste de renseigner l’adresse IP du serveur sur l’interface d’administration MongoDB Atlas.