Documentation Technique

Introduction

Cette documentation technique couvre les aspects clés du projet, y compris la réflexion initiale sur le choix des technologies, la configuration de l'environnement de travail, et les étapes de déploiement. Le projet a été entièrement réalisé avec Docker, sans utiliser de librairies tierces.

Table des Matières

Réflexion Initiale Technologique

Choix des Technologies

- **Docker** : Utilisation de conteneurs pour assurer un environnement de développement cohérent et simplifier le déploiement.
- **PHP avec Apache** : Pour servir les pages web dynamiques.
- **MySQL** : Base de données relationnelle pour stocker les données de l'application.

Justification

- **Portabilité** : Docker permet de s'assurer que l'application fonctionne de la même manière sur différents environnements (développement, test, production).
- **Isolation** : Chaque service (Apache, MySQL) tourne dans son propre conteneur, ce qui améliore la sécurité et la gestion des ressources.
- **Facilité de Déploiement** : Docker Compose simplifie la gestion et le déploiement des multiples conteneurs nécessaires à l'application.

Configuration de l'Environnement de Travail

Prérequis

- **Docker** : Installer Docker sur votre machine. Guide d'installation Docker
- **Docker Compose** : Installer Docker Compose. Guide d'installation Docker Compose

Configuration des Fichiers

Dockerfile

Le Dockerfile définit les instructions pour construire les images Docker pour MySQL et PHP avec Apache.

docker-compose.yml

Le fichier docker-compose.yml définit les services pour l'application, notamment WAMP (Windows, Apache, MySQL, PHP) et MySQL.

.htaccess

Le fichier .htaccess pour la gestion des erreurs personnalisées.

Étapes de Déploiement

1. **Cloner le dépôt** : Clonez le dépôt contenant le code source et les fichiers de configuration.

   ```
   git clone <url-du-depot>
   cd <nom-du-depot>
   ```

2. **Créer le fichier docker-compose.yml** : Créez un fichier docker-compose.yml avec le contenu ci-dessus.
3. **Créer le Dockerfile** : Créez un fichier Dockerfile à la racine du projet avec le contenu ci-dessus.
4. **Configurer les fichiers nécessaires** :
   - Créez le dossier `www` pour le contenu web.
   - Créez le dossier `db-init` pour les scripts de base de données d'initialisation.
   - Assurez-vous que les fichiers d'erreur (ex. `error404.html`) et le fichier `.htaccess` sont en place.
5. **Démarrer les services Docker** : Utilisez Docker Compose pour démarrer les services.

   ```
   docker-compose up -d
   ```

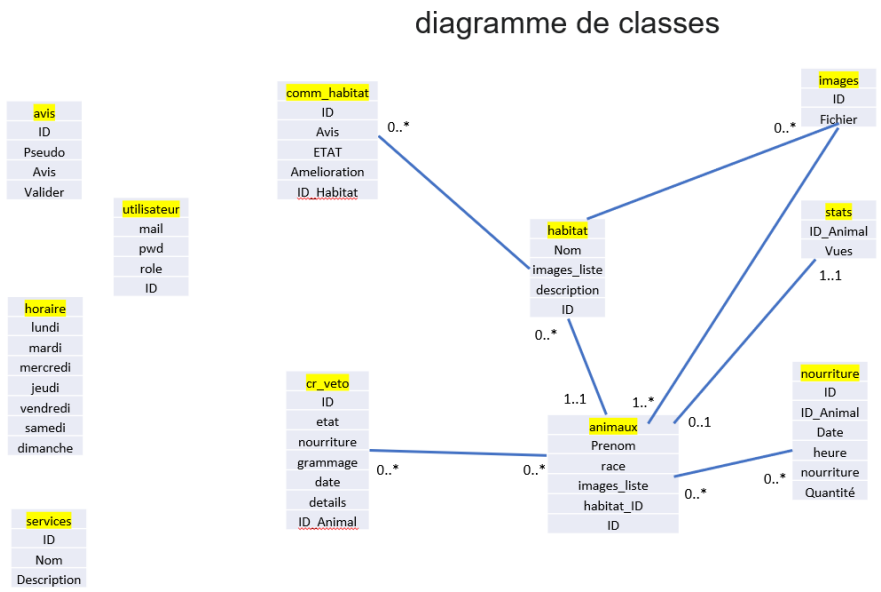
6. **Accéder à l'application** : Ouvrez votre navigateur et accédez à `http://localhost:8080` pour voir l'application en cours d'exécution.

Vérification

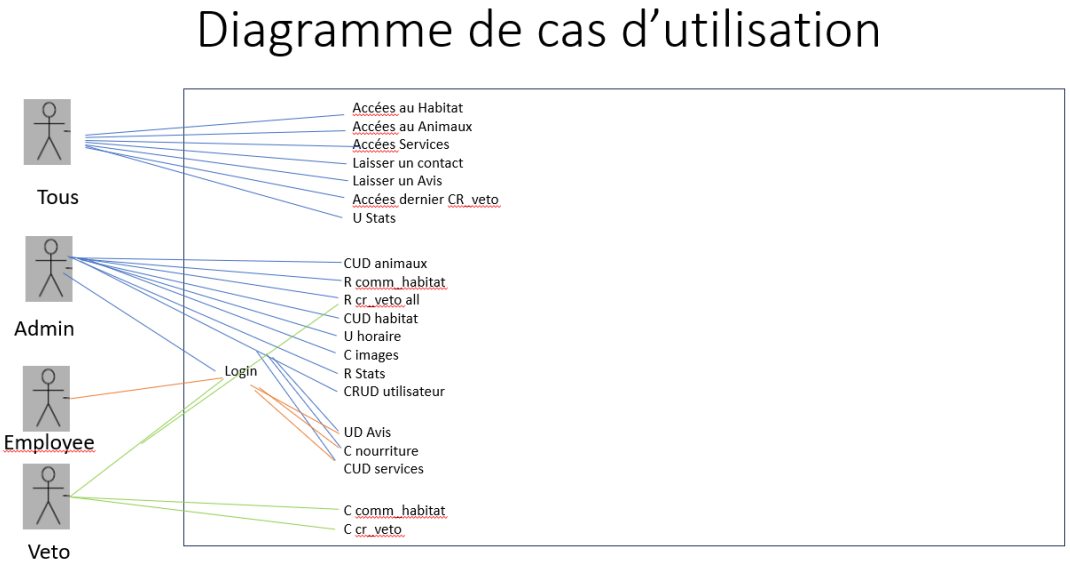
- **Accès à la base de données** : Vérifiez que MySQL est en cours d'exécution et accessible sur le port 3306.
- **Fonctionnalité du site web** : Assurez-vous que le site web est fonctionnel et que les erreurs personnalisées sont gérées correctement.
- **Logs et monitoring** : Vérifiez les logs des conteneurs pour détecter toute erreur ou problème potentiel.

Diagrammes

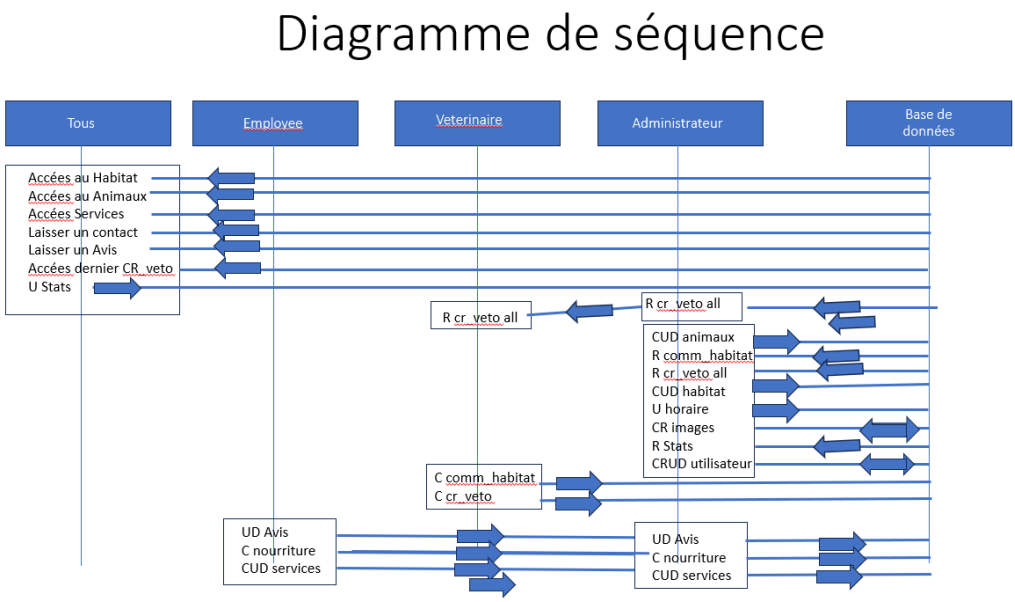
- **Diagramme de classe**



- **Diagramme d'utilisation**



- **Diagramme de séquence**



Documentation du Déploiement

Prérequis

1. **Compte Heroku** : Assurez-vous d'avoir un compte Heroku. Vous pouvez créer un compte gratuit sur heroku.com.
2. **Heroku CLI** : Installez l'interface en ligne de commande Heroku (Heroku CLI). Téléchargez-la depuis Heroku CLI.
3. **Git** : Assurez-vous que Git est installé sur votre machine. Si ce n'est pas le cas, téléchargez-le depuis git-scm.com.

Initialiser un Dépôt Git

1. Ouvrez un terminal ou une invite de commande.
2. Naviguez jusqu'à la racine de votre projet.
3. Initialisez un dépôt Git :

   ```
   git init
   ```

4. Ajoutez vos fichiers au dépôt :

   ```
   git add .
   ```

5. Faites un commit initial :

   ```
   git commit -m "Initial commit"
   ```

Créer une Application Heroku

1. Connectez-vous à Heroku via la CLI :

   ```
   heroku login
   ```

2. Créez une nouvelle application Heroku :

   ```
   heroku create <nom-de-votre-application>
   ```

   Remplacez `<nom-de-votre-application>` par le nom que vous souhaitez donner à votre application.

Déployer sur Heroku

1. Ajoutez Heroku comme remote dans votre dépôt Git :

   ```
   git remote add heroku https://git.heroku.com/<nom-de-votre-application>.git
   ```

2. Poussez votre code sur Heroku :

   ```
   git push heroku main
   ```

   Si votre branche principale s'appelle `master` :

   ```
   git push heroku master
   ```

Ajout de SQL

1. Aller dans **Resources** sur le dashboard Heroku.
2. Chercher et ajouter **ClearDB MySQL**.
3. Aller dans **Settings** :
   - **Config Vars** => **Reveal Config Vars**

Récupérer le lien :

```
mysql://b0de82451659ae:664958ce@eu-cluster-west-01.k8s.cleardb.net/heroku_014671c5cd10699?reconnect=true
```

- **Username** : b0de82451659ae
- **Pwd** : 664958ce
- **Host** : eu-cluster-west-01.k8s.cleardb.net

4. Modifier PhpMyAdmin localement :
   - Aller dans `config.inc.php`.
   - Rajouter :

   ```
   $i++;
   $cfg['Servers'][$i]['host'] = 'eu-cluster-west-01.k8s.cleardb.net';
   $cfg['Servers'][$i]['user'] = 'b0de82451659ae';
   $cfg['Servers'][$i]['password'] = '664958ce';
   $cfg['Servers'][$i]['auth_type'] = 'config';
   ```

avant la fin du fichier.

5. Accéder à la plateforme PhpMyAdmin et gérer la DB + modifier les composants d’accès.

Conclusion

Cette documentation technique fournit une vue d'ensemble complète sur la manière dont le projet a été conçu, configuré et déployé. En utilisant Docker et Docker Compose, nous avons assuré un environnement de développement cohérent et reproductible, tout en facilitant le déploiement de l'application.